Università degli Studi di Trieste

Dipartimento di Ingegneria e Architettura

XXXIII Ciclo del Dottorato di Ricerca in
Ingegneria Industriale e dell'Informazione

Feasibility Investigation on Several Reinforcement Learning Techniques to Improve the Performance of the FERMI Free-Electron Laser

Settore Scientifico-Disciplinare: ING-INF/04 AUTOMATICA

Dottorando:
Niky Bruchon
[illegible]

Coordinatore:
Prof. Fulvio Babich
[illegible]

Supervisore di Tesi:
Prof. Felice Andrea Pellegrino
[illegible]

ANNO ACCADEMICO 2019/2020

Abstract

The research carried out in particle accelerator facilities does not concern only particle and condensed matter physics, although these are the main topics covered in the field. Indeed, since a particle accelerator is composed of many different sub-systems, its proper functioning depends both on each of these parts and their interconnection. It follows that the study, implementation, and improvement of the various sub-systems are fundamental points of investigation too. In particular, an interesting aspect for the automation engineering community is the control of such systems that usually are complex, large, noise-affected, and non-linear.

The doctoral project fits into this scope, investigating the introduction of new methods to automatically improve the performance of a specific type of particle accelerators: seeded free-electron lasers. The optimization of such systems is a challenging task, already faced in years by many different approaches in order to find and attain an optimal working point, keeping it optimally tuned despite drift or disturbances. Despite the good results achieved, better ones are always sought for. For this reason, several methods belonging to reinforcement learning, an area of machine learning that is attracting more and more attention in the scientific field, have been applied on FERMI, the free-electron laser facility at Elettra Sincrotrone Trieste. The research activity has been carried out by applying both model-free and model-based techniques belonging to reinforcement learning. Satisfactory preliminary results have been obtained, that present the first step toward a new fully automatic procedure for the alignment of the seed laser to the electron beam.

In the meantime, at the *Conseil Européen pour la Recherche Nucléaire*, CERN, a similar investigation was ongoing. In the last year of the doctoral course, a collaboration to share the knowledge on the topic took place. Some of the results collected on the largest particle physics laboratory in the world are presented in the doctoral dissertation.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AE-Dyna	Anchored Ensemble Dyna
AI	Artificial Intelligence
AWAKE	Advanced WAKefield Experiment
BE-OP-SPS	BEam OPerations on Super Proton Synchrotron
BESSY II	*Berliner ElektronenSpeicherring-gesellschaft für SYnchrotronstrahlung II*
CCD	Charge-Coupled Devices
CERN	*Conseil Européen pour la Recherche Nucléaire*
DESY	*Deutsches Elektronen-SYnchrotron*
DQN	Deep Q-Learning
EOS	Electro-Optical Sampling
FEL	Free-Electron Laser
FERMI	Free Electron laser Radiation for Multidisciplinary Investigations
GA	Gradient Ascent
GD	Gradient Descent
HZB	*Helmholtz Zentrum Berlin*
ICFA	International Committee for Future Accelerators
iLQR	Iterative Linear Quadratic Regulator
KIT	*Karlsruher Institut für Technologie*
LEIR	Low Energy Ion Ring
LQR	Linear Quadratic Regulator
MBRL	Model-Based Reinforcement Learning
MFRL	Model-Free Reinforcement Learning
ML	Machine Learning
NAF	Normalized Advantage Function
NPG	Natural Policy Gradient
PADReS	Photon Analysis Delivery and Reduction System
PER	Prioritized Experience Replay
PG	Policy Gradient
PSI	*Paul Scherrer Institut*
RBF	Radial Basis Function
RL	Reinforcement Learning
ROI	Region Of Interest
SAC	Soft Actor Critic
SASE	Self-Amplified Spontaneous Emission

SCADA	Supervisory Control And Data Acquisition
SLAC	Stanford Linear Accelerator Center
TRPO	Trust-Region Policy Optimizer
TT	Tip-Tilt
YAG	Yttrium Aluminium Garnet

Nomenclature

x	State
u	Control Input (Action)
r	Reward
π	Policy
f	State-transition Function
J	Objective Function
V	Value Function
Q	Action-value Function (or state weight matrix for iLQR problem)
γ	Discount Factor
α	Learning Rate
δ	Temporal Difference Error
θ	Vector of Parameters
ξ	State-input Trajectory
R	Cumulative Reward Function (or control input weight matrix for iLQR problem)
F	Fisher Information Matrix (or objective function for Gradient Ascent)
ζ	Gradient Ascent Step-size

Chapter 1

Introduction

Machine Learning (ML) is ubiquitous in science and technology nowadays. It is a field of computer science as well as an application of Artificial Intelligence (AI) able to outperform traditional computational methods in many areas. In the last decades, this approach has been able to face and solve a great variety of problems.

The first definition of machine learning belongs to Arthur L. Samuel in 1959 who defined it as a "field of study that gives computers the ability without being explicitly programmed". Nowadays, the most quoted definition is a more formal one. In 1997, Tom M. Mitchell defined that: "machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience" [1]. In the same book, the author defines also the algorithms studied in machine learning field: "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E".

Indeed, ML advantages concern e.g. the speeding up of tedious procedures by automatizing them, the capability to handle a huge amount of data, and the wide applicability of such techniques. The main difference between traditional methods and machine learning is the conceptual approach to problems. If a traditional algorithm takes some input and a code with some logic to produce an output, a machine learning algorithm takes an input and an output and returns some "logic" which can then be used to work with new input to produce the corresponding output. The logic generated is what makes it ML.

The machine learning field is usually divided into three main categories as shown in Figure 1.1. These sub-fields are Supervised Learning, Unsupervised Learning, and Reinforcement Learning. For each category, a brief overview is now proposed.

- **Supervised Learning:** Class of systems and algorithms that determines a predictive model using data points with known outcomes. The model is learned by training through an appropriate learning algorithm (e.g., linear regression, random forests, or neural networks) that typically works through some optimization routine to minimize a loss or error function.

- **Unsupervised Learning:** Class of systems and algorithms in which the users do not need to supervise the model. Instead, it allows the model to work on its own to discover regularity. It mainly deals with the unlabelled data.
- **Reinforcement Learning:** Class of systems and algorithms that learns by interacting with the environment. A reinforcement learning agent learns from the consequences of the actions it selects based on its past experiences (exploitation) and also on new choices (exploration), which is essentially trial and error learning. The signal that the agent receives is a numerical reward, which depends from the action, and the agent seeks to learn to select actions that maximize the accumulated reward over time.

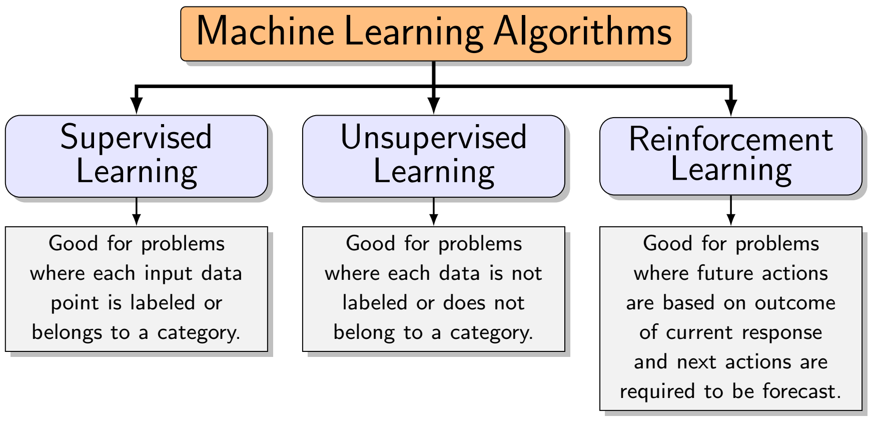
Machine Learning Algorithms
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Good for problems where each input data point is labeled or belongs to a category.
Good for problems where each data is not labeled or does not belong to a category.
Good for problems where future actions are based on outcome of current response and next actions are required to be forecast.

Figure 1.1: Machine learning categories.

1.1 Machine Learning in Particle Accelerators

Nowadays, ML techniques are technologically mature and can be applied to large and complex systems. This led many researchers involved in particle physics to start widely using such methods. Furthermore, several research centers supported the introduction of ML in particle accelerators through specific workshops and schools. In 2018, the mini-workshop on Machine

Learning Application for Particle Accelerators took place at Stanford Linear Accelerator Center (SLAC), while the Workshop on Machine Learning for Charged Particle Accelerators has been organized at *Paul Scherrer Institut* (PSI) in 2019. Finally, the *Helmholtz Zentrum Berlin* (HZB) held the Machine Learning Summer School in 2020. Consequently, ML has become each year a more valuable tool for particle accelerators. These research facilities are extensively used for basic research in particle physics and condensed matter physics. Particle accelerators systems are huge and complex; indeed, they are usually characterized by a great number of variables, devices, and uncertainties with the constant presence of noise. In recent years, machine learning techniques have led to many improvements and successful implementations in the field of particle accelerators, from automated alignment of various devices with beam to optimising different parameters in FELs, see for example [2–7].

One of the most important ML teams is at SLAC. Recently they provided an overview of the opportunities provided by the application of machine learning for particle physics [8].

1.1.1 Reinforcement Learning in Particle Accelerators

In the last years, the community of researchers involved in controlling and optimizing particle accelerators has paid particular attention to Reinforcement Learning (RL). The interest is in finding innovative and smart methods to realize a fully autonomously operated accelerator. The investigation concerns both the improvement of the performance and the simplification of routine operations. Even though the cost of these methods should not be underestimated (indeed they typically require a huge data set and long learning time), their capability to work on many parameters without the exact knowledge of a model may be of great benefit. Indeed, RL involves measuring state values and adjusting control variables to determine their influence on each other, thus learning a control strategy that also takes into account its effects in the future. In the long run, the hope is that it will be able to completely replace manual intervention.

Several facilities around the world already started investigating such an approach, the most important ones are SLAC, CERN, HZB, and Elettra Sincrotrone Trieste while at *Deutsches Elektronen-SYnchrotron* (DESY) a project on RL in collaboration with the *Karlsruher Institut für Technologie* (KIT) will start in late 2020. In particular, at SLAC the use of neural networks is advocated for modeling and control of particle accelerators, in a supervised fashion; moreover, some possible applications, in combination with RL methods are mentioned [9]. Furthermore, in [10] the authors use neural networks to define a policy and an FEL model, and then, enabling the interaction between the policy and the learned model, back-propagate the cost through the model network to the controller network; the whole study is conducted only through simulations. Satisfactory results have been collected also at CERN where some beam alignment problems have been investigated. Several algorithms have been successfully applied to the various systems involved in the project [11, 12]. At HZB, reinforcement learning has been introduced to face different optimization cases on the *Berliner ElektronenSpeicherring-gesellschaft für SYnchrotronstrahlung II* (BESSY II) light source by self-tuning the machine parameters [13]. A further investigation in controlling particle accelerators has been carried out at Elettra Sincrotrone Trieste, where various techniques have been applied to automatically improve the performance of seeded free-electron lasers [14–16]. Moreover, several promising results have been obtained within the collaboration with the University of Salzburg [17].

In the present document, a precise and complete presentation about the implementation of model-free and model-based reinforcement learning techniques on the Free Electron laser Radiation for Multidisciplinary Investigations (FERMI) facility is provided and the promising results acquired are shown and discussed in the following. In addition, also some information about the experiments carried out at the *Conseil Européen pour la Recherche Nucléaire* (CERN) in the first part of 2020 are here provided as part of the collaboration between the University of Trieste, Elettra Sincrotrone Trieste, and CERN.

1.2 Elettra Sincrotrone Trieste

Elettra Sincrotrone Trieste is an international research center specialized in generating high quality synchrotron and free-electron laser light applying it in materials and life sciences. The main assets of the research centre are two advanced light sources, the electron storage ring Elettra and the free-electron laser FERMI. These facilities enable the international community of researchers from academy and industry to characterize structure and function of matter with sensitivity down to molecular and atomic levels, to pattern and nanofabricate new structures and devices, and to develop new processes. An aerial overview of the site is provided in Figure 1.2.

In particular, this thesis focuses on the preliminary studies required to introduce reinforcement learning as a possible method to automatically improve the performance of FERMI. An exhaustive description of the main topic is therefore presented in the following section.

The entire project has been supported by Elettra Sincrotrone Trieste, which provided the access to a real free-electron laser light source allowing the direct interaction with it. In addition, the research center has contributed also with its know-how and the supervision of the expert physicists involved in the machine operations.

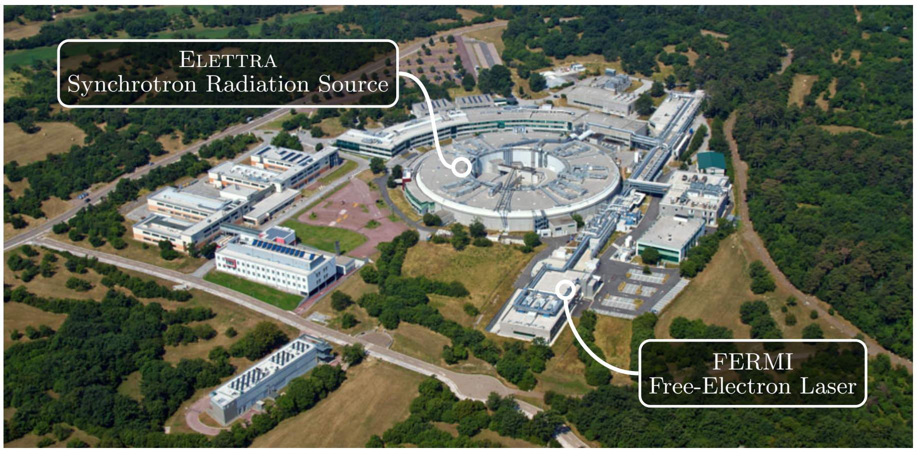

Figure 1.2: Elettra Sincrotrone Trieste, an aerial overview.

1.2.1 FERMI Automatic Performance Optimization

FERMI is the acronym for Free Electron laser Radiation for Multidisciplinary Investigations. Its lasing medium consists of very-high-speed electrons moving freely through a magnetic structure, hence the term free electron. The free-electron laser is tunable and has the widest frequency range of any laser type, currently ranging in wavelength from extreme ultraviolet to soft x-rays.

Nowadays one of the most challenging problems for the FEL facilities is the automatic maximization of the performance. The difficulty is due to the complexity and high sensitivity of the system. In fact, the factors that contribute to the light generation through the free-electron laser process are many and often depend on each other. Although several aspects have already been addressed in the past years using more traditional approaches [18], the introduction of learning methods would allow to face new and much more complex tasks without a priori knowledge of the system.

The approach proposed by RL perfectly fits these requests. Precisely, it concerns learning how to map states to actions in order to maximize a numerical reward signal. The fundamental characteristics of this particular learning category are the trial-and-error method and the delayed-reward. The first defines how the actions are selected to discover which one yields the most reward. The second concerns the effect of the selected action that may affect not only the immediate but also all the subsequent reward.

Accordingly, the present doctoral thesis investigates the automatic optimization of a seeded free-electron laser, i.e. FERMI, through several reinforcement learning techniques. The algorithms and the promising results recorded during experiments on the real machine are therefore shown in this dissertation.

1.3 Reinforcement Learning Experience at CERN

While the studies were ongoing at Elettra Sincrotrone Trieste, other research centers involved in particle physics started their investigation on reinforcement learning to automatically optimize several other particle accelerators. Some promising results have been recorded through reinforcement learning at the *Conseil Européen pour la Recherche Nucléaire* (CERN) which is a European research organization that operates the largest particle physics laboratory in the world. A schematic representation of the largest particle physics laboratory in the world operated by CERN is shown in Figure 1.3.

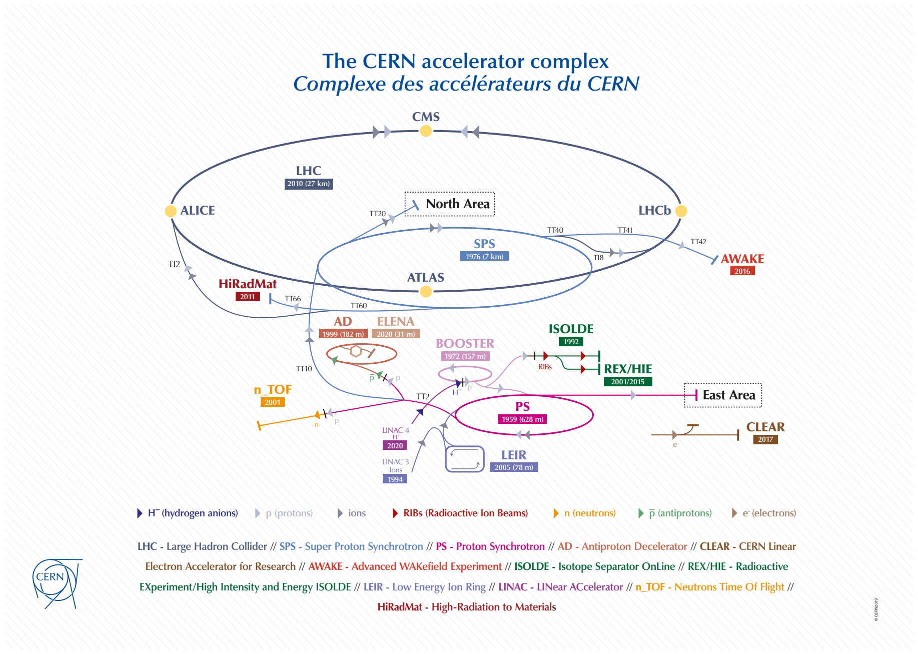
The CERN accelerator complex
Complexe des accélérateurs du CERN
CMS
LHC
2010 (27 km)
ALICE
North Area
LHCb
TT20
TT40
TT41
TT42
SPS
1976 (7 km)
TI8
AWAKE
2016
TI2
HiRadMat
2011
TT66
ATLAS
TT60
AD
1999 (182 m)
ELENA
2020 (31 m)
ISOLDE
1992
BOOSTER
1972 (157 m)
RIBs
REX/HIE
2001/2015
TT10
n_TOF
2001
TT2
East Area
PS
1959 (628 m)
CLEAR
2017
LINAC 4
H⁻
2020
LINAC 3
Ions
1994
LEIR
2005 (78 m)
H⁻ (hydrogen anions)
p (protons)
ions
RIBs (Radioactive Ion Beams)
n (neutrons)
p̄ (antiprotons)
e⁻ (electrons)
LHC - Large Hadron Collider // SPS - Super Proton Synchrotron // PS - Proton Synchrotron // AD - Antiproton Decelerator // CLEAR - CERN Linear Electron Accelerator for Research // AWAKE - Advanced WAKefield Experiment // ISOLDE - Isotope Separator OnLine // REX/HIE - Radioactive EXperiment/High Intensity and Energy ISOLDE // LEIR - Low Energy Ion Ring // LINAC - LINear ACcelerator // n_TOF - Neutrons Time Of Flight // HiRadMat - High-Radiation to Materials
CERN

Figure 1.3: CERN accelerator complex.

Improving the performance of particle accelerators is one of the main problems also at CERN, with several groups investigating in different directions. In particular, the reinforcement learning approach has been investigated by the BEam OPerations on Super Proton Synchrotron (BE-OP-SPS) group. The researchers involved in this investigation presented their preliminary results at the workshop on Machine Learning for Charged Particle Accelerators held during the second International Committee for Future Accelerators (ICFA) in 2019. Their presentation focused on the first steps towards reinforcement learning for automatic performance optimization of the Low Energy Ion Ring (LEIR) at CERN [11].

Given the common goals, a close collaboration between CERN, Elettra Sincrotrone Trieste, and the University of Trieste started in late 2019 and it is still ongoing. Thanks to this partnership, also other systems have been involved in the study. These are the Advanced WAKefield Experiment (AWAKE) and the linear accelerator 4 (Linac 4). In both new systems, the main goal is the finding of the optimal trajectory for the particle beams to improve their final outcome.

The mutual exchange of knowledge provided several improvements to reinforcement learning applications on particle accelerators. Indeed, some methods developed for FERMI have been successfully tested at CERN and vice-versa. A brief survey of the results obtained at CERN is provided in Chapter 5.

1.4 Outline

After this short introduction to artificial intelligence in the field of particle accelerators, the rest of this thesis is organized as follows - Chapter 2 presents a general overview of free-electron lasers, focusing on the target one, the FERMI FEL at Elettra Sincrotrone Trieste, where the study has been carried out. In Chapter 3, the basic information on the reinforcement learning approach and deployed algorithms are proposed. The experimental configuration and the achieved results are described and discussed in Chapter 4. A brief report about the collaboration with CERN is available in Chapter 5 where some interesting results are shown and discussed. Finally, conclusions are drawn in the last chapter.

Chapter 2

FERMI Introduction

A challenging and non-trivial problem concerns the optimization of the Free-Electron Lasers (FELs) setpoint. It usually requires a good knowledge of the system model and take quite a long time to achieve good results. In order to overcome these limitations, automatic learning approaches could be very proficient. It is from this idea that the research presented here originates.

In particular, the research consists of appraising the feasibility of different reinforcement learning methods to automatically optimize the performance of the FERMI free-electron laser at Elettra Sincrotrone Trieste. Even if the study concerns a specific light source, the approaches here presented could be easily adapted to other facilities and complex systems.

Before going deeply into reinforcement learning, a brief introduction of the FEL is given in this chapter. Furthermore, a more detailed description of FERMI light source is proposed, focusing on the relevant part of the FEL process. Finally, some information about the FERMI control system and the used software is reported.

2.1 Free-Electron Laser

A free-electron laser is a very flexible source of coherent radiation. It consists of a beam of relativistic electrons moving in vacuum inside a magnetic structure. The free-electron laser is tunable and has the widest frequency range of all laser types, currently ranging from microwaves to X-rays. Such light sources are used for cutting-edge research in material science, chemical technology, biophysical science, medical applications, surface studies, and solid-state physics.

In a FEL, an electron beam and a beam of electromagnetic waves collinearly travel through a magnetic structure called undulator, interact with each other, and generate amplified, coherent radiation. The interaction between the two beams modulates in energy the electron bunches. The energy modulation then is converted into a charge density modulation that produces microbunching in the longitudinal direction according to the radiation wavelength. Finally, the microbunched electron beam radiates in phase with the electromagnetic waves amplifying the radiation until the saturation is reached.

With a sufficiently high gain, the amplified radiation continues to interact with the electron beam traveling together with the radiation. Thus the system becomes a positive feedback system with exponential growth, called high-gain FEL. This configuration is called Self-Amplified Spontaneous Emission (SASE) and produces partially coherent radiation starting from an incoherent spontaneous emission. Even if at saturation the radiation generated has a high peak power and a good transverse coherence, its longitudinal coherence is affected by strong pulse-to-pulse spectral and temporal fluctuations.

An alternative is provided by seeded free-electron lasers (see Section 2.1.1) that inject an external optical laser together with the electron beam in the undulator. The energy modulation is then based on the seed laser properties, in particular on its high degree of longitudinal coherence. Such an improvement guarantees the generation of high-intensity radiation with both transverse and longitudinal coherence. In addition, the better density modulation of the electron beam requires a shorter magnetic path to amplify the output until the saturation.

The main elements involved in a free-electron laser are (Figure 2.1):

- **Electron beam.** It is a stream of bunched electrons emitted by a single source, that move in the same direction and at the same speed.
- **Linear accelerator (linac).** It is a sequence of electromagnetic structures accelerating the electrons.
- **Undulators.** It consists of series of periodic structures of dipole permanent magnets (undulators) to modulate the electron beam and amplify the radiation intensity.

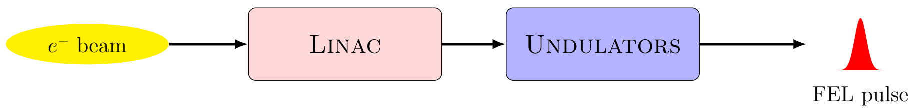

Figure 2.1: Free-electron laser simplified scheme.

2.1.1 Seeded Free-Electron Laser

In order to generate coherent radiation, seeded free-electron lasers [19–22] require an external optical laser pulse. The interaction with the seed laser modulates in energy the relativistic electron beam. The magnetic structure in which this process takes place is named modulator undulator. The energy modulation is then converted into a charge modulation by a dispersive magnet that is a magnetic chicane where the electrons follow a path length inversely proportional to their energy. The charge-modulated electron beam then generates a high coherent radiation passing through other undulators called radiators. A generic simplified scheme of a seeded free-electron laser is shown in Figure 2.2.

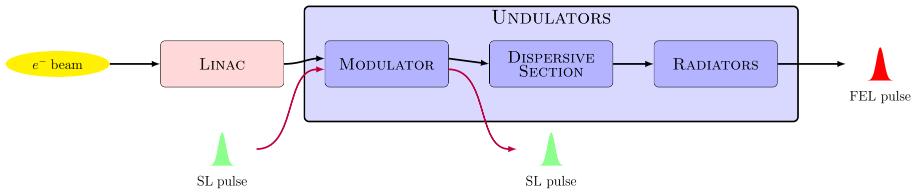

Figure 2.2: Seeded free-electron laser simplified scheme.

One of the most challenging operations required to properly set up a seeded FEL concerns the alignment of optical laser pulses and electron beams in the modulator. Hence, the better is their superimposition the more coherent and intense is the produced radiation.

In the present doctoral project, the problem of the alignment optimization has been faced using Reinforcement Learning algorithms. In order to reduce the complexity of the problem and to ease the implementation of the chosen RL methods, the overall problem has been simplified. In particular, the path of the electrons is kept constant while the seed laser trajectory is modified.

General information about the FERMI FEL is provided in Section 2.3.1 which describes the seed laser alignment procedure and presents the simplified scheme of the system.

2.2 FERMI Light Source

At Elettra two different particle accelerators are available for users to investigate material science and physical bio-sciences. The first one is ELETTRA, a 3^{rd} generation storage ring that has been in operation since October 1993, while the other is the *Free Electron laser Radiation for Multidisciplinary Investigations*, better known as FERMI. The FERMI FEL is an international user facility for scientific investigation of ultra-fast and high resolution processes with high brilliance photon pulses in the range from ultraviolet to soft x-ray. The three main parts of the overall facility are the linear accelerator (linac) where the electron beam previously extracted by the photo-injector is time-compressed and accelerated up to ∼1.5 GeV. Then the section deputed to the FEL radiation generation, and finally the experimental area where the FEL output is used by the various beamlines. Figure 2.3 shows an aerial overview of the Elettra site and the FERMI free-electron laser buildings.

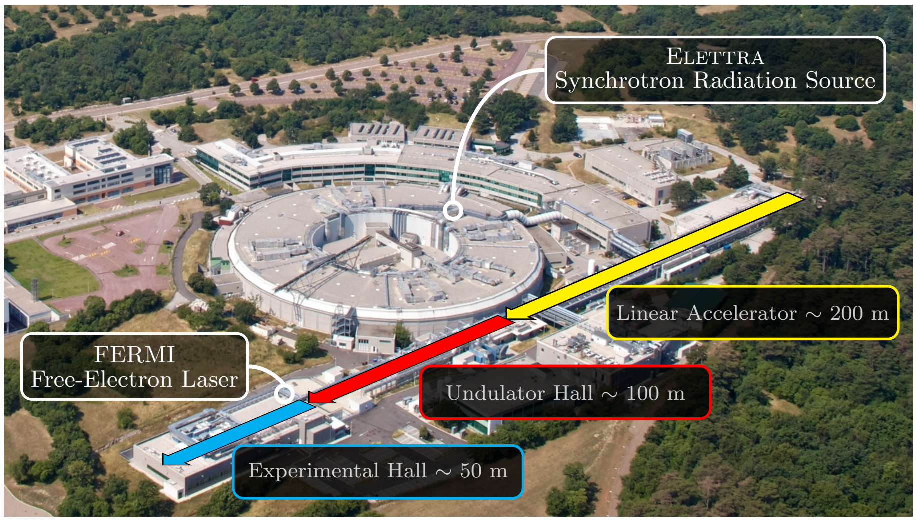

Figure 2.3: The Elettra research center and the FERMI free-electron laser.

FERMI is a seeded free-electron laser with two FEL undulator lines, FEL-1 and FEL-2, each covering a different spectral region in the wavelength range from 100 nm to 4 nm. FERMI provides users with photon pulses having unique characteristics such as high peak power, tunable wavelength, and variable polarization. In the following, further information is provided on the two FEL lines.

FEL-1. It is based on a single-stage high gain harmonic generation scheme with an UV seed laser covering the spectral range from ~100 nm down to 20 nm. The electron beam is energy modulated in the modulator undulator (MOD) by the seed laser, then the energy modulation is converted into a charge density modulation by a dispersive magnet and finally one selected resonant harmonic component is amplified in the radiators chain (RAD). Figure 2.4 shows a schematic representation of FEL-1.

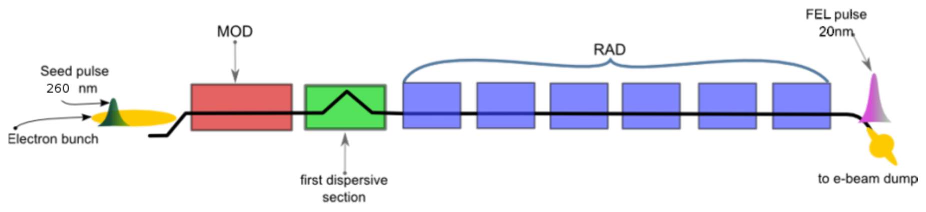

Figure 2.4: Schematic layout of FEL-1 in the high-gain harmonic generation configuration.

FEL-2. It is the cutting-edge FEL line in FERMI. In order to be able to reach the wavelength range from 20 nm to ∼4 nm starting from a seed laser in the UV, FEL-2 is based on a double cascade of high gain harmonic generation. The nominal layout uses a magnetic electron delay line in order to improve the FEL performance by using the fresh bunch technique. The first stage is similar to the high-gain harmonic generation scheme shown in Figure 2.4. The second stage is based on the same concept, but the seed is the radiation produced in the first stage. The two stages are separated by a delay line which lengthens the electron path with respect to the radiation (straight) path allowing to seed a "fresh" portion of the electron beam. A schematic representation of FEL-2 is proposed in Figure 2.5.

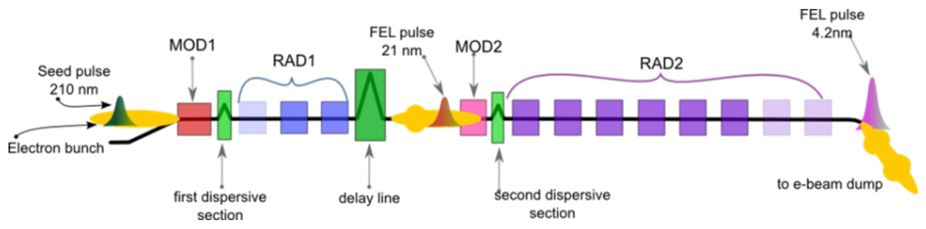

Figure 2.5: Schematic layout of FEL-2 in the fresh-bunch high-gain harmonic generation configuration.

In the experimental hall the produced photon beams are forwarded to one of the five experimental stations by the Photon Analysis Delivery and Reduction System (PADReS). This system is also in change of the characterization of the photons using several diagnostics such as photon beam position monitors, intensity monitors based on ionization gas cells, spectrometers, Yttrium Aluminium Garnet (YAG) screens and photodiodes.

2.3 FERMI Seed Laser Alignment

In a seeded free-electron laser, the temporal and transverse overlap of the electron and laser beams are the most critical parameters. While the temporal alignment between the two beams is regulated by a single actuator (a mechanical delay line), the transverse alignment depends on many parameters. In fact, the overlap of the two beams depends on the transverse position of each beam at the entrance and at the exit of the modulator undulator. In order to ease the tuning of these parameters and, at the same time, guarantee a steady FEL intensity, beam-based feedback systems [23] control, shot to shot, the trajectories of electrons and seed laser. While the electron beam trajectory is kept constant during the user operations, the position of the seed laser has to be readjusted in correspondence of changes of wavelength, which slightly modify the laser transverse profile, or due to small thermal drifts that affect the optical transport of the seed laser.

During standard operations, the horizontal and vertical transverse position and angle (pointing) of the laser beam inside the undulators is kept optimal by an automatic process exploiting the correlation of the FEL intensity with the natural noise of the trajectory [24]. Whenever the natural noise is not sufficient to determine in which direction to move the pointing of the laser, artificial noise can be injected. This method improves the convergence of the optimization, but the injected noise can affect the quality of the FEL radiation. This kind of model-free optimization techniques (ex. Gradient Ascent and Extremum Seeking [18, 25]) are widely used in FEL facilities, but have some intrinsic disadvantages:

1. the need to evaluate the gradient of the objective function, which can be difficult to estimate when the starting point is far from the optimum;
2. the difficulty to determine the hyper-parameters, whose appropriate values depend on the environment and the noise of the system;
3. the lack of "memory" to exploit the past experience.

Modern algorithms such as RL, are able to automatically discover the hidden relationship between input variables and objective function without human supervision. Although they usually require large amounts of data sets and long leaning time, they are becoming popular in the particle accelerator community thanks to their capability to work with no knowledge of the system.

In the next section, the seed laser alignment system at FERMI is presented. However, due to its intensive use, FERMI availability for testing the algorithms is very limited. Therefore, some alternatives have been necessary at least for testing and defining the preliminary values of the hyper-parameters of the algorithms. Preliminary experiments have been conducted on another optical system which is part of the Electro-Optical Sampling station. Its description is provided in Section 2.3.2. In addition, part of the algorithm testing has been carried out using a "toy problem" implemented by a simulator, which allowed the debug and tune the code and the preliminary verification of the performance of the different approaches in a more efficient way and with no hardware implications.

2.3.1 FERMI Seed Laser Alignment System

In a seeded FEL, an initial seed signal, provided by a conventional high peak power pulsed laser, is temporally synchronized to overlap the electron bunches inside a first undulator section called modulator. In the transverse alignment process two Yttrium Aluminium Garnet (YAG) screens equipped with Charge-Coupled Devicess (CCDs) are properly inserted and extracted, in order to measure the electron beam transverse position before and after the modulator [26]. After the electron beam inhibition, using the same YAG screens, the seed laser position is measured and the correct positions of two Tip-Tilt (TT) mirrors are manually found by moving the coarse-motors in order to overlap the electron beam. The above destructive procedure (a screen has to be inserted) is repeated several times and, at the end, the screens are removed to switch on the FEL. A simplified scheme of the alignment set up is shown in Figure 2.6. After the above described coarse tuning, a further optimization is carried out by moving the tip-tilt mirrors to maximize the FEL intensity measured by the I_0 monitor. The working principle of this diagnostics is the atomic photo-ionization of a rare gas. The FEL photon beam, traveling through a chamber filled by a rare gas, generates ions and electrons, which are extracted and collected separately. From the resulting currents it is possible to derive the absolute number of photons per pulse, shot by shot.

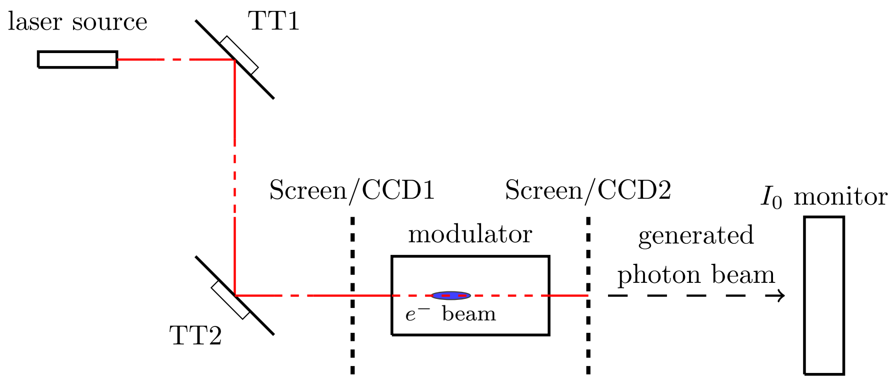

Figure 2.6: Simple scheme of the FERMI FEL seed laser alignment set up. TT1 and TT2 are the tip-tilt mirrors, Screen/CCD1 and Screen/CCD2 are the two removable YAG screens with CCDs and I_0 monitor is the employed intensity sensor.

2.3.2 Electro-Optical Sampling Alignment System

This considered optical system is a portion of the Electro-Optical Sampling (EOS) station, located upstream of the second line of the free-electron laser (FEL-2). An EOS station is a non-destructive diagnostic system designed to perform on-line single-shot longitudinal profile and arrival time measurements of the electron bunches using an UV laser [27–29]. Since the aim of this work is to control a part of the laser trajectory, the EOS process will not be explained in details, but rather the parts of the device used for our purposes will be introduced. The examined device, depicted in Figure 2.7, is a standard optical alignment system composed of two planar tip-tilt mirrors [30], each of which is driven by two piezo-motors (horizontal and vertical movements). The mirrors inclination determines a particular position of the laser spot on two CCDs. Each TT mirror performs fast rotations around the two axis in order to correct the path of the laser beam. The CCDs are not intercepting the laser beam thanks to the use of semi-reflecting mirrors.

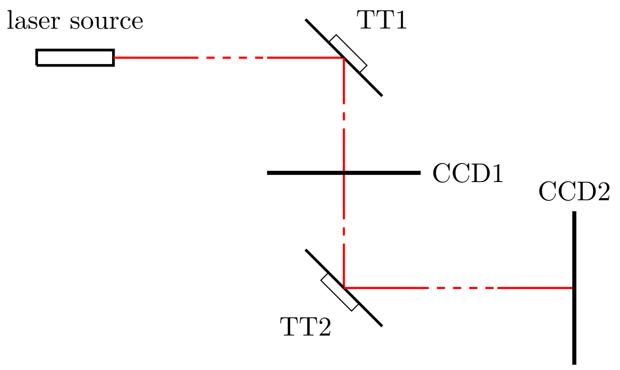

Figure 2.7: Simplified scheme of the EOS laser alignment set up. TT1 and TT2 are the tip-tilt mirrors while CCD1 and CCD2 are the CCDs.

The ultimate goal is to steer and keep the laser spot inside a pre-defined Region Of Interest (ROI) of each CCD. To achieve this result, a proper voltage has to be applied to each piezo-motor. The product of the two light intensities detected by the CCDs in the ROIs can be used as an evaluation criterion for the correct positioning of the laser beam. In particular, it can be interpreted as an AND logic operator, that is "true" when the laser is inside both of the ROIs.

2.4 Software Development Frameworks and Tools

This section briefly introduces the software used to control the FERMI facility and to implement the algorithms object of this work. In particular, the FERMI control system and the programming environment used to develop and deploy the RL algorithms are presented.

2.4.1 TANGO Control System

The FERMI control system is based on TANGO, an open source distributed framework for controlling any kind of hardware or software. Its functions and capabilities are similar to the ones of an industrial Supervisory Control And Data Acquisition (SCADA) system [31]. Elettra is participating in an international collaboration consortium with other European and International institutes to develop and maintain TANGO.

A TANGO-device server is a software component that interfaces to a physical or non-physical device implementing properties and methods in a unified environment. The system is organized hierarchically in order to interconnect complex systems. Any device is identified by a name with three fields in the format “domain/family/member". By remotely accessing the various devices, it is possible to read and possibly modify device attributes and execute commands.

Usually, the TANGO device server are programmed in the C/C++ programming language. However, to easily develop and implement algorithms for machine physics tests on FERMI, two interpreted programming languages are also used at client level: MATLAB and Python. Both are able to connect to the various devices in TANGO through the Matlab-binding [32] and the PyTango [33] libraries respectively.

2.4.2 Programming Languages and Software Tools

During this doctorate, both Matlab and Python have been used. Initially, the code has been developed in Matlab, better known from previous studies and experience such as the definition of a new FEL quality index [18].

Despite the existence of a Reinforcement Learning Matlab Toolbox, all the initial steps have been done writing the algorithms from scratch to better understand the Reinforcement Learning principle. The first published results [14] have been achieved with a custom implementation of the RL algorithm in Matlab, while in the extended version of that paper [15], the additional results presented have been obtained using Python.

The advantage of using Python is mainly due to the availability of tools: OpenAI Gym [34], and TensorFlow [35]. The former is a toolkit for developing and comparing reinforcement learning algorithms which have been used to create a simple simulator for testing the algorithm and defining the hyper-parameters. The latter is an end-to-end open-source machine learning platform. Both tools require Python and many tutorials, as well as examples, are available online.

Chapter 3

Optimization Methods

In the previous chapter, one of the most challenging problems in seeded free-electron lasers has been chosen to be solved through automatic methods. Precisely, it regards the maximization of the radiation intensity by setting the overlapping between the electron beam and the laser pulse.

This chapter consists of three main parts. Section 3.1 introduces reinforcement learning providing the basic tools to approach RL, while the following parts present specific categories of the RL methods. The model-free and model-based approaches are described in Section 3.2 and 3.3 respectively.

3.1 Reinforcement Learning

In RL, basically, data collected through experience are employed to select future inputs of a dynamical system [36, 37]. An *environment* is a discrete dynamical system whose model can be defined by:

$$x_{k+1} = f(x_k, u_k)$$

in which $x_k \in \mathcal{X}$ and $u_k \in \mathcal{U}$ respectively are the environment state and the external control input (the *action*) at the k-th instant; while f is the state-transition function. A controller, or *agent*, learns a suitable state-action map, also known as *policy* ($\pi(u_k|x_k)$), by interacting with the environment through a trial and error process. For each chosen action $u_k \in \mathcal{U}$, in state $x_k \in \mathcal{X}$, the environment provides a *reward* $r(x_k, u_k)$. This interaction between agent and environment is shown in Figure 3.1.
The aim of the learning process is to find an optimal policy π^* with respect to the maximization of an objective function J, which is a design choice.

It could be useful for readers to provide a short commented list of the fundamental and basics concepts of reinforcement learning.

- **Agent:** It is a controller that learns a policy by interacting with the environment.

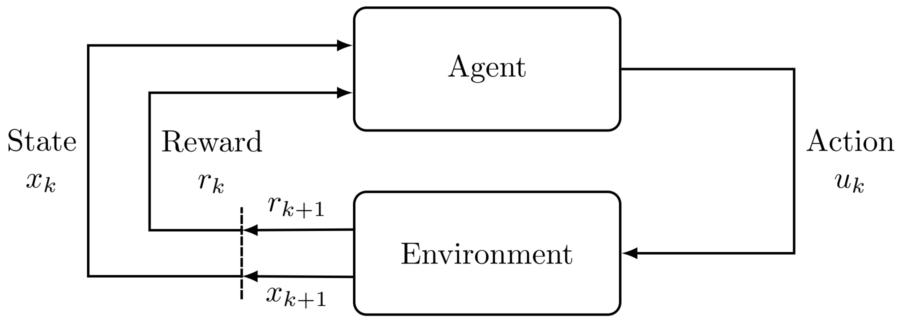

Figure 3.1: Block scheme of agent-environment interaction.

- **Environment:** It is the discrete dynamical system to control.
- **State:** It describes the environment at a certain time.
- **Action:** The agent interacts with the environment by performing an action and moves from one state to another.
- **Reward:** It is a numerical value that the agent receives based on its action.
- **Policy:** The agent makes a decision based on the policy. A policy tells the agent what action to perform in each state.

3.1.1 Sample-efficiency

Generally, a crucial point in applying RL on real systems is sample-efficiency. This because machine time is usually expensive and the reduction of the number of interactions required by the algorithm is a pro to speed up the optimization. Reinforcement learning approaches are usually divided into two main classes: model-free and model-based. Model-Free Reinforcement Learning (MFRL) includes the algorithms that do not require a model of the system to optimize. On contrary, if a model is required, the technique belongs to Model-Based Reinforcement Learning (MBRL). As presented in [38], Figure 3.2 shows that the presence of a model improves the sample efficiency. Furthermore, the way in which the policy is evaluated or improved, affects efficiency. Indeed, on-policy methods attempting to evaluate or improve the policy that is used to make decisions are less efficient than off-policy methods that evaluate or improve a policy different from that used to generate the data.

3.1.2 Investigated Algorithms

The tree diagram depicted in Figure 3.3 shows the algorithms of RL involved in the study carried out during the doctoral course. Later in this chapter,

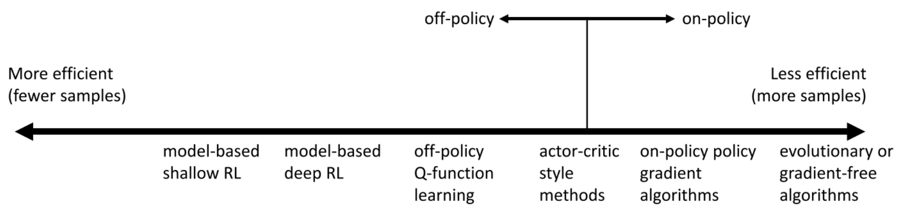
off-policy
on-policy
More efficient (fewer samples)
Less efficient (more samples)
model-based shallow RL
model-based deep RL
off-policy Q-function learning
actor-critic style methods
on-policy policy gradient algorithms
evolutionary or gradient-free algorithms

Figure 3.2: Sample efficiency in reinforcement learning.

each element of the tree will be discussed following the branches from top to bottom and from left to right.

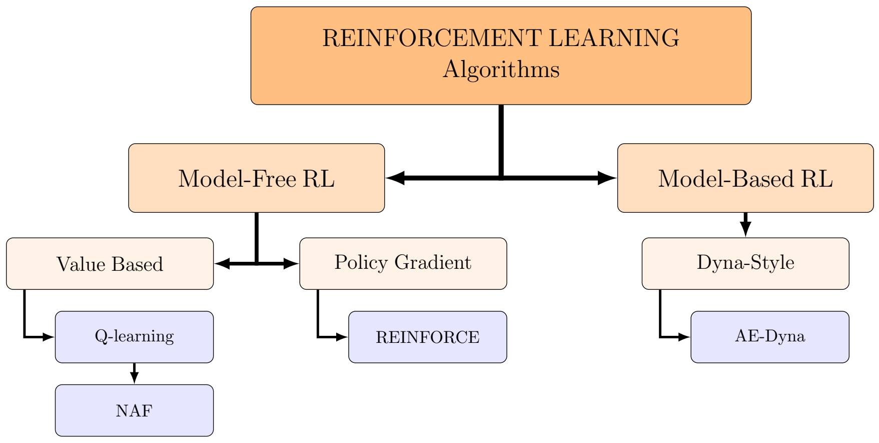

Figure 3.3: Taxonomy of RL algorithms employed.

In addition, two more methods have been investigated within the family of model-based optimization algorithms. They are the Gradient Method and the Iterative Linear Quadratic Regulator, which are applied to a model provided by a single neural network properly trained over real-data.

3.2 Model-Free Reinforcement Learning

Since model-free methods do not have any knowledge of the environment, they just try actions and observe the results, following a trial-and-error approach to reach the optimal policy. This means that the agent learns through direct interaction with the environment using the state/action values or the policy. These simple quantities can achieve the same optimal behavior without considering a model. Indeed, given a policy, a state has a value defined in terms of future utility that is expected to accrue starting from that state. The efficiency of such methods is in general worse than in model-based techniques. This is due to the combination of the new information recorded from the environment with the previous estimates that could be wrong or non sufficiently accurate.

The two main approaches to represent and train the agent with MFRL are given by Value Based and Policy Gradient methods.

3.2.1 Value Based Methods

The algorithms belonging to this category rely on finding the optimal policy by maximizing a value or action-value function. The best-known algorithm of this family is Q-learning [39], which is introduced in the next section in its flavour with linear function approximation. An extension of this method is then provided by the Normalized Advantage Function (NAF) technique presented in section 3.2.1.2.

3.2.1.1 Q-learning

Among the different approaches to the RL problem, the *approximate dynamic programming* aims at solving the problem

$$\text{maximize } J = \lim_{N\to\infty} \mathbb{E}\left[\sum_{k=0}^{N} \gamma^k r(x_k, u_k)\right],$$

in which $\gamma \in [0, 1[$ is the discount factor, by iteratively estimating an *action-value function* (or *Q-function*) from data. Here, J takes the form of an expected discounted cumulative reward. Assuming that there exists a stationary[1] optimal policy, the Q-function is defined as the optimum value of the expected discounted reward when action u is selected being in state x. Therefore, given the action-value function $Q(x, u)$, the optimal policy is

$$\pi^*(u|x) = \arg\max_u Q(x, u). \tag{3.2.1}$$

[1]stationarity is the consequence of the infinite time horizon ($N \to \infty$) and implies that the optimal action for a given state x at time k depends only on x and not on k.

In other words, estimating the Q-function amounts to solving the learning problem. An attractive and well-known method for estimating the Q-function is the *Q-learning* algorithm [39].

In the present work, the Q-learning has been employed in an episodic framework (meaning that the learning is split into episodes that end when some terminal conditions are met). The choice of the Q-learning among other RL approaches is due to its simplicity and the fact that the problem admits a non-sparse reward which is beneficial for speeding up the learning [40]. During learning, the exploration of the state-action space can be achieved by employing a so-called ϵ-greedy policy:

$$u = \begin{cases} \underset{u \in \mathcal{U}}{\arg\max}\ Q(x,u), & \text{with probability } 1-\epsilon \\ \text{random } u \in \mathcal{U}, & \text{with probability } \epsilon \end{cases}, \tag{3.2.2}$$

in which ϵ defines the probability of a random choice (exploration) while $\mathcal{U} = \{u^{(1)}, \dots, u^{(N)}\}$. The Q-learning update rule is:

$$Q(x,u) \leftarrow Q(x,u) + \alpha\delta, \tag{3.2.3}$$

where α is the learning rate and δ is the *temporal difference error*, the difference between the discounted optimal Q in the state x_{k+1} and the value $Q(x_k, u_k)$ (see Algorithm 1 for more details). Defining the state set as $\mathcal{X} \subset \mathbb{R}^n$ (where n is the dimension of the state vector), since there are N possible actions, the action-value function can be represented as a collection of maps $Q(x, u^{(1)}), \dots, Q(x, u^{(N)})$ from $\mathcal{X} \times \mathcal{U}$ to $\mathbb{R}$. In order to work with a continuous state space, a linear function approximation version of the Q-learning algorithm has been employed. More precisely, each $Q(x, u^{(j)})$ has been parametrized as $Q(x, u^{(j)}) = \theta_j^T \varphi(x)$, where $\varphi(x)$ is a vector of features and θ_j a weight vector associated to the j-th input $u^{(j)}$. Thus, the whole $Q(x,u)$ is specified by the vector of parameters $\theta = [\theta_1^T, \dots, \theta_N^T]^T$, and the corresponding policy will be identified by π_θ. In particular, Gaussian Radial Basis Functions (RBFs) have been employed as features; given a set of centers $\{c_i \in \mathcal{X},\ \ i = 1\dots,d\}$, $\varphi(x) = [\varphi_1(x), \dots, \varphi_d(x)]^T$ are defined. $\varphi_i(x) : \mathbb{R}^n \to \mathbb{R}$ is:

$$\varphi_i(x) = \exp\left(-\frac{\|x - c_i\|^2}{2\sigma_i^2}\right), \tag{3.2.4}$$

where σ_i determines the decay rate of RBF. The pseudo-code of the Q-learning with linear function approximation proposed in [41] is reported in Algorithm 1.

Algorithm 1 Q-learning algorithm with linear function approximation

```
Initialize $\theta$ and set $\alpha$, $\gamma$
For each episode:
    Set $k = 0$, initialize $x_0$
    Until $x_{k+1}$ is terminal:
        Choose $u^{(j)} \in \mathcal{U}$ using $\pi_\theta$
        Perform $u_k = u^{(j)}$
        Observe $x_{k+1}$ and $r(x_k, u_k)$

        $i \leftarrow \arg\max_l \theta_l^T \varphi(x_{k+1})$
        $\delta \leftarrow r(x_k, u_k) + \gamma\, \theta_i^T \varphi(x_{k+1}) - \theta_j^T \varphi(x_k)$

        $\theta \leftarrow \theta + \alpha\delta\varphi(x_k)$
        $x_k \leftarrow x_{k+1}$
        $k \leftarrow k + 1$
```

3.2.1.2 Normalized Advantage Function

A serious limitation for Q-learning consists in the non-trivial maximization of a Q-function with respect to continuous actions. Indeed, applications often require a continuous action space, as in particle accelerators. To overcome this limitation a Q-learning algorithms for continuous domains, named Normalized Advantage Function (NAF), is provided in [42]. It consists in a continuous variant of Q-learning algorithm combined with leaned models to accelerate the learning by preserving the benefits of model-free RL.

The discrete action space of Q-learning is overcame by assuming a specific algebraic form of the Q-function, such that $Q(x, u)$ is straightforward to optimize with respect to the action. The Normalized Advantage Function algorithm assumes a quadratic dependence of Q on the action u (θ are the networks parameters to be fitted):

$$Q\left(x, u|\theta^Q\right) = -\frac{1}{2}\left(u - \mu\left(x|\theta^\mu\right)\right)^T P\left(x, \theta^P\right)\left(u - \mu\left(x|\theta^\mu\right)\right) + V\left(x, u|\theta^V\right),$$

where $V\left(x, u|\theta^V\right)$ is the value function, i.e. the expected return when starting in state x and following policy π, while $P\left(x, \theta^P\right)$ is a state-dependent, positive definite square matrix; $u^* = \mu\left(x|\theta^\mu\right)$ corresponds to the action that maximizes the Q function.

Assuming this specific representation of the action-value function $Q(x, u)$ of course limits the representational power of the algorithm, but optimization problems are often convex and thus a quadratic action-value function can be a reasonable assumption.

The NAF agent is therefore a neural network with an input layer that receives the state x, while the outputs are $\mu\left(x, u|\theta^{\mu}\right)$, $P\left(x, u|\theta^{P}\right)$, and $V\left(x|\theta^{V}\right)$. In Algorithm 2, the pseudo-code of the NAF from [42] is provided .

Algorithm 2 Continuous Q-learning algorithm with NAF

```
Randomly initialize normalized $Q$ network $Q\left(x, u|\theta^{Q}\right)$
Initialize target network $Q'$ with weight $\theta^{Q'} \leftarrow \theta^{Q}$
Initialize replay buffer $R \leftarrow 0$
For each episode:
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial observation state $x_1 \sim p\left(x_1\right)$
    For each time $k$:
        Select action $u_k = \mu\left(x_k|\theta^{\mu}\right)$
        Execute $u_k$ and observe $r_k$ and $x_{k+1}$
        Store transition $\left(x_k, u_k, r_k, x_{k+1}\right)$ in $R$
        for each iteration $i$:
            Sample a random minibatch of $m$ transitions from $R$
            Set $y_i = r_i + \gamma V'\left(x_{i+1}|\theta^{Q'}\right)$
            Update $\theta^{Q}$ by minimizing the loss:
                $L = \frac{1}{N}\sum_i\left(y_i - Q\left(x_i, u_i|\theta^{Q'}\right)\right)$
            Update the target network: $\theta^{Q'} \leftarrow \tau\theta^{Q} + \left(1-\tau\right)\theta^{Q'}$
```

3.2.2 Policy Gradient Methods

Policy Gradient (PG) methods are an alternative to value-based methods. The key concept behind PG is to define a parametrized policy $\pi_\theta(u_k|x_k)$, and to produce the highest return directly optimizing policy parameters following the policy gradient ascent. The approaches belonging to this category present several advantages: (i) they guarantee the convergence to a global or local (in the worst case) optimum; (ii) they are able to manage problems with a continuous and high dimensional action space; and moreover (iii) they can learn stochastic policies, i.e., policies able to output a probability distribution over actions given a particular state of the environment. Such an approach avoids relying on the computation of the value or action-value function for solving the optimization process.

The method here investigated is the Natural Policy Gradient (NPG) version of the REINFORCE algorithm, introduced in the following section.

3.2.2.1 Natural Policy Gradient REINFORCE

Given a policy $\pi(u|x,\theta)$ parametrized by a vector θ, Policy Gradient (PG) methods [37] aim at finding an optimal θ^* which ensures to

$$\text{maximize}\, \mathbb{E}\left[R(\xi)\right], \tag{3.2.5}$$

in which $\xi = (x_0, u_0, x_1, u_1, \dots, x_{T-1}, u_{T-1}, x_T)$ is a state-input trajectory obtained by following a particular π, and $R(\xi) = \sum_{k=0}^{T-1} r(x_k, u_k)$; $(x_k, u_k) \in \xi$ is the corresponding cumulative reward. The trajectory ξ can be thought of as a random variable having a probability distribution $P(\xi|\theta)$. REINFORCE algorithm [43] has been employed since it aims at finding the optimal θ^* for $P(\xi|\theta)$, solution of the optimization problem (3.2.5), by updating θ along gradient of the objective function. More precisely, a stochastic gradient ascent is performed:

$$\theta \leftarrow \theta + \alpha R(\xi)\nabla_\theta \log P(\xi|\theta), \tag{3.2.6}$$

where $\alpha \in [0, 1]$ is the learning rate.
Since $P(\xi|\theta) = p(x_0) \prod_{k=0}^{T-1} \pi(u_k|x_k,\theta)p(x_{k+1}|x_k,u_k)$, where $p(x_{k+1}|x_k,u_k)$ is the transition probability from x_k to x_{k+1} when the action u_k is applied, the update rule (3.2.6) becomes:

$$\theta \leftarrow \theta + \alpha \sum_{i=0}^{T-1} r(x_i, u_i)\nabla_\theta \log \pi(u_i|x_i,\theta). \tag{3.2.7}$$

Such an update is performed every time a path ξ is collected. In order to reduce the variance of the gradient estimates, typical of the PG approaches [44], a NPG version [45] of the REINFORCE algorithm has been employed, in which a linear transformation of the gradient is adopted by using the inverse Fisher information matrix $F^{-1}(\theta)$. The pseudo code of the REINFORCE is reported in Algorithm 3.

Algorithm 3 Natural Policy Gradient REINFORCE

```
Set $\theta = 0$ and set $\alpha$
**While** $True$:
    Obtain $\xi = (x_0, u_0, x_1, u_1, \dots, x_{T-1}, u_{T-1}, x_T)$ applying $\pi(u|x,\theta)$
    Observe $r(x_k, u_k)$ for each $(x_k, u_k) \in \xi$

    $\theta \leftarrow \theta + \alpha F^{-1}(\theta) \sum_{k=0}^{T-1} r(x_k, u_k)\nabla_\theta \log \pi(u_k|x_k,\theta)$
```

3.3 Model-Based Reinforcement Learning

In contrast with model-free reinforcement learning, which aims to maximize the cumulative reward by defining the optimal policy, model-based reinforcement learning methods rely on planning the control through the dynamics model. A nice overview of this category is provided in [46]. Although model-based approaches are more sample-efficient than model-free, their effectiveness is usually reduced by model-bias. Since the MBRL algorithm builds an artificial model of the system and uses it to optimize policy, if the model does not fit the real environment (model-bias) then the policy fails in the real application. However, this problem has been recently mitigated by characterizing the uncertainty of the learned models through probabilistic models, and ways to compute the epistemic uncertainty such as variational inference, Markov Chain Monte Carlo [47], Drop out [48], and ensemble techniques [49]. Choosing for example a model-ensemble, model-free algorithms can be used within model-based approaches improving the sample-efficiency [50]. The present study will focus on a particular algorithm that belongs to the Dyna-style family which is introduced in the next section.

3.3.1 Dyna-Style Methods

The idea behind Dyna-style planning [51–53] is to perform model-free reinforcement learning algorithms without needing to interact with the real world. Such methods use the experience of the real system to build a model on which the policy is improved. Model and policy are the key points of this approach, which schematic representation proposed by [17, 54] is shown in Figure 3.4 where the left part (red) presents a generic Dyna-Style approach while the right part (green) is a solution to avoid model bias using a model-ensemble. Following the current policy, data are collected and used to update the model of the system dynamics on which then the agent optimizes the policy. The training iterates between (i) improving the model, (ii) collecting more data, and (iii) optimizing the policy by interacting with the updated model. Once the convergence is reached, the optimal policy should be able to solve the real problem. Reducing the number of iterations with the target system, this simple procedure provides a sample efficient approach. The Dyna-style algorithm is introduced in the next section.

3.3.1.1 Anchored Ensemble Dyna

The Anchored Ensemble Dyna (AE-Dyna) is a novel custom way to implement the Dyna-style approach developed at the University of Salzburg [17, 54]. It focuses on capturing the uncertainties of both the model and the data [55]. The former, epistemic, is answered by Bayesian statistics while the latter, aleatoric, by probabilistic models. Two are the main characters

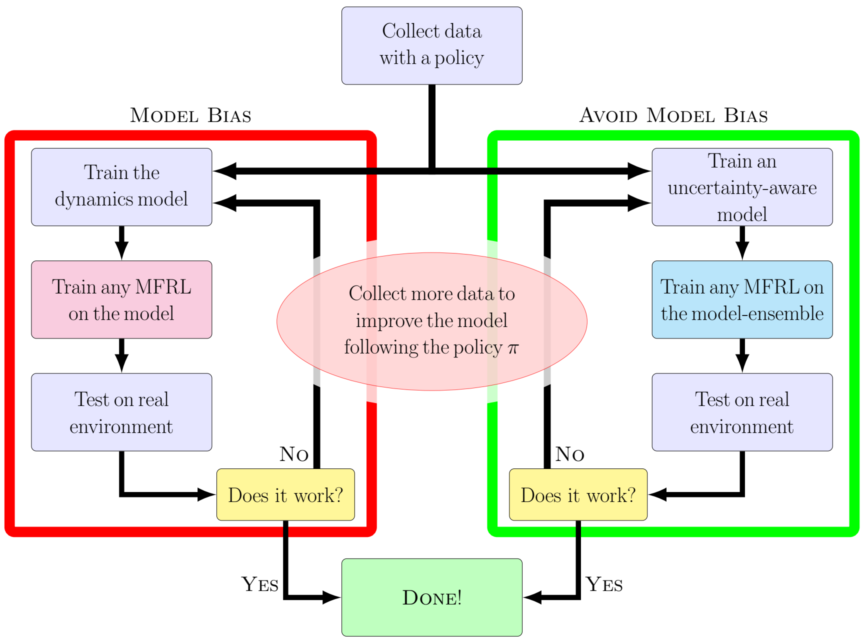
Collect data with a policy
Model Bias
Avoid Model Bias
Train the dynamics model
Train any MFRL on the model
Test on real environment
Does it work?
No
Yes
Collect more data to improve the model following the policy π
Train an uncertainty-aware model
Train any MFRL on the model-ensemble
Test on real environment
Does it work?
No
Yes
Done!

Figure 3.4: Dyna-style schematic approach.

of this approach: an ensemble of models whose purpose is to model the system dynamics combating at the same time the model-bias, and the policy optimizer, i.e. a model-free reinforcement learning algorithm to apply on the model as schematized in Figure 3.4.

By focusing on the model ensemble, it indicates a set of M dynamics models $\hat{f}_1, ..., \hat{f}_M$ that have been trained with the same data from the target environment. All of them are trained through the application of standard supervised learning, and the only differences between the models are due to the prior (defined by initial weights), and the batches order. More details can be found in [17, 54]. Once the models have been set up, a MFRL algorithm, such as Trust-Region Policy Optimizer (TRPO) [56], Proximal Policy Optimization (PPO) [57], and Soft Actor Critic (SAC) [58], can be easily used to interact with the models in order to improve the policy as much as possible. When it no longer improves, new data are collected from the target system to refine the models. In conclusion, the fundamental concepts of AE-Dyna are Policy Optimization and Policy Validation.

Policy Optimization The models define the trajectories used by the MFRL algorithm to estimate the gradient required for the optimization of the policy. To avoid overfitting on a wrong model, at each episode a model is randomly selected out of the ensemble to predict the following state from the current state and action. Following this procedure, more stable learning is achieved.

Policy Validation After a fixed number of training steps, the policy is tested on each model individually for a certain number of episodes. If there is no improvement, on at least 70% of the models, more data is collected to improve the model using the latest policy.

Algorithm 4 Anchored Ensemble Dyna

```
Select the MFRL agent
Initialize a policy $\pi_\theta$ and all models $\hat{f}_1, \hat{f}_2, ..., \hat{f}_M$.
Initialize an empty dataset $\mathcal{D}$.
repeat
    Collect samples from the real system $f$ using $\pi_\theta$ and add them to $D$.
    Train all models using $\mathcal{D}$.
    repeat                                        ▷ Optimize $\pi_\theta$ using all models.
        Collect fictitious samples from $\{\hat{f}_i\}_{i=1}^M$ using $\pi_\theta$.
        Update the policy using the MFRL agent on the fictitious samples.
        Estimate the performances $\hat{\eta}(\theta; i)$ for $i = 1, ..., M$.
    until the performances stop improving.
until the policy performs well in real environment $f$.
```

3.4 Standard Methods for Model-Based Optimization

Commonly, if the analytical model is available, standard techniques are preferred for their efficiency and easy adjustment. However, it is not always possible to have such a model of the system, especially if they are complex and affected by noise. The identification of these systems could be easily done using a proper neural network trained on real data. This procedure provides a virtual model on which the algorithms can make their calculation and find the best control action to apply to the real system.

The biggest limitation of this approach is that the identified system is static and it is not able to follow any evolution of the system that occurred after the data collection. Nevertheless, if the dynamical system to optimize can be supposed constant for a certain amount of time, in that range the optimization based on the identified model can produce good results.

The methods applied to the identified model, i.e. Gradient Ascent and Iterative Linear Quadratic Regulator, are introduced in the following.

3.4.1 Gradient Ascent

Gradient methods are well-known optimization techniques. They consist of first-order iterative algorithms whose main target is to find a local minimum or maximum. The main elements required by these techniques are a starting state $x \in \mathbb{R}^n$, a function to optimize $F(x)$, and a step size $\zeta \in \mathbb{R}^+$. The desired extremal point is achieved thanks to the direction provided by the gradient of the function to optimize $\nabla F(x)$. If the extremal point is researched by following the direction provided by the gradient, $\zeta \cdot \nabla F(x)$, a local maximum will be reached while moving in the opposite direction, $-\zeta \cdot \nabla F(x)$, a local minimum will be found. In the first case, the method is called Gradient Ascent (GA), in the second Gradient Descent (GD).

In order to get a model on which apply the gradient ascent algorithm, the function which returns the performance y of the system

$$y = F(x)$$

is modelized by a neural network $\hat{F}(x)$ that returns a prediction of the performance $\hat{y}$. In such a way, the gradient can be calculated on the virtual model and the control action applied to the target system as proposed in Algorithm 5.

Algorithm 5 Gradient Ascent on Neural Network

```
Initialize model $\hat{F(x)}$
Set $\zeta$, initialize $x_0$ and get $y_0 = F(x_0)$ Set $k = 0$
**Until** $y$ is terminal:
    Calculate $u_k = \zeta \cdot \frac{\nabla \hat{F}(x_k)}{\|\nabla \hat{F}(x_k)\|_2}$
    Set $u_k$ to the real system
    Get new state $x_{k+1}$ and new intensity $y = F(x_{k+1})$ from the real system
```

3.4.2 Iterative Linear Quadratic Regulator

Outside the RL family, the last method investigated is the Iterative Linear Quadratic Regulator (iLQR) [59] which calculates an optimal trajectory from the initial to the target state by optimizing a cost function. In particular, iLQR uses an iterative linearization of the non-linear system around a nominal trajectory and computes a locally optimal feedback control law via a modified Linear Quadratic Regulator (LQR) technique. Furthermore, by refining iteratively the trajectory, iLQR will eventually converge to the optimal solution.

Discrete-time finite-horizon iLQR optimizes an objective function J_0 with respect to affine state-transition dynamics from an initial state x_0. Defining the state variable $x_k \in \mathbb{R}^n$, and the control $u_k \in \mathbb{R}^m$, the dynamical system is

$$x_{k+1} = f(x_k, u_k)$$

while the cost function in the quadratic form is

$$J_0 = \frac{1}{2}\left(x_N - x^*\right)^T Q_f \left(x_N - x^*\right) + \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right)$$

where x_N describes the final state, and x^* is the given target. Matrices Q and Q_f are the state cost-weighting matrices positive semi-definite while the control matrix R is positive definite.
Once defined the temporal horizon, N, the algorithm requires two sequences: a nominal control sequence u_s that is a set of N control actions u_k, and a nominal trajectory x_s which is obtained by applying the actions $u_k \in u_s$ to the dynamical system initialized at x_0. At each iteration, the non-linear system is linearized around x_k, u_k, and the modified LQR problem solved. The final result of the iteration is an improved control sequence. A more accurate description of the iLQR is provided in [59].

If the system dynamics is unknown, it could be identified by sampling the system to train a neural network. This procedure creates a virtual model $\hat{f}\left(x_k, u_k\right)$, able to overcome the absence of an analytical model $f\left(x_k, u_k\right)$. The system dynamics is modeled as

$$\hat{x}_{k+1} = \hat{f}\left(x_k, u_k\right)$$

where $\hat{x}_{k+1}$ is the next state predicted by the neural network.

At iteration ending, the prediction of the control sequence $\hat{u}_s$ is returned and the first action $\hat{u}_k = \hat{u}_s\left[0\right]$ applied to the real environment. Therefore, iLQR can optimize real systems without knowing their analytical model but just considering a virtual model to define the control actions to apply. The pseudo-code of this approach is presented in Algorithm 6.

Algorithm 6 Iterative Linear Quadratic Regulator on Neural Network

```
Initialize model $\hat{f}\left(x_k, u_k\right)$
Set $Q$, $Q_f$, $R$ and $J_0$
Initialize $x_0$ and $u_s$
Set $k = 0$
**Until** $x_{k+1}$ is terminal:
    $\hat{u}_s \leftarrow$ modified LQR on $\hat{f}(x_k, u_k)$ and $J_0$
    Select $\hat{u}_k = \hat{x}_s[0]$
    Apply $\hat{u}_k$ to real system
    Get new state $x_{k+1}$
```

Chapter 4

Implementation, Results and Discussion

The optimization methods described in the previous chapters have been implemented and applied on FERMI to control the trajectory of the external optical source, i.e. the seed laser, in the modulator. In particular, two different problems have been addressed:

- The attainment of an optimal working point, starting from random initial conditions.
- The recovery of the optimal working point when some drifts, or working conditions changes, occur.

The former has been faced by all the algorithms previously introduced except the REINFORCE method that has been applied on the latter. In the present chapter, the experimental protocols are described and the results reported are discussed.

In the optical systems considered, the state x is a 4-dimensional vector that provides the current-voltage values applied to each piezo-motor[1] (two values for the first mirror and two values for the second mirror). The input u is also a 4 dimensional vector; denoting the component index as a superscript, the update rule is:

$$x_{k+1}^{(i)} = x_k^{(i)} + u_k^{(i)}, \quad i = 1, \dots, 4,$$

i.e., the input is the incremental variation of the state itself.
Moreover, the state x can only assume values that satisfy the physical constraints of the piezo-motors [30]:

$$x_{\text{MIN}} \leq x \leq x_{\text{MAX}}, \tag{4.0.1}$$

[1]the dynamics of the piezo-motors has been neglected, being their transients much shorter than the time between shots.

hence, for each state x of both systems, only those inputs u for which the component-wise inequality (4.0.1) is not violated are allowed.

It must be noticed that in the iLQR algorithm the state has been differently defined, while the control action form has remained the same. Further information is provided in Section 4.1.5 where its implementation and results are presented.

In the following, when referring to the intensity of the EOS, the product of the two intensities detected in the ROIs when the laser hits both ROIs is intended; by FEL intensity, the intensity measured by the I_0 monitor is meant. Finally, for both systems, the target intensity (computed as explained below) is denoted by I_T.

4.1 Optimal Working Point Attainment Problem

The problem of defining a policy, able to lead the plant to an optimal working point starting from random initial conditions, requires to split the experiments in two phases: (i) a training, which allows the controller to learn a proper policy, and (ii) a test (in the following also called verification), to validate the ability of the learned policy to properly behave, possibly in conditions not experienced during training.

At the beginning of the training, the target value I_T is selected. It remains the same for all the training episodes. At each time-step k, the input provided by the agent is applied, and the new intensity $I_D(x_{k+1})$ is compared with the target (I_T). The episode ends in two cases:

- when the detected intensity in the new state $I_D(x_{k+1})$ is greater than or equal to a certain percentage p_T of the target ($\frac{1}{100} \cdot p_T I_T$);
- when the maximum number of allowed time-steps is reached.

When the first statement occurs, the goal is achieved.
At the end of each episode, a new one begins from a new initial state, randomly selected, until the maximum number of episodes is reached. Then, a test (with random initial states) is carried out for the same target conditions of the training.

The same procedure is followed for the EOS system, with the only difference that at the beginning of the training the ROIs are selected, and therefore the target value I_T. Both values will be kept constant for all the episodes.

4.1.1 Q-Learning

The Q-learning with linear function approximation approach has been employed for both physical systems, precisely the EOS, and FERMI. The Q-learning with linear function approximation is a well-known and rather simple RL technique, and for that reason has been chosen for the first attempt to apply RL to the considered problem. In particular, the implementation of the algorithm foresees a discrete action space, thus the module of each i-th component of u is set equal to a fixed value.

The study concerning this algorithm and the results obtained has been already presented in [14, 15].

Q-learning on EOS

The training of the EOS alignment system consists of 300 episodes. The number of episodes has been chosen after preliminary experiments on a EOS device simulator. However, based on the results obtained on the real device, the number of episodes can be actually reduced (see Fig. 4.1).
During the training procedure the values of ϵ (exploration) of (3.2.2) and α (learning rate) of (3.2.3), decay according to the following rules [60, 61]:

$$\alpha \leftarrow \alpha \cdot \frac{N_0 + 1}{N_0 + \#\text{episode}}, \quad \epsilon \leftarrow \frac{1}{\#\text{episode}}; \tag{4.1.1}$$

where the N_0 value is set empirically. In addition, the reward is shaped according to [40]:

$$r(x_k, u_k) = \bar{r} + k \cdot \frac{\gamma_{rs}\, I_D(x_{k+1}) - I_D(x_k)}{I_T}, \tag{4.1.2}$$

where $\bar{r}$ is taken equal to 1 if the target is reached, 0 otherwise; the values of $\gamma_{rs} > 0$ and $k > 0$ are set empirically. The specific design of (4.1.2) allows to reward the agent in correspondence of state-action pairs that lead to a sufficiently increased detected intensity $\gamma_{rs} I_D(x_{k+1}) > I_D(x_k)$ ($r(x_k, u_k) > 0$) and to penalize it otherwise ($r(x_k, u_k) < 0$). After the training, the test phase starts with a fixed $\epsilon = 0.05$ - to minimize the possibility of overfitting during evaluation as in [62] - and $\alpha = 0$. Training and test have been repeated 10 times (i.e. 10 different runs have been performed) and the collected results are reported in terms of the average duration of each episode. The parameter values employed during experiments are reported in Table 4.1; they result from offline experiments on a simulator of the EOS system.

Table 4.1: Parameters used in EOS Q-learning.

Parameter	Training	Test
number of episodes	300	100
max number of steps	10000	10000
p_T	95%	92.5%
σ^2_{RBF}	0.0075	0.0075
initial ϵ	1	0.05
initial learning rate α	0.1	-
N_0 in learning rate decay	20	-
discount factor γ	0.99	-
γ_{rs}	0.99	-
k factor	1	-

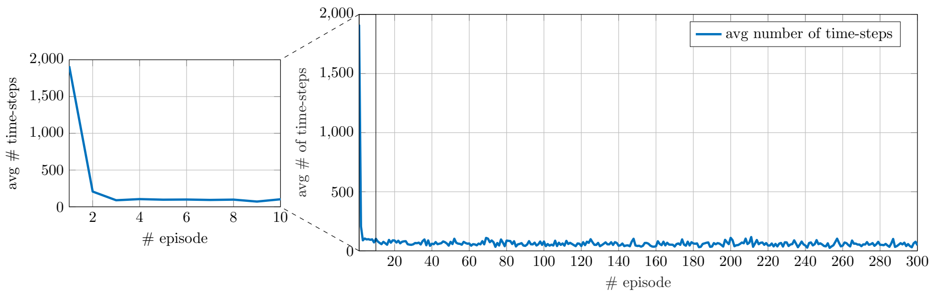

Figure 4.1: Average number of time-steps for each episode during the 10 runs in training performed on the EOS system. The average number of time-steps required in the first 10 episodes is highlighted in the enlarged portion.

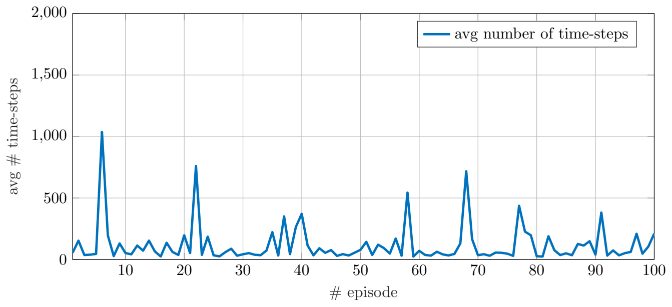

Figure 4.2: Average number of time-steps for each episode during the 10 runs in test performed on the EOS system.

The average number of time-steps per episode for the whole training phase is reported in Figure 4.1. The steep decrease of the average number of time-steps shows that a few episodes are sufficient to get a performance close to the one obtained after a whole training phase. Indeed, thanks to the reward shaping (4.1.2), the Q-function is updated at each step of each episode instead of just at the end of the episode (see Section 4.3 for further details). The average number of time-steps per episode during the test phase is visible in Figure 4.2 and is consistent with the training results.

Q-Learning on FEL

The experiment carried out on the FEL system consists of a training of 300 episodes and a test of 50 episodes. The chosen target value I_T is kept constant throughout the whole training and test. At the beginning of each episode, a random initialization is applied. Each episode ends when the same conditions defined in Section 4.1.1 occur. The ϵ and the α values decay according to (4.1.1), and the reward is shaped in the same way of the EOS case. The parameter values are reported in Table 4.2.

Table 4.2: Parameters used in FERMI FEL Q-learning.

Parameter	**Training**	**Test**
number of episodes	300	50
max number of steps	10000	10000
p_T	90.0%	90.0%
σ^2_{RBF}	0.0075	0.0075
initial ϵ	1	0.05
initial learning rate α	0.1	-
N_0 in learning rate decay	20	-
discount factor γ	0.99	-
γ_{rs}	0.99	-
k factor	1	-

The results are reported in Figure 4.3 and Figure 4.4, for training and test respectively. It can be observed that the overall behaviors, in training and test, resemble those in Figure 4.1 and Figure 4.2.

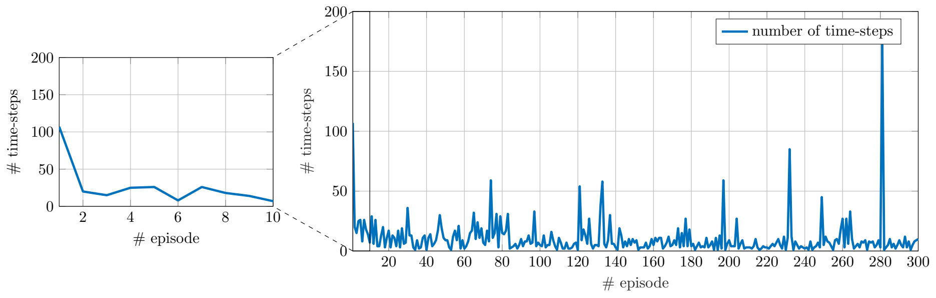

Figure 4.3: Number of time-steps for each episode during a single run of training performed on the FERMI FEL system. The number of time-steps required in the first 10 episodes is highlighted in the enlarged portion.

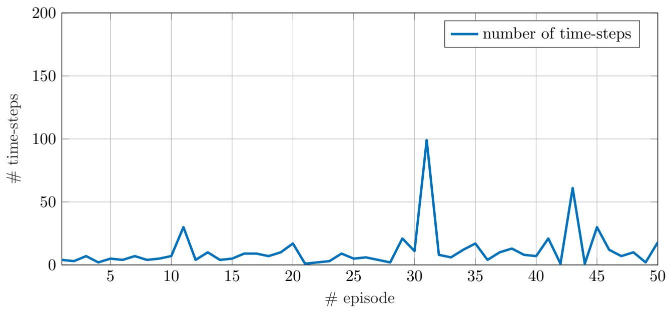

Figure 4.4: Number of time-steps for each episode during a single run of test performed on the FERMI FEL system.

4.1.2 NAF

At CERN the NAF algorithm has been empowered by the Prioritized Experience Replay (PER) technique for prioritizing the selection probability on data revealing a larger temporal difference error [63]. It leads to a better sample efficiency and has been successfully investigated (additional details available in section 5.2). In particular, the PER-NAF implementation is available in [64]. Following the good results presented in [12], the NAF algorithm has been applied also on FERMI FEL at Elettra Sincrotrone Trieste. In the collaboration with the University of Salzburg, a novel NAF implementation, namely NAF2 [17, 65], has been applied to FERMI. Specifically, NAF2 uses the double estimator method to overcome the overestimations of action values. A comparison between the results collected using both NAF and NAF2 methods to control the seed laser alignment at FERMI is now provided.

Maintaining the definition of state and action previously provided, the reward signal has been defined as

$$r\left(x_k, u_k\right) = -\frac{I_T - I_D\left(x_k + 1\right)}{I_T} \tag{4.1.3}$$

in order to have a negative reward the more the performance is far from its target. Anyhow, positive values can be achieved near the target; this is due to the inherent noise of the system. Furthermore, system performance is defined by normalizing the detected intensity with respect to the target intensity; this means that $I_T = 1$ always. I_T is kept constant for both training and test phases.

Similarly to Q-learning, both implementations consist of a training phase of 100 episodes and 50 episodes of verification. As for the parameters listed in Table 4.3, also the length of these two phases has been defined out of tests. The algorithm as used on FERMI can be found in [65].

Table 4.3: NAF and NAF2 parameters used on FERMI.

Parameter	**Training**	**Test**
p_T	95%	0.95%
number of episodes	150	50
max number of steps	10	10
model hidden layers	2	2
learning rate	0.001	0.001

Averaging the results collected in the two runs for each implementation, the following figures have been obtained. The average number of steps per episode is shown in Figure 4.5, whose paths are quite similar showing a steep decrease in the first 20 episodes, while in the following ones they are quite overlapped. In terms of sample efficiency, the NAF2 (solid lines) overcomes the NAF (dashed lines) as presented in the enlargement of the figure. The same information is highlighted in Figure 4.6 where the cumulative number of steps averaged on both runs for each technique is reported.

In verification the trends are very similar, requiring a comparable amount of time-steps to reach the goal. For both implementations, the average number of time-steps per verification is depicted in Figure 4.7, while the initial (blue) and final (green) intensities are presented in Figure 4.8 for NAF and Figure 4.9 for NAF2. In these two last figures, the intensities are compared to the dashed red line that represents the target intensity.

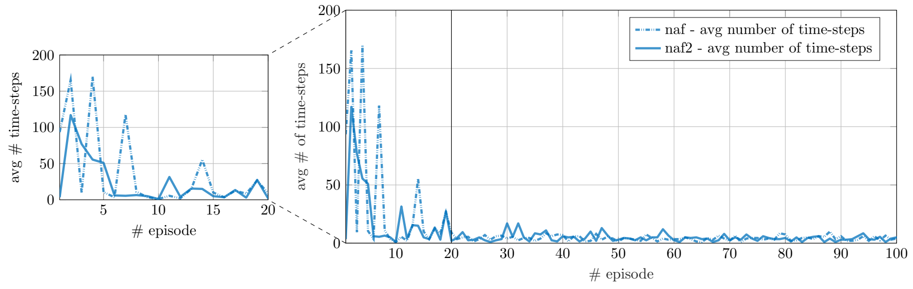

Figure 4.5: Average number of time-steps for each episode during the 2 runs of training performed on the FERMI FEL system. The number of time-steps required in the first 20 episodes is highlighted in the enlarged portion.

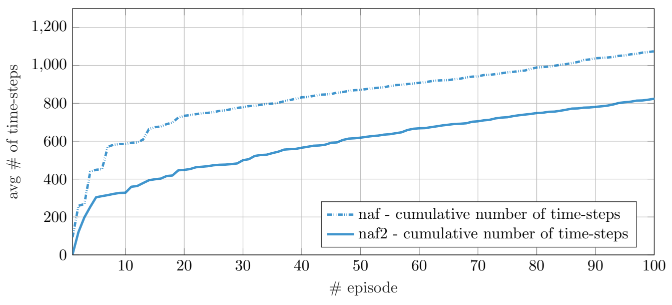

Figure 4.6: Cumulative number of time-steps averaged on both the 2 runs of training performed on FERMI FEL system. In terms of sample efficiency NAF2 overcomes NAF requiring less interactions with the real system.

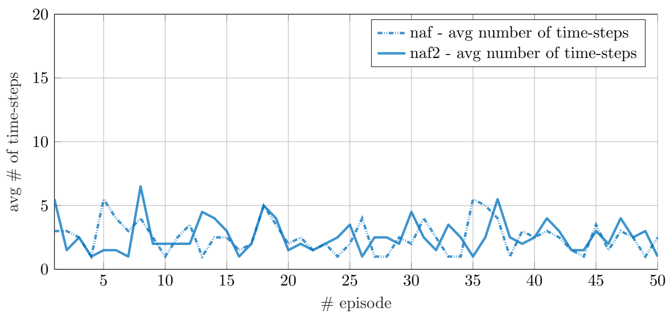

Figure 4.7: Average number of time-steps for each episode during the 2 runs in test performed on the FERMI FEL system.

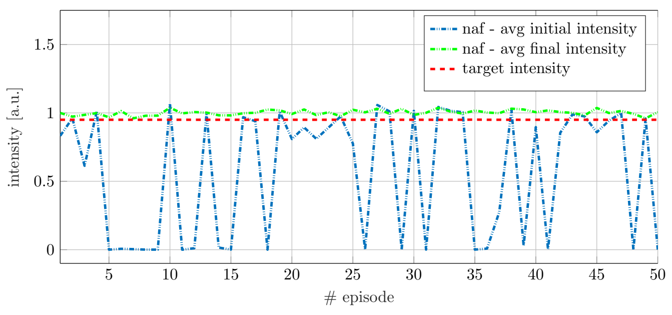

Figure 4.8: Average intensity during 50 episodes during 2 runs of NAF on FERMI FEL. For each episode, the blue line represents the initial intensity, while the green line represents the final intensity. The target intensity is the constant dashed red line.

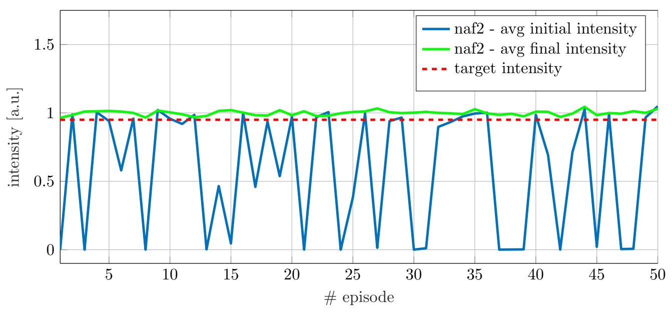

Figure 4.9: Average intensity during 50 episodes during 2 runs of NAF2 on FERMI FEL. For each episode, the blue line represents the initial intensity, while the green line represents the final intensity. The target intensity is the constant dashed red line.

4.1.3 AE-Dyna

Moving to model-based reinforcement learning techniques, the Dyna-style category has been investigated introducing a novel AE-Dyna approach, two variants of which have been implemented and deployed on FERMI in the collaboration with the University of Salzburg. The approach takes its origins from the Model-Ensemble Trust-Region Policy Optimizer (ME-TRPO) algorithm presented in [50] as a method belonging to Dyna-style category. Since the ME-TRPO is reported to be sensitive to aleatoric uncertainties [46], the proposed AE-Dyna improves it considering both epistemic and aleatoric uncertainties. In particular, two variants of this approach have been studied, implemented, and finally tested on FERMI. The two methods mainly differ for the model-free reinforcement learning algorithm used to optimize the policy interacting off-line with the model ensemble; indeed, the first technique searches the optimal policy applying the TRPO while the second one, the SAC. Since the general description of AE-Dyna has been previously provided in section 3.3.1, the following descriptions focus on the main aspects of the implementation as well as an overview of the results collected on FERMI.

4.1.3.1 AE-Dyna with TRPO

The AE-Dyna approach based on TRPO algorithm has been the first MBRL method to be applied to maximize FERMI performance. The TRPO is a model-free on-policy technique not suitable for solving online the proposed problem since it requires a long time and many interactions with the real system to find the optimal policy. Anyhow, its integration in the AE-Dyna structure creates a sample-efficient model-based method able to solve complex problems in just a few interactions with the real environment. In searching for the optimal policy the agent is refined every time new data are collected. The crucial settings of this approach are listed in Table 4.4. The code as used on FERMI tests can be found in [65].

Table 4.4: AE-Dyna with TRPO parameters used on FERMI.

Parameter	**Training**	**Test**
p_T	95%	0.95%
max number of steps	10	10
initial random steps	50	-
data-collection steps	25	-
total steps	450	-
number of models	3	3
model hidden layers	2	2
learning rate	0.001	0.001

The main data from training are depicted in Figure 4.10 with respect to the simulated epochs that are reported on the x-axis. The solid blue line presents the mean reward obtained by testing the agent on the individual models, while its standard deviation is defined by the light blue area. The number of data collected from the real environment, i.e. the green line, is shown in the same plot. It starts from the random initial collection of 50 samples and increases by 25 samples each time the Dyna approach needs to refine the models. The total number of data required in the training is 450.

The policy calculated during the training has been then applied for 50 episodes of verification in which the desired intensity has been always reached. The number of steps per episode required during the test phase is shown in Figure 4.11. Furthermore, the initial (blue) and final (green) intensities for each verification episode are presented in Figure 4.12. This plot highlights how starting also from low intensity the final performance overcomes the target intensity (dashed red).

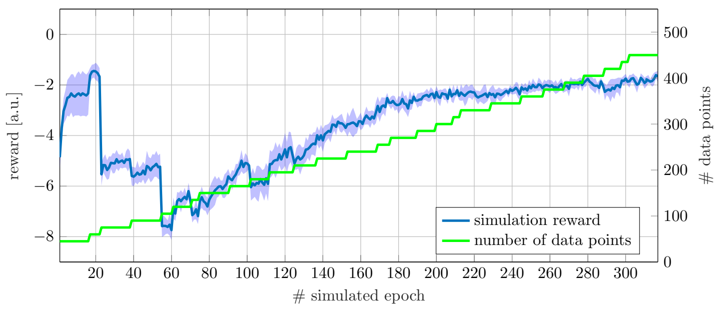

Figure 4.10: Number of data-points (green) and performance (blue) of the controller on the models which is used as a criteria to decide if more data are required to improve the models.

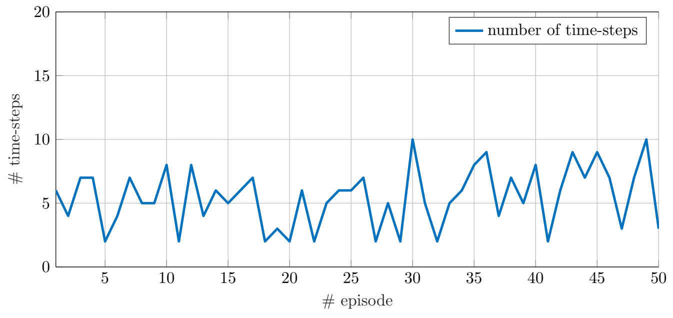

Figure 4.11: Number of time-steps for each episode during the verification performed on the FERMI system using AE-Dyna with TRPO.

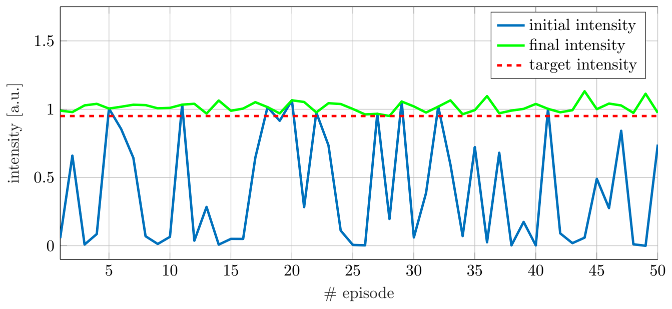

Figure 4.12: Intensity during 50 episodes of AE-Dyna with TRPO on FERMI FEL. For each episode, the blue line represents the initial intensity, while the green line represents the final intensity. The target intensity is the constant dashed red line.

4.1.3.2 AE-Dyna with SAC

The second variant of the AE-Dyna approach uses the SAC to find the optimal policy. The SAC is a model-free actor-critic method that improves the exploration managing a term associated to the entropy of the taken actions [58]. Differently from previous techniques, in this implementation the agent is re-trained from scratch every time new data are collected. During the experiments carried out on FERMI, the main parameters of the algorithm have been set as reported in Table 4.4. The code used on the FERMI tests is available in [65].

Table 4.5: AE-Dyna with SAC parameters used on FERMI.

Parameter	Training	Test
p_T	0.95%	0.95%
max number of steps	25	10
initial random steps	200	-
total steps	500	-
number of models	3	3
model hidden layers	2	2
learning rate	0.001	0.001

The training information is collected in Figure 4.13. The light blue band shows the standard deviation of the reward provided by the model-ensemble while the blue line presents its mean value collected by testing offline the policy. Moreover, it drops down when new data are acquired. The data points collected during training are shown by the green line that starting from 200 initial samples ends at 500 samples.

The time-steps required in the 50 episodes of verification are presented in

Figure 4.14. While the initial (blue), final (green), and target (dashed red) intensities are depicted in Figure 4.15. The AE-Dyna has been always able to reach a better performance than the desired intensity in about 5 steps.

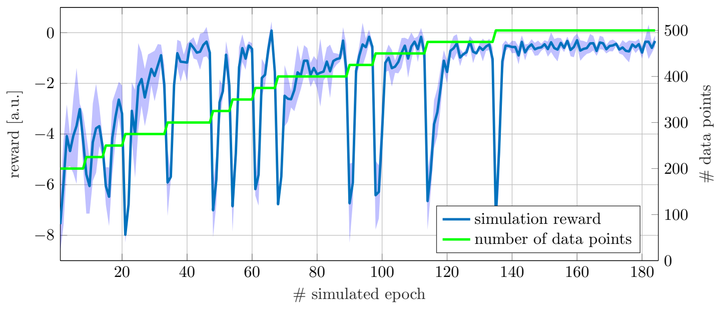

Figure 4.13: Number of data-points (green) and performance (blue) of the controller on the models which is used as a criteria to decide if more data are required to improve the models.

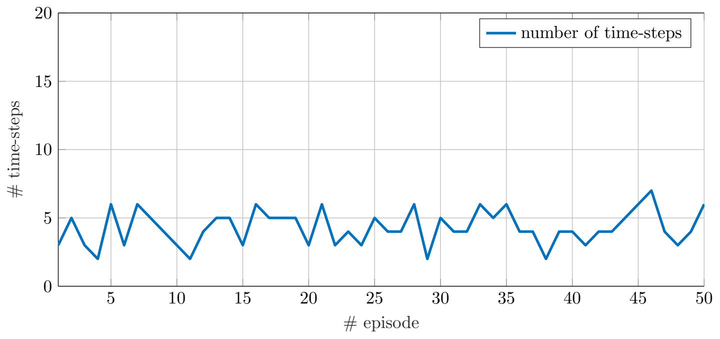

Figure 4.14: Number of time-steps for each episode during the verification performed on the FERMI system using AE-Dyna with SAC.

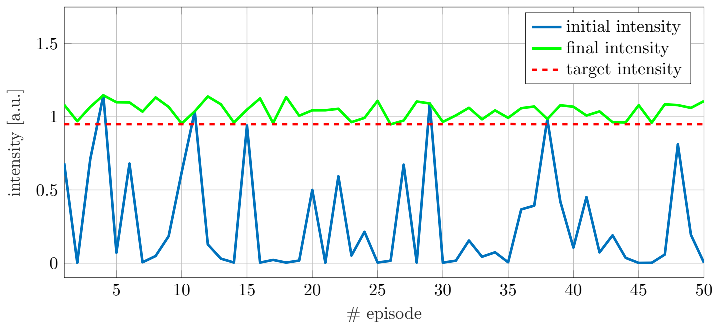

Figure 4.15: Intensity during 50 episodes of AE-Dyna with SAC on FERMI FEL. For each episode, the blue line represents the initial intensity, while the green line represents the final intensity. The target intensity is the constant dashed red line.

4.1.4 Gradient Ascent

Outside reinforcement learning, classical optimization is commonly based on gradient calculation. Indeed, the most well-known approach to finding the local minimum or maximum of a function is the gradient method. In particular, maximization is carried out by Gradient Ascent, the algorithm described in Section 3.4.1. Indeed, it searches the extremal point moving along $\zeta\nabla F\left(x\right)$ where $F\left(x\right)$ is the function to optimize. In the proposed problem, the objective function returns I_D, the intensity corresponding to a certain state x, but it is not analytically known. To overcome this inconvenience, the analytical form of $F\left(x\right)$ has been replaced with a neural network. This is a simple way to generate a model on which to apply the Gradient Ascent method reducing the number of interactions with the real system. Indeed, using for training M data-points collected directly from the real system, the model returns the predicted intensity $\hat{I}_D$ given the state x:

$$\hat{I}_D = \hat{F}\left(x\right) \approx F\left(x\right) = I_D.$$

In the application here proposed, a dataset $\mathcal{D}$ of $M = 1024$ samples has been collected from FERMI through several episodes following a random policy to select the actions. A dense (fully-connected) neural network of 3 hidden layers of 10, 16, and 10 neurons each has been used to identify the relation between intensity and state. Once its training is ended by early stopping to avoid overfitting, the real system is randomly initialized to state x_0. Calculating the gradient of the model, a control action is returned and applied to correct the laser trajectory. Such control action is defined by

$$u_k = \zeta \cdot \frac{\nabla\hat{F}(x_k)}{\parallel \nabla\hat{F}(x_k) \parallel_2},$$

where the step-size ζ is 0.1.
The entire procedure is repeated for 50 episodes which end when the maximum number of steps, 20, or an $I_D > 0.95 \cdot I_T$ is reached.

Figure 4.16 reports the initial and final intensities, together with the target, during the execution of the Gradient Ascent algorithm on FERMI. During the verification, 49 episodes out of 50 end by reaching the desired intensity. In Figure 4.17 the number of time-steps per episode for the whole run is reported.

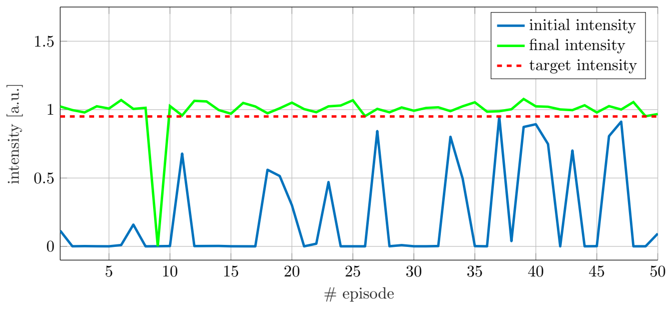

Figure 4.16: Intensity during 50 episodes of Gradient Ascent on FERMI FEL. For each episode, the blue line represents the initial intensity, while the green line represents the final intensity. The target intensity is the constant dashed red line.

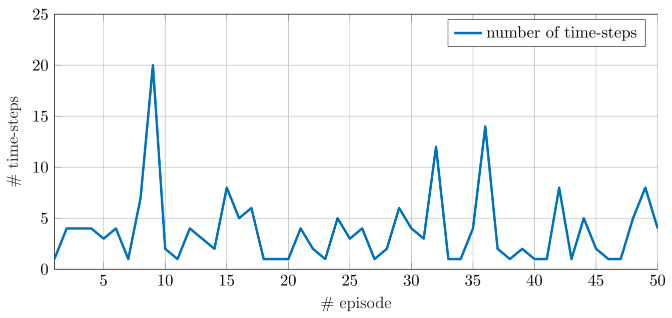

Figure 4.17: Number of time-steps for each episode during a single run of test performed on the FERMI FEL system.

4.1.5 iLQR

The second model-based method outside the RL family proposed in this study is the iterative linear quadratic regulator, or iLQR. As described in Section 3.4.2, the absence of an analytical model of the system has been solved by identifying its dynamics through a neural network. This neural network should predict the next state given the current state and action. However, an extended state has been considered for iLQR, indeed, the state x and its corresponding intensity I_D are appended into the new state $\tilde{x} = [x, I_D]$. By adding I_D in the state, the LQR cost function receives performance information of the system. In order to lead the FEL intensity as close as possible to the desired one, a control law is therefore developed. Accordingly to this modification, the system dynamics considered is given by

$$\tilde{x}_{k+1} = \hat{f}(\tilde{x}_k, u_k).$$

The model inputs are current state $\tilde{x}_k$ and action u_k, while the output is the new state $\tilde{x}_{k+1} = \left[x_{k+1}, \hat{F}\left(x_{k+1}\right)\right]$ where $x_{k+1} = x_k + u_k$, and $\hat{F}\left(x_k, u_k\right)$ is the same network of Gradient Ascent. To train it, a dataset $\mathcal{D}$ of $M = 1024$ data-points has been recorded from the real environment.

Focussing on the parameters of the algorithm, the most important ones are the matrices of the quadratic cost function. Their values empirically defined are $Q = Q_f = diag([1, 1, 1, 1, 1000])$ and $R = diag([1, 1, 1, 1])$ respectively. Another important element in the cost function is the target state x^*, a role that has been played initially by the state $\tilde{x}_k$ with higher I_D in dataset $\mathcal{D}$. During operation, x^* has been updated if during the episode a state with a better I_D has been encountered. In this application, the prediction time has been set equal to 3.

The initial and final intensities of each episode are shown in Figure 4.18, respectively in solid blue and green. The target value ($p_T = 95\%$) is depicted by the dashed red line. In all 50 episodes, the algorithm increased the intensity above the target. The number of steps required to reach the goal is provided in Figure 4.19.

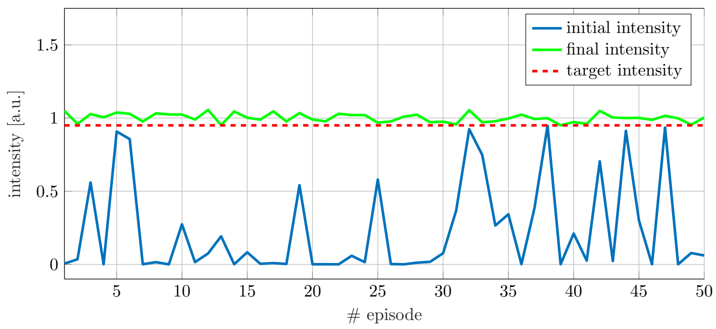

Figure 4.18: Intensity during 50 episodes of iLQR on FERMI FEL. For each episode, the blue line represents the initial intensity, while the green line represents the final intensity. The target intensity is the constant dashed red line.

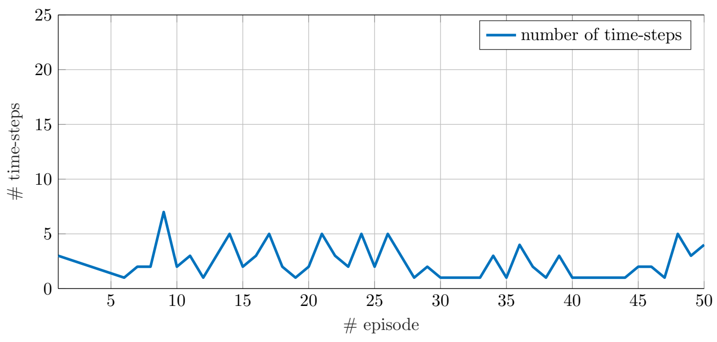

Figure 4.19: Number of time-steps for each episode during a single run of test performed on the FERMI FEL system.

4.2 Recovery of Optimal Working Point

In particle accelerators facilities, the working conditions are constantly subject to fluctuations. Indeed, thermal drifts or wavelength variations requested by users are common and result in a displacement of the optimal working point. Therefore, a controller must be able to quickly and properly adapt its policy to such drifts. For the purpose, the NPG REINFORCE algorithm (Section 3.2.2.1) has been adopted for its ability to work with a continuous action space and, thus, to allow for precise fine tuning. Here, the learning is applied as an adaptive mechanism, to face the machine drifts. Thus, in this case, a test phase would be meaningless, since adaptation occurs during learning only.

Similarly to Section 4.1, the state is a four-dimensional vector of the voltage values applied to each piezo-motor (two values for the first mirror and two values for the second mirror), and the action is composed of four references, one for each piezo-motor actuators, from which the new state depends.

4.2.1 NPG REINFORCE

The agent consists of four independent parametrized policies, one for each element of the action vector ($u_k^{(i)},\ i \in \{1,2,3,4\}$). In particular, assuming that the action choice u_k is uncorrelated with the current state x_k:

$$\pi(u_k|x_k;\theta) = \pi_i(u_k|\theta).$$

Given the bounded nature of the state and action spaces, each policy has been shaped according to the Von Mises distribution:

$$\pi_i(u_k^{(i)}|\theta_i) = \frac{e^{\psi_i cos(u_k^{(i)} - \mu_i)}}{2\pi\mathcal{I}_0(\psi_i)} \quad \text{s.t., } i \in \{1,2,3,4\},$$

where $\psi_i = e^{\phi_i}$ is a concentration measure, μ_i is the mean, $\mathcal{I}_0(\psi_i)$ is the modified Bessel function of the first kind [66] and $\theta_i = [\mu_i, \phi_i]$ is the i-th policy parameter vector, updated at each step of the procedure.

At each training step k, when the system is in state x_k, the agent performs an action u_k, according to the current policy, thus leading the system in a new state x_{k+1}. Then, the intensity $I_D(x_{k+1})$ is detected and the reward is computed according to:

$$r(x_k, u_k) = \frac{I_D(x_{k+1})}{I_T} - 1, \tag{4.2.1}$$

where I_T is the target intensity. In the EOS system, in order to emulate drifts of the target condition, I_T is initialized by averaging values collected at the beginning of the training procedure and then updated, each time that $I_D(x_{k+1})$ results greater than I_T, according to:

$$I_T \leftarrow I_T + 0.1(I_D(x_{k+1}) - I_T). \tag{4.2.2}$$

In the FEL, however, the system has been initialized in a manually found optimal setting (including both the state and the I_T), and some disturbances have been manually imposed. The possibility to update the target intensity is still enabled though.

NPG Reinforce on EOS

The NPG REINFORCE experiment performed on the EOS system consists of a single training phase, at the beginning of which the EOS system is randomly initialized, as well as the I_T. The learning rate α (3.2.7) is kept constant and equal to 0.1 (empirical setting). Only when the detected intensity $I_D(x_{k+1})$ assumes a value greater than I_T, the latter is updated according to (4.2.2), and the algorithm continues with the new target to be reached. The procedure is stopped when θ vectors lead each Von Mises distributions close enough to Dirac delta functions, after no target update has been performed for a predefined time. Figure 4.20 shows the detected intensity $I_D(x_{k+1})$ (blue line), its moving average (green line) and the target intensity I_T (red dashed line) during the experiment. In Figure 4.21, the reward (blue line) is reported along with its moving average (green line) and the target I_T (red dashed line). By comparing the two figures, it can be seen that once the target does not change, the reward approaches zero and the detected intensity variance shrinks, evidence that the optimal working point is close.

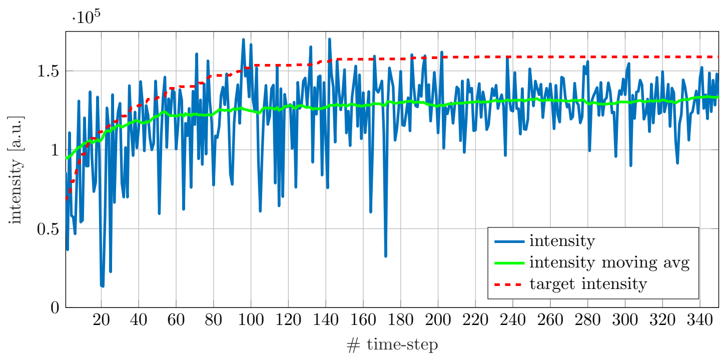

Figure 4.20: Intensity during a single run of NPG REINFORCE on EOS. The blue line represents the detected intensity, while the green line is its moving average obtained with a fixed window size of 50 samples. The dashed red line represents the target intensity. Until time-step 200 the improvement of the intensity is appreciable; a further evidence is the update of the target intensity. In the remaining time-steps, the target intensity exhibits only small updates.

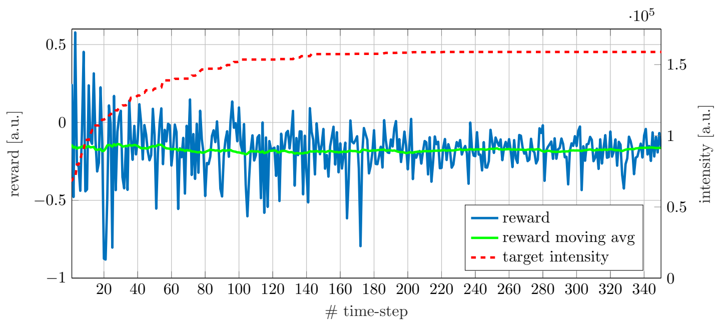

Figure 4.21: Reward (blue), moving average of reward (green) with a fixed window size of 50 samples and target intensity (red, dashed) during a single run of NPG REINFORCE on EOS. The target intensity increases each time a reward greater than zero is observed.

NPG Reinforce on FEL

Even in this case the experiment consists of a single training phase, at the beginning of which, however, the system is set on an optimal working point, manually found by experts. During the experiment, some misalignment are forced by manually changing the coarse motors position. The learning rate of (3.2.7) is kept constant and equal to an empirically set value ($\alpha = 0.5$).

Figures 4.22 and 4.23 report the detected intensity and the reward, together with the target, during the execution of the NPG REINFORCE algorithm on the FEL. It can be seen that, contrary to the EOS experiment, the target intensity is not significantly updated. Indeed, in this case the system is initialized on an optimal working point. Two drift events took place, the first around time-step 120, and the second around time-step 210. Both plots, (detected intensity and reward), clearly show the capability to recover an optimal working point.

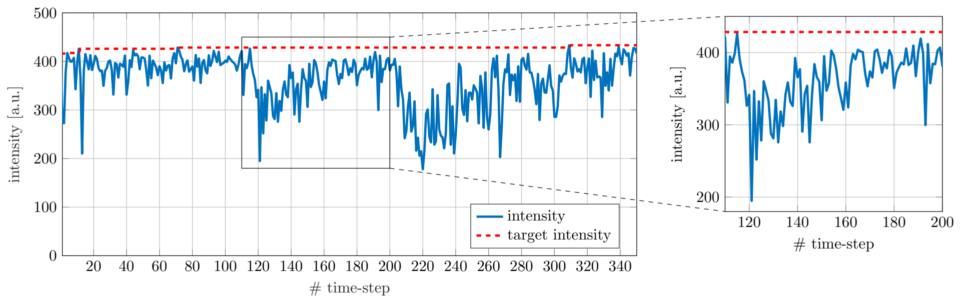

Figure 4.22: Intensity during a single run of NPG REINFORCE on FEL. The blue line represents the detected intensity while the dashed red line the target intensity. The target intensity is almost constant during the whole run. Two perturbations have been manually introduced by moving the coarse motors. It is possible to appreciate the capability to recover FEL intensity in both events. The first perturbation and subsequent recovery are highlighted in the enlarged portion.

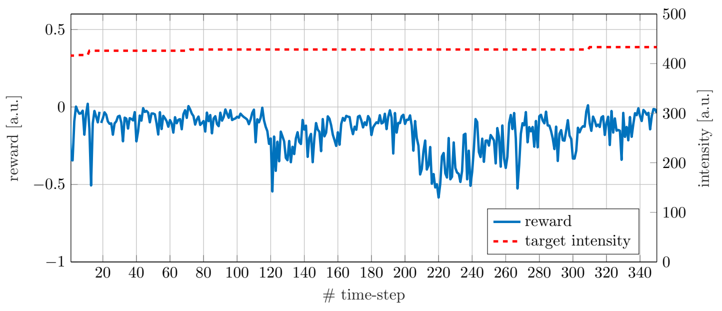

Figure 4.23: Reward (blue) and target intensity (red, dashed) during a single run of NPG REINFORCE on FEL. The slight increases of target intensity correspond to positive rewards.

4.3 Discussion

The results obtained during experiments by applying the proposed algorithms to control the trajectory of a laser beam have been reported. Indeed, both the problems presented in this chapter have been successfully solved.

The optimal working point attainment problem (section 4.1) has been addressed initially by Q-learning (3.2.1.1), which has been selected since it is a well known and rather simple RL technique, the promising results collected motivated the application of NAF (3.2.1.2) that is the extension to continuous action space of Q-learning. Moreover, to improve the sample-efficiency the investigation involved also two MBRL methods, both of which belong to the AE-Dyna category (3.3.1.1), one based on TRPO the other on SAC. Finally, two optimization algorithms based on the identification of the system through a neural network have been deployed, the gradient ascent and the iLQR (3.4.2).
As for the recovery of the optimal working point (section 4.2) problem, it has been addressed by just the REINFORCE (3.2.2.1), a policy gradient method from RL.

Despite the differences between the various algorithms, the same procedure has been followed to deploy and debug them before being applied on the FERMI FEL. The outcome of these preliminary operations is a runnable algorithm whose main parameters have been already grossly tuned. To correctly run the algorithm on the real system, several shifts to optimally tune the parameters have been always necessary. Furthermore, most of the algorithms here proposed have been investigated, developed, implemented, tested on a custom simulator, and applied on the FERMI FEL by the author. However, NAF and AE-Dyna techniques have been developed at the University of Salzburg, and thanks to the collaboration with this institute it was possible to successfully test them on the studied system.

All the satisfactory outcomes deserve some further comments that are here provided.

4.3.1 Outcomes of the Attainment Problem

All the experiments yielded satisfactory results showing that a non-linear and noisy problem could be solved in a feasible number of time-steps. A summary of the data points collected to train the RL agent or to identify the system is reported in table 4.6. The amount of data required by the RL methods agrees with the considerations introduced in section 3.1.1. In MFRL, the number of data decreases from about 3100 to 800, while the MBRL techniques require only about 450-500 samples. In the system identification required to run gradient ascent and iLQR a fixed number of data has been used, it is 1024 and further investigations will be carried out to improve the sample efficiency of these methods. Except for one test episode

of gradient ascent, all the test episodes have been successful. In terms of episode length, NAF, NAF2, and iLQR perform similarly, and better than the others. This information, as well as the intensity reached, is provided in Table 4.7. However, the experiments have been conducted on different days and under slightly different conditions.
Usage of the proposed methods in an operational way is attractive and could replace in the future the current optimization method, which needs the destructive screen measurement. Hence online retraining could be done and valuable time could be saved. Furthermore, additional studies regarding a long-time performance would be interesting.

Table 4.6: Summary of the number of interactions with the FERMI FEL for the training/identification phase.

Algorithm	**Data points**
Q-learning	3128
NAF	1074
NAF2	824
AE-Dyna with TRPO	450
AE-Dyna with SAC	500
Gradient Ascent	1024
iLQR	1024

Table 4.7: Performance of the different algorithms in the test phase. At the time this thesis was written, no data about Q-learning intensity was available.

Algorithm	**Episode length (mean)**	**Final intensity (mean)**
Q-learning	11.28	-
NAF	2.56	1.0019
NAF2	2.64	0.9995
AE-Dyna with TRPO	4.46	1.0150
AE-Dyna with SAC	3.28	1.0427
Gradient Ascent	3.82	0.9911
iLQR	2.54	1.0019

Q-Leaning has been applied to face the problem of finding an optimal working point, starting from a random initialization. The results are reported in Subsection 4.1.1. The enlarged portions reported in Figure 4.1 and Figure 4.3 show that a few episodes are sufficient to drastically reduce the number of steps required to reach the goal. In other words, the exploration carried out during the first episodes provides a valuable information for the estimation of the Q-function and, as a consequence, of an appropriate policy. Probably the main reason is the effectiveness of the reward shaping (4.1.2), that allows for obtaining a reward at each time-step, as opposite of a sparse reward occurring only at the end of the episodes. Such a shaping seems reasonable for the problem at hand, and is based on the assumption that the observed intensity change of two subsequent steps is significant for guiding the learning. On the other hand, during the test phase, some unsuccessful trials occurred. Although some further investigation is needed, it might be due to either (i) the occurrence of unexpected drifts of the target during the test or (ii) the discrete set of actions employed, consisting of fixed steps that can prevent reaching the goal, starting from random initial conditions.

NAF algorithm allowed continuous actions to search more accurately the optimal setup for the system. Two different implementations, NAF and NAF2, have been proposed and compared. Both in training and verification, the number of steps required per episode are quite similar, see Figure 4.5 and 4.7. However, the algorithm with a double network (solid lines) requires a fewer number of total steps with respect to the usage of a single network (dashed lines). Similarly to Q-learning, few episodes are required to drastically reduce the number of steps necessary to reach the goal. In addition, the graphics depicted in Figure 4.8 and 4.9 confirm the effectiveness of the method in reaching the desired target.

AE-Dyna approach has been introduced to improve FERMI performance introducing two variants that produced successful results. Both of them required a reasonable amount of data in the training phase reducing to 450 for the TRPO variant, and 500 for the SAC one, the number of interactions with the real system, and proving that AE-Dyna is a sample-efficient approach, see Figure 4.10 and 4.13. Furthermore, the reaching of the target intensity at each verification episode, as shown in Figure 4.12 and 4.15, confirms its applicability in automatically controlling complex systems with unknown dynamics slightly better than the NAF.

Gradient Ascent has been able to collect good results despite its simplicity and the absence of an analytical model. Indeed, identifying the system dynamics through a neural network trained on 1024 data-points it has been able to reach the target in 49 out of 50 episodes as depicted in Figure 4.16.

iLQR boosted Gradient Ascent. Indeed, with this technique the target has been always reached as presented in Figure 4.18. Furthermore, the recursive calculation of the next action improved its efficiency by reducing the number of steps per episode necessary to improve FERMI performance. The number of steps per episode in the test phase is shown in Figure 4.19.

4.3.2 Outcomes of the Recovery Problem

Unlike the previous section, the recovery of the optimal working point problem has been addressed by only one method. Therefore, it does not allow any significant comparison.

NPG REINFORCE algorithm has been applied for restoring the optimal working point in case of drifts. The results are reported in Subsection 3.2.2.1. In particular, Figures 4.22 and 4.23 show the response to manual perturbations of the FEL operating conditions, set initially in an optimal working point. It is possible to observe how the algorithm quickly replies to disturbances of environment settings (marked by negative reward spikes), by learning a policy able to recover the optimal pointing of the laser.

Chapter 5

Reinforcement Learning at CERN

Preliminary studies on RL have been carried out at CERN to increase and stabilize the performance of particle accelerators. Numerical optimization algorithms are approaches commonly used to reach similar targets. These algorithms in fact have many advantages such as their availability out of the box and their adaptability to a wide range of problems in accelerator operation. However, not all problems in accelerators can be addressed with similar approaches. In many cases, it is not possible to model the physics as e.g. the beam dynamics in the low energy regime for the electrons cooling in Low Energy Ion Ring (LEIR) [67].

Automated and sample-efficient controlling is an elegant solution to optimize complex systems in particle accelerators. The trial-and-error approach followed by reinforcement learning reduces the steps to a minimum value - one step in the best case - once the training is done. The sample efficiency of RL algorithms is described section 3.1.1. Conversely, numerical optimization needs an exploration phase at each deployment. An additional advantage of RL is given by learning the underlying dynamics of the problems requiring just an additional input: state information. Indeed, given a state, the RL agent applies the action to achieve the maximum reward.

A first preliminary investigation [11] has been carried out on the LEIR using the Deep Q-Learning (DQN), a reinforcement learning algorithm that combines Q-Learning with deep neural networks to let RL work for complex, high-dimensional environments. During the collaboration here described a model-free RL approach, the NAF, has been applied on AWAKE and Linac4, now discussed in section 5.2. While subsequent studies concerned a model-based technique, the TD3 in MBRL-Dyna mode.

Furthermore, outside RL family the iLQR algorithm has been implemented and applied on AWAKE. A summary of the application is provided in section 5.3.

5.1 CERN Accelerators Involved in RL Studies

In 2019, as well as for most of 2020, CERN accelerators have been in shutdown, except AWAKE and Linac 4. Therefore, the RL investigation has been carried out on these two systems to control the beam trajectory. Both of them are ideal test cases having a well-defined state x, high dimensional action/state spaces, and results can be compared with existing algorithms. The project goal is to correct the trajectory as well as existing methods, achieving a similar RMS in the fewest interactions with the system, ideally within one step.

To interact with such systems appropriate OpenAI Gym environments have been created. In addition, since AWAKE transfer line is modeled in MAD-X, the response matrix between beam position monitors and correctors has been used to create a simulated OpenAI Gym environment. The main purpose of this simulator is to test RL algorithms off-line and define the hyper-parameters for optimum sample efficiency.

AWAKE is a proton-driven plasma wakefield test facility which trajectory is controlled with 10 horizontal and 10 vertical steering dipoles according to the measurements of 10 beam position monitors (BPMs). During the test, only the horizontal plane has been considered reducing the system to 10 degrees of freedom. Since the state is defined by BPMs and the action by correctors the action and state spaces have the same dimension. Applied actions are limited by a *max action* value equal to 300 µrad, while the reward signal is calculated as the negative RMS value of the difference trajectory between measured and reference one. A certain value of the reward is fixed as the goal of the optimization, this value is -0.2 cm, while the minimum allowed reward is -1.2 cm. Finally, the maximum episode length is 50 steps.

Linac 4 is a H^- linear accelerator which trajectory is controlled similarly to AWAKE. The number of correctors per plane is 16 such as the overall number of BPMs. Also on this system, only the horizontal plane has been considered defining state and action spaces to dimension 16, while the corrections, i.e. actions, are limited to 0.5 A. The reward signal is defined as on AWAKE but the minimum allowed reward is -3 mm and the reward value to achieve the goal is -1 mm. The maximum number of steps per episode is 15.

5.2 NAF on AWAKE and Linac 4

The PER-NAF [64], a custom variant of NAF algorithm implemented at CERN, has been applied on the two systems just described, AWAKE and Linac 4, to optimize the beam trajectory [12]. In both applications, the agent starting from scratch learns the system dynamics and using this knowledge it manages the correctors to optimize the beam trajectory. The algorithm consists of a single neural network with two fully-connected dense hidden layers. In the following figures, the results collected on both particle accelerators are shown.

On AWAKE the PER-NAF has been able to understand the system in few episodes, indeed, Figure 5.1 depicts the number of steps per episode and it is possible to notice how the algorithm converges after about 25 episodes. Furthermore, the initial, final, and target RMS are shown in Figure 5.2. After the first 25 episodes, the target RMS is always reached.

Similar satisfactory results have been collected also on Linac 4. They are here presented in Figure 5.3 and 5.4. In the former, the time-steps recorded during the experiments are reported, while the latter proposes the initial, final, and target RMS that have similar behaviour to the ones recorded on AWAKE. Also in this system, the convergence is reached in about 25 steps.

After some training the agent corrects any initial steering to below the target RMS within few steps in both applications. The implementations used on the AWAKE and Linca 4 tests are available in [64].

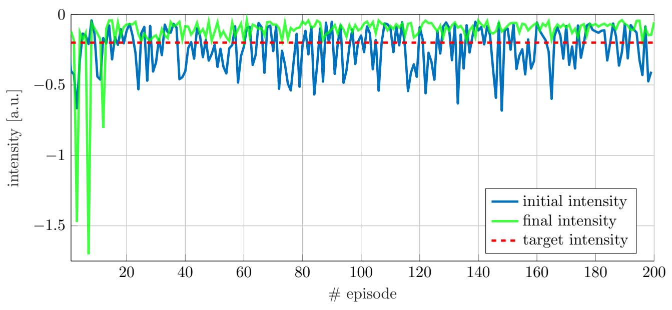

Figure 5.1: Number of time-steps for each episode during a single run of test performed on AWAKE.

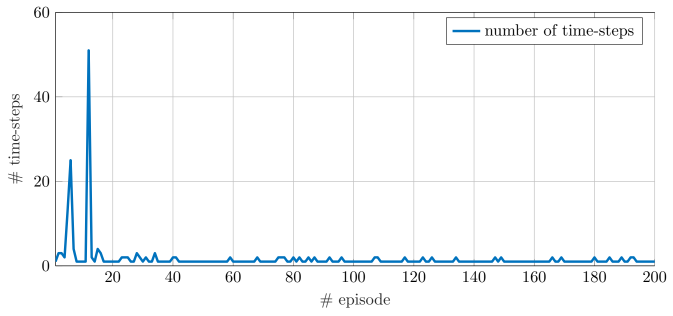

Figure 5.2: Intensity during 200 episodes of NAF on AWAKE. For each episode, the blue line represents the initial intensity, while the green line represents the final intensity. The target intensity is the constant dashed red line.

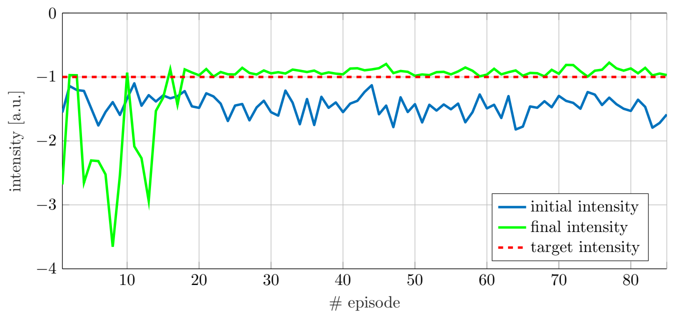

Figure 5.3: Number of time-steps for each episode during a single run of test performed on Linac 4.

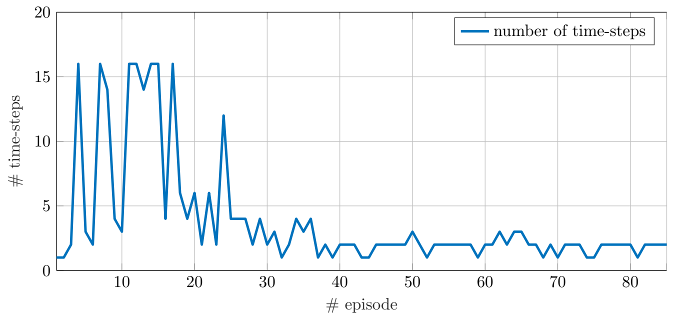

Figure 5.4: Intensity during 85 episodes of NAF on Linac 4. For each episode, the blue line represents the initial intensity, while the green line represents the final intensity. The target intensity is the constant dashed red line.

5.3 iLQR on AWAKE

During the collaboration with CERN also the iLQR algorithm has been tested on AWAKE. Similarly to experiments on FERMI, the system dynamics of AWAKE has been modelized with a neural network using real data. The initial dataset $\mathcal{D}$ consisted of 1000 data-points collected by applying randomly selected actions u_k to update the state from x_k to x_{k+1}. During verification, the dataset has been successfully reduced to 200 samples in order to improve the sample-efficiency of the approach. With this small amount of data the model $\hat{f}(x_k, u_k)$ has been trained and the algorithm applied.

To apply iLQR on the AWAKE problem, the matrices of the cost function have been defined as positive and diagonal. In particular, the non-zero values of Q and Q_f have been put to 1000, while the diagonal values of R have been all set to 1. The prediction time has been imposed on 3.

After the identification of the system, a test of 50 episodes has been done. In all episodes, the target has been reached in a few steps as shown in Figure 5.5. Initial (blue) and final (green) RMS are reported in Figure 5.6 where the dashed red line indicates the target RMS of -0.2 mm.

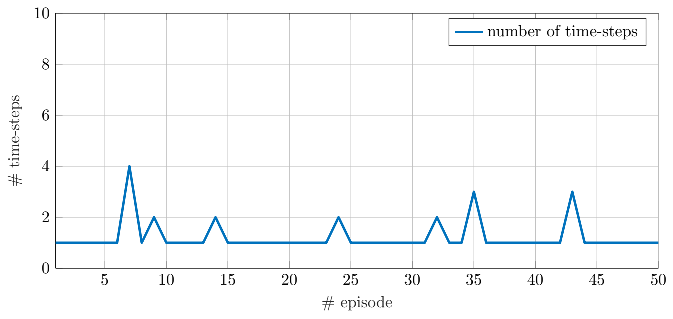

Figure 5.5: Number of time-steps for each episode during a single run of test performed on AWAKE.

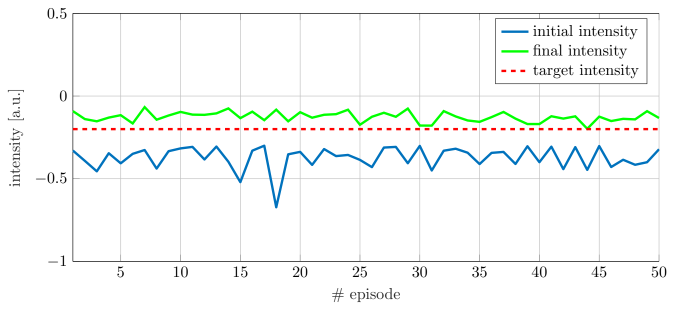

Figure 5.6: Intensity during 50 episodes of iLQR on AWAKE. Initial, final, and target intensity in blue, green, and dashed red respectively.

5.4 Discussion

Both of the presented experiments are part of a program which aims to introduce advanced control algorithms able to increase the stability and reproducibility of the accelerator operation at the CERN complex. The collected results prove the effectiveness and sample-efficiency of the two methods. Indeed, the NAF has been able to successfully face the trajectory steering on AWAKE as well as on Linac 4. After reaching the convergence it improved the RMS of both systems above the desired value in about 2 steps. Similar results have been obtained on AWAKE by the iLQR that identifying the system dynamics using 200 data-points reached the target at each verification episode in a few steps.

Conclusions

Several approaches to automatically control and improve performance in particle accelerators have been introduced. In particular, the study focused on maximizing the outcome of FERMI the seeded free-electron laser at Elettra Sincrotrone Trieste by dealing with the transverse overlap between the external optical source, i.e. the seed laser, and the electron beam in the modulator undulator. In this context, two different tasks have been faced, namely (i) the attainment of the optimal working point and (ii) its recovery after machine drifts.

The algorithms deployed to face these tasks have been introduced in Chapter 3 and they mainly belong to the reinforcement learning field. From the MFRL class, the Q-Learning and its extension to continuous action space given by Normalized Advantage Function have been successfully addressed to reach the optimal working point starting from random initializations. To solve the same problem also an MBRL approach, the Anchored-Ensemble Dyna, has been proposed. Its results proved the goodness in dealing with problems in the accelerator optimization, capturing the epistemic and aleatoric uncertainty of model and data. The proposed variants, based on TRPO and SAC algorithms, build an ensemble of models and use it to find the optimal policy. To conclude with task (i), two additional techniques outside RL have been applied to FERMI. These are Gradient Ascent and Iterative Linear Quadratic Regulator, that manage the trajectory alignment by identifying the system with a neural network trained on collected data. Concerning the recovery of performance after machine drift (ii), it has been addressed by another reinforcement learning algorithm, the non-episodic NPG REINFORCE. This approach has been successfully applied to FERMI restoring the radiation intensity after the introduction of some system perturbations.

In addition, the collaboration with CERN helped to achieve promising results in the application of RL on the challenging problems in the particle accelerators field. The teamwork in improving the performance of AWAKE and Linac 4 allowed the sharing of knowledge and experience on accelerator controls that led to the successful results presented in Chapter 5.

In conclusion, the work introduced an innovative approach to automatically improve the performance of particle accelerators. The main results obtained on FERMI at Elettra Sincrotrone Trieste by applying several RL algorithms, were shown in this thesis to be able to solve the control problem in a feasible operational setup. Some important improvements have been obtained, especially in terms of sample-efficiency reducing the number of interactions required. The most interesting and promising approach is given by the AE-Dyna that is the first uncertainty aware MBRL algorithm to be applied in improving particle accelerators performance.

The study here proposed provides a solid starting point from which continuing the investigation on RL techniques to introduce even more smart and intelligent approaches in controlling particle accelerators. Extending the field of application of the proposed methods to different systems and problems, indeed, is the necessary step to increase the presence of artificial intelligence in the control rooms of the various facilities involved in particle physics.

Bibliography

[1] T. Mitchell, *Machine Learning*, ser. McGraw-Hill international editions - computer science series. McGraw-Hill Education, 1997. [Online]. Available: https://books.google.it/books?id=xOGAngEACAAJ

[2] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad, "Machine learning at the energy and intensity frontiers of particle physics," *Nature*, vol. 560, no. 7716, pp. 41–48, 2018.

[3] C. Emma, A. Edelen, M. Hogan, B. O'Shea, G. White, and V. Yakimenko, "Machine learning-based longitudinal phase space prediction of particle accelerators," *Physical Review Accelerators and Beams*, vol. 21, no. 11, p. 112802, 2018.

[4] A. Edelen, N. Neveu, M. Frey, Y. Huber, C. Mayes, and A. Adelmann, "Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems," *Physical Review Accelerators and Beams*, vol. 23, no. 4, p. 044601, 2020.

[5] E. Fol, J. C. de Portugal, G. Franchetti, and R. Tomás, "Optics corrections using machine learning in the LHC," in *Proceedings of the 2019 International Particle Accelerator Conference, Melbourne, Australia*, 2019.

[6] G. Azzopardi, B. Salvachua, G. Valentino, S. Redaelli, and A. Muscat, "Operational results on the fully automatic LHC collimator alignment," *Physical Review Accelerators and Beams*, vol. 22, no. 9, p. 093001, 2019.

[7] R. Müller, A. Balzer, P. Baumgärtel, O. Sauer, G. Hartmann, and J. Viefhaus, "Modernization of experimental data taking at bessy ii," in *MOCPL02, Proceedings of this conference*, 2019.

[8] A. Edelen, C. Mayes, D. Bowring, D. Ratner, A. Adelmann, R. Ischebeck, J. Snuverink, I. Agapov, R. Kammering, J. Edelen *et al.*, "Opportunities in machine learning for particle accelerators," *arXiv preprint arXiv:1811.03172*, 2018.

[9] A. Edelen, S. Biedron, B. Chase, D. Edstrom, S. Milton, and P. Stabile, “Neural networks for modeling and control of particle accelerators,” *IEEE Transactions on Nuclear Science*, vol. 63, no. 2, pp. 878–897, 2016.

[10] A. L. Edelen, J. P. Edelen, L. RadiaSoft, S. G. Biedron, S. V. Milton, and P. J. van der Slot, “Using neural network control policies for rapid switching between beam parameters in a free-electron laser,” *NIPS*, 2017.

[11] S. Hirlaender, V. Kain, and M. Schenk, “New paradigms for tuning accelerators: Automatic performance optimization and first steps towards reinforcement learning at the CERN low energy ion ring,” URL: https://indico.cern.ch/event/784769/contributions/3265006/attachments/1807476/2950489/CO-technical-meeting-_Hirlaender.pdf, 2019, 2nd ICFA Workshop on Machine Learning for Charged Particle Accelerators.

[12] V. Kain, S. Hirlander, B. Goddard, F. M. Velotti, G. Zevi Della Porta, N. Bruchon, and G. Valentino, “Sample-efficient reinforcement learning for CERN accelerator control,” *Physical Review Accelerators and Beams*, 2020.

[13] L. V. Ramirez, T. Mertens, R. Mueller, J. Viefhaus, and G. Hartmann, “Adding machine learning to the analysis and optimization toolsets at the light source BESSY II,” *ICALEPCS*, 2019.

[14] N. Bruchon, G. Fenu, G. Gaio, M. Lonza, F. A. Pellegrino, and E. Salvato, “Toward the application of reinforcement learning to the intensity control of a seeded free-electron laser,” in *2019 23rd International Conference on Mechatronics Technology (ICMT)*. IEEE, 2019, pp. 1–6.

[15] N. Bruchon, G. Fenu, G. Gaio, M. Lonza, F. H. O’Shea, F. A. Pellegrino, and E. Salvato, “Basic reinforcement learning techniques to control the intensity of a seeded free-electron laser,” *Electronics*, vol. 9, no. 5, p. 781, 2020.

[16] F. O’Shea, N. Bruchon, and G. Gaio, “Policy gradient methods for free-electron laser and terahertz source optimization and stabilization at the FERMI free-electron laser at Elettra,” *Physical Review Accelerators and Beams*, vol. 23, no. 12, p. 122802, 2020.

[17] S. Hirlaender and N. Bruchon, “Model-free and bayesian ensembling model-based deep reinforcement learning for particle accelerator control demonstrated on the fermi fel,” *arXiv preprint arXiv:2012.09737*, 2020.

[18] N. Bruchon, G. Fenu, G. Gaio, M. Lonza, F. A. Pellegrino, and L. Saule, “Free-electron laser spectrum evaluation and automatic optimization,”

Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 871, pp. 20–29, 2017.

[19] L. H. Yu, "Generation of intense UV radiation by subharmonically seeded single-pass free-electron lasers," *Physical Review A*, vol. 44, no. 8, p. 5178, 1991.

[20] E. Allaria, L. Badano, S. Bassanese, F. Capotondi, D. Castronovo, P. Cinquegrana, M. Danailov, G. D'Auria, A. Demidovich, R. De Monte *et al.*, "The FERMI free-electron lasers," *Journal of synchrotron radiation*, vol. 22, no. 3, pp. 485–491, 2015.

[21] E. Allaria, R. Appio, L. Badano, W. Barletta, S. Bassanese, S. Biedron, A. Borga, E. Busetto, D. Castronovo, P. Cinquegrana *et al.*, "Highly coherent and stable pulses from the FERMI seeded free-electron laser in the extreme ultraviolet," *Nature Photonics*, vol. 6, no. 10, p. 699, 2012.

[22] E. Allaria, D. Castronovo, P. Cinquegrana, P. Craievich, M. Dal Forno, M. Danailov, G. D'Auria, A. Demidovich, G. De Ninno, S. Di Mitri *et al.*, "Two-stage seeded soft-x-ray free-electron laser," *Nature Photonics*, vol. 7, no. 11, p. 913, 2013.

[23] G. Gaio, M. Lonza *et al.*, "Evolution of the FERMI beam based feedbacks," in *Proc. ICALEPCS*, 2013, pp. 1362–1365.

[24] G. Gaio, M. Lonza, N. Bruchon, and L. Saule, "Advances in automatic performance optimization at FERMI," *ICALEPCS*, 2018.

[25] K. B. Ariyur and M. Krstić, *Real-time optimization by extremum-seeking control.* John Wiley & Sons, 2003.

[26] G. Gaio and M. Lonza, "Automatic FEL optimization at FERMI," *ICALEPCS, Melbourne*, 2015.

[27] M. Veronese, E. Allaria, P. Cinquegrana, E. Ferrari, F. Rossi, P. Sigalotti, and C. Spezzani, "New results of FERMI FEL1 EOS diagnostics with full optical synchronization," *IBIC2014, Monterey, California*, 2014.

[28] M. Veronese, M. Danailov, and M. Ferianis, "The electro-optic sampling stations for FERMI@Elettra, a design study," *TUPTPF026, Proceedings of BIW08, Tahoe City, California*, 2008.

[29] M. Veronese, A. Abrami, E. Allaria, M. Bossi, M. Danailov, M. Ferianis, L. Fröhlich, S. Grulja, M. Predonzani, F. Rossi *et al.*, "First operation of the electro optical sampling diagnostics of the FERMI@ Elettra FEL," *IBIC*, vol. 12, p. 449, 2012.

[30] S. Cleva, L. Pivetta, P. Sigalotti *et al.*, "BeagleBone for embedded control system applications," in *Proc. ICALEPCS2013*, 2013.

[31] A. Götz, E. Taurel, J. Pons, P. Verdier, J. Chaize, J. Meyer, F. Poncet, G. Heunen, E. Götz, A. Buteau *et al.*, "TANGO a CORBA based Control System," *ICALEPCS2003, Gyeongju, October*, 2003.

[32] "Tango-Controls Matlab-Bindings," URL: https://github.com/tango-controls/matlab-binding.

[33] "PyTango," URL: https://pytango.readthedocs.io/en/stable/.

[34] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: A system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI16)*, 2016, pp. 265–283.

[36] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[37] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[38] S. Levine, "Deep reinforcement learning," URL: http://rail.eecs.berkeley.edu/deeprlcourse/, 2020, cS 285 at UC Berkeley.

[39] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[40] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, 1999, pp. 278–287.

[41] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[42] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning*, 2016, pp. 2829–2838.

[43] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[44] T. Zhao, H. Hachiya, G. Niu, and M. Sugiyama, “Analysis and improvement of policy gradient estimation,” in *Advances in Neural Information Processing Systems*, 2011, pp. 262–270.

[45] S. M. Kakade, “A natural policy gradient,” in *Advances in neural information processing systems*, 2002, pp. 1531–1538.

[46] E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, “Benchmarking model-based reinforcement learning,” *arXiv preprint arXiv:1907.02057*, 2019.

[47] K. Gallagher, K. Charvin, S. Nielsen, M. Sambridge, and J. Stephenson, “Markov chain monte carlo (MCMC) sampling methods to determine optimal models, model resolution and model choice for earth science problems,” *Marine and Petroleum Geology*, vol. 26, no. 4, pp. 525–535, 2009.

[48] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *international conference on machine learning*, 2016, pp. 1050–1059.

[49] T. Pearce, F. Leibfried, and A. Brintrup, “Uncertainty in neural networks: Approximately bayesian ensembling,” in *International conference on artificial intelligence and statistics.* PMLR, 2020, pp. 234–244.

[50] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, “Model-ensemble trust-region policy optimization,” *arXiv preprint arXiv:1802.10592*, 2018.

[51] R. S. Sutton, “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” in *Machine learning proceedings 1990.* Elsevier, 1990, pp. 216–224.

[52] ——, “Dyna, an integrated architecture for learning, planning, and reacting,” *ACM Sigart Bulletin*, vol. 2, no. 4, pp. 160–163, 1991.

[53] ——, “Planning by incremental dynamic programming,” in *Machine Learning Proceedings 1991.* Elsevier, 1991, pp. 353–357.

[54] S. Hirlaender, N. Bruchon, V. Kain, A. Scheinker, B. Goddard, G. Valentino, F. Velotti, and the ML Coffee, “Towards artificial intelligence in accelerator operation,” URL: https://indico.gsi.de/event/11539/attachments/33466/43622/Praesentation_Simon_Hirlaender.pdf, 2020, gSI Accelerator Seminar.

[55] T. Pearce, F. Leibfried, A. Brintrup, M. Zaki, and A. Neely, “Uncertainty in neural networks: Approximately bayesian ensembling,” *arXiv preprint arXiv:1810.05546*, 2018.

[56] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[58] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[59] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*, 2004, pp. 222–229.

[60] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Machine Learning: ECML 2005*, J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 437–448.

[61] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, J. P. How *et al.*, "A tutorial on linear function approximators for dynamic programming and reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 6, no. 4, pp. 375–451, 2013.

[62] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[63] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[64] S. Hirlaender, "MathPhysSim/PER-NAF: Initial release," URL: 10.5281/ZENODO.4271647, 2020, open Access.

[65] S. Hirlaender and N. Bruchon, "MathPhysSim/FERMI_RL_Paper: Initial release," URL: 10.5281/ZENODO.4271581, 2020, mIT License.

[66] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 10th ed. US Government printing office, 1972, vol. 55.

[67] M. Chanel, "LEIR: the low energy ion ring at CERN," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 532, no. 1-2, pp. 137–143, 2004.