Karlsruhe Institute of Technology

Automatic Control of Linear Particle Accelerators with Machine Learning Methods

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

von der KIT-Fakultät für Physik
des Karlsruher Instituts für Technologie (KIT)

angenommene

Dissertation

von

M. Sc. Chenran Xu

aus Shanghai

Tag der mündlichen Prüfung: 29. November 2024
Erster Gutachter: Prof. Dr. Anke-Susanne Müller
Zweiter Gutachter: Prof. Dr. Torben Ferber

Abstract

Particle accelerators are among the most complex physical systems globally, with their design and operation posing significant challenges due to versatile operation modes, stringent beam quality requirements, and the demand for high availability. This dissertation provides a comprehensive overview of integrating machine learning (ML) techniques throughout the life cycle of linear particle accelerators, from design to operation.

In the simulation, the neural networks (NNs) are used as surrogate models for fast, high-quality predictions in place of computationally expensive physics simulations and virtual diagnostics for real-time, non-destructive measurements. In the design phase, parallel Bayesian optimization (BO) is introduced for more efficient parameter optimization. A fast-executing backward-differentiable beam dynamics simulation code Cheetah is developed. Some of its application cases are highlighted in this dissertation, including gradient-based simulated optimization and support for other algorithms.

This dissertation further explored the use of convolutional neural networks (CNNs) for control of spatial light modulators for laser pulse shaping, with results demonstrated at the FLUTE accelerator. Such fine-grained photo-injector laser pulse shaping is expected to allow tailored generation of electron bunches and increase the accelerator's dynamic range.

For the online tuning of the accelerator, the performance of BO and reinforcement learning (RL) is compared for online accelerator tuning, with results showing that BO is a turn-key tuning solution and RL has superior performance at the cost of increased upfront engineering effort. Several proposed techniques for building robust and generalizable ML-based controllers across different accelerators are discussed, including domain randomization, meta-RL, and GP-MPC which combines the strengths of BO and RL for beam trajectory tuning tasks.

These advancements highlight the potential of ML-driven solutions in improving the accelerator design, tuning, and control. These preliminary studies pave the way for the development and operation of more efficient and reliable accelerators in the future.

Contents

Abstract . . . . . **i**

1. Introduction **1**
1.1. Terminology 3
1.2. Machine Learning Enabled Future Accelerator Operation Scheme 4
1.3. Contributions of this Dissertation 5
1.4. Collaborators 7

2. Beam Dynamics and Linear Particle Accelerators **9**
2.1. Beam Dynamics in Particle Accelerators 9
2.1.1. Multipole Expansion of Transverse Magnetic Fields 10
2.1.2. Hamiltonian of Charged Particles 11
2.1.3. Transfer Maps 12
2.1.4. Collective Effects 14
2.2. Accelerator-based Radiation Generation 14
2.3. Linear Particle Accelerators 17
2.4. Accelerators Studied in this Dissertation 18
2.4.1. FLUTE 18
2.4.2. ARES 19
2.4.3. European XFEL 20

3. Machine Learning Methods **21**
3.1. Neural Networks 21
3.1.1. Automatic Differentiation and Gradient Descent Optimization . . 23
3.2. Bayesian Optimization 25
3.2.1. Gaussian Process Modeling 26
3.2.2. Acquisition Function 30
3.2.3. Tailoring Bayesian Optimization Methods for Particle Accelerators 31
3.3. Introduction to Reinforcement Learning 32
3.3.1. Reinforcement Learning Problem Formulation 33
3.3.2. Policy and Value Functions 34
3.3.3. Basic Learning Concepts 36
3.3.4. Policy Gradient Methods 38
3.3.5. Modern RL Algorithms 38
3.3.6. Reinforcement Learning Applications for Accelerators 41

4. Applying Machine Learning Methods for Accelerator Simulation 43
4.1. Surrogate Modeling of FLUTE 43
4.1.1. Training the Surrogate Model 45
4.1.2. Applications of the Surrogate Model 47
4.2. Simulated Optimization for Intense THz Radiation 53
4.2.1. Calculation of the Coherent Synchrotron Radiation Generation at FLUTE 53
4.2.2. Parallelized Bayesian Optimization 55
4.2.3. Optimization Settings 57
4.2.4. Optimization Results 58
4.3. Differentiable Beam Dynamics Simulation 62
4.3.1. Simulation Code Cheetah 62
4.3.2. Differentiable Modeling of the THz CSR Generation at FLUTE . . 64
4.4. Bayesian Optimization with Physics-informed Prior 68
4.5. Summary Machine Learning Assisted Simulated Optimization 71

5. Photo-Injector Laser Pulse Shaping 75
5.1. Motivation for Photo-injector Drive Laser Shaping 75
5.2. Laser Shaping with Spatial Light Modulators 78
5.2.1. Implementing the Laser Modulation Setup at FLUTE 80
5.2.2. Controlling the Spatial Light Modulator 81
5.3. Demonstration of Transverse Laser Modulation 83
5.3.1. Laser Modulation Correction using Convolutional Neural Network 83
5.4. Transverse Spatial Light Modulator Setup for FLUTE Photo-Injector Laser 87
5.5. Outlook for Full Laser Modulation at FLUTE 90
5.6. Summary Machine Learning Enabled Laser Pulse Shaping 93

6. Autonomous Online Accelerator Tuning 95
6.1. SASE Tuning at European XFEL 96
6.1.1. Implementing Bayesian optimization for SASE tuning 97
6.1.2. European XFEL SASE tuning results 99
6.2. Reinforcement Learning Control for FLUTE Tuning 103
6.2.1. Formulation of the FLUTE Tuning as a Reinforcement Learning Task 103
6.2.2. Implementation of FLUTE Tuning in a Reinforcement Learning Framework 104
6.3. Comparing Reinforcement Learning with Bayesian Optimization for Online Tuning 107
6.3.1. Transverse Beam Tuning at ARES Experimental Area 107
6.3.2. ARES Experimental Area Tuning with Reinforcement Learning . 109
6.3.3. ARES Experimental Area Tuning with Bayesian Optimization . . 111
6.3.4. Benchmarking Reinforcement Learning and Bayesian Optimization results 112
6.3.5. Practical Challenges for Online Accelerator Tuning 116
6.3.6. Inference Time 122
6.3.7. Discussion on the Benchmark Study 122

6.4. Towards Generalizable Machine Learning-based Controller 123
6.4.1. Generalizable Reinforcement Learning Agent with Domain Randomization 123
6.4.2. Fast Reinforcement Learning Deployment with Meta-Learning . 127
6.4.3. Towards More Sample Efficient and Explainable Machine Learning-based Controller 131
6.5. Summary Machine Learning-based Online Accelerator Tuning 132

7. Summary and Outlook 135

List of Figures 139

List of Acronyms 143

List of Publications 147

Bibliography 151

AI Assistance Disclosure 167

A. Appendix 169
A.1. Feature importance study of the Surrogate Model 169
A.2. Kernel Density Estimation 171
A.3. Laser Modulation Results with Convolutional Neural Network 173
A.4. Laser Modulation with Zernike Polynomials 174
A.5. Non-Machine Learning Algorithms for Accelerator Tuning 176
A.5.1. Nelder-Mead Simplex 176
A.5.2. Extremum Seeking 179
A.6. FLUTE Reinforcement Learning Training Configurations 180
A.7. Benchmarking Bayesian Optimization Implementations in the Xopt Package on EA Tuning Task 181
A.8. Lattice-agnostic Reinforcement Learning Training Configurations 182
A.9. Code Availability 183

Acknowledgments 185

1. Introduction

Particle accelerators are pivotal engines that drive our understanding of the world and facilitate groundbreaking discoveries across a wide range of scientific disciplines [1]. High-energy particle colliders, for instance, enable the discovery of new particles and push the boundaries of our knowledge in high-energy physics. In addition, dedicated light sources, which utilize high-energy electrons to generate high-brightness synchrotron radiation ranging from THz to X-ray, serve as essential tools for discovering new materials and illuminating atomic and molecular structures across various scientific fields.

X-ray free electron lasers (FELs) driven by linear accelerators are excellent candidates as the new generation light sources, capable of providing femtosecond coherent pulses and peak intensities orders of magnitude higher than the storage ring-based third generation sources [2]. Since the first soft X-ray FEL FLASH started operation in 2005, several FELs have been built and proposed worldwide, including the LCLS [3], SACLA [4], FERMI, PAL-XFEL [5], EuXFEL [6], SwissFEL [7], and SXFEL [8]. The timeline when these accelerators started operation is indicated in Fig. 1.1. The existing accelerators are also undergoing continuous upgrades, improving the stability of the operation and allowing more operation modes. New beamlines were being built to provide radiation with different wavelengths. For example, the recently finished LCLS-II upgrade brings the repetition rate up to 1 MHz using superconducting cavities. In the next decade, more FELs are expected to be constructed and start operation, such as SHINE and S3FEL, and existing facilities like the SwissFEL and SACLA will also undergo major upgrades. To meet the requests from user experiments, the FELs all aim to provide ultrashort coherent light pulses with high output power. This requires a stable operation with a high repetition rate, high bunch charge, low emittance, and short bunch length.

These modern particle accelerators are some of the most complex scientific systems with sizes up to multiple kilometers, consisting of thousands of components and subsystems like magnets, RF cavities, and diagnostic devices. Detailed first-principles physics-based simulation models are developed to support the operation of accelerators and various predefined subroutines exist to control certain subsystems or perform specific beam-tuning tasks automatically. The extreme operation conditions such as ultrashort bunches with charges are very challenging in accelerator modeling. Higher-order effects and collective effects, which are necessary to model the beam dynamics in such conditions, need considerably more computation resources. To conduct the simulation in a feasible time, certain approximations need to be made which introduce errors in the simulation model. The accuracy of the simulation is further limited by real-world errors, as all the components in particle accelerators are physical devices. They are subject to environmental variables like temperature, magnetic hysteresis, field imperfections, calibration errors, and misalignment of the components. Most of these issues are well-known and can be modeled and compensated by dedicated routines. For example, the hysteresis effects can

be explicitly modeled for individual magnets and mitigated by doing power cycles. The positioning of the magnets can be improved in dedicated alignment campaigns. As such processes take up a considerable amount of time, light sources with a requirement of high availability can not afford to perform them frequently enough. Additionally, systematic drifts, such as those caused by temperature changes over the course of operations are not accounted for in the simulation models, but they can hamper the machine's performance. As a result, mismatches are inevitable between the real-world operation condition and the ideal working point. In the commissioning phase or during long-term operation, manual tuning by operators is still frequently required to bring the machine to the target working point or to maintain a certain performance level in the operation of particle accelerators nowadays.

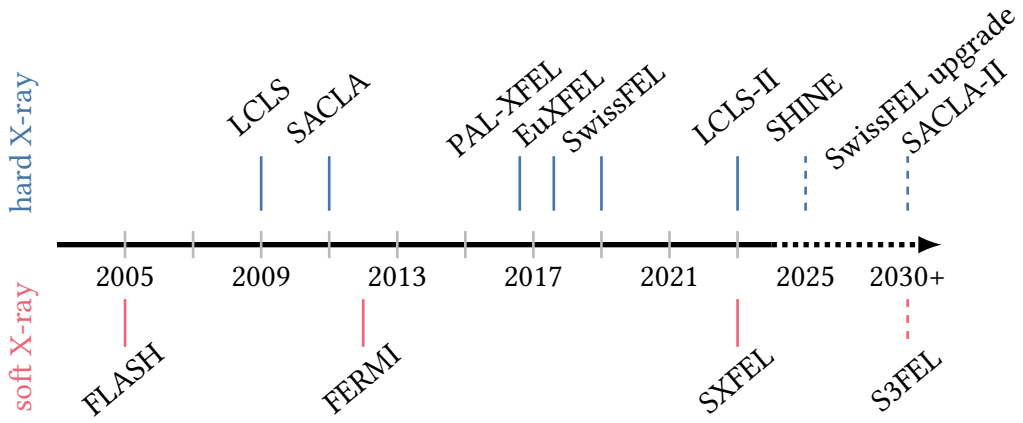


Figure 1.1.: Timeline indicating the time when X-ray free electron lasers (FELs) started in user operation. The dashed lines include the expected time for upgrades of existing facilities and the operation start of new FELs. [2]

Specifically, being single-pass accelerators, linac-based FELs have multiple operation modes and often need to quickly switch between those to meet the requests of different beamline experiments. For example, SACLA needs to simultaneously provide bunches for two undulator beamlines and injection into a storage ring at 60 Hz, demanding concurrent optimization of three different working points. For such facilities, the downtime needs to be minimized, and the desired beam needs to be constantly delivered with high quality. This demand would only increase as future accelerators have finer control of the bunches and can provide radiation tailored to individual experiments. Therefore, versatile and intelligent controllers would be essential to operate the accelerators efficiently and reliably under various conditions.

The integration of ML and artificial intelligence (AI) techniques offers a promising strategy to address the challenges faced by future accelerator operations. They can model the remaining mismatches between the simulation model and the real-world accelerators in a data-driven way and provide fast and intelligent control both in setting up the accelerator and maintaining their peak performance. Due to advancements in algorithms and

computing hardware, various ML methods have been successfully applied to accelerator operation and more promising application scenarios are proposed [9, 10], concurrent to the period when this dissertation is conducted. Both the European strategy [11] and the US Snowmass reports [12, 13] have identified ML and AI as important assets for aiding future accelerator operations. The integration of ML/AI techniques has been considered from the beginning of the Electron Ion Collider (EIC) [14], one of the largest particle accelerators to be built in the next decades. The ML/AI techniques are expected to enhance the precision, versatility, and efficiency of particle accelerators, making them more effective tools for scientific discovery and technological advancement.

A large portion of the early applications of ML in accelerators is dedicated to developing customized methods for a certain task. Despite their success, the tailored methods are seldom transferred to other tasks or routinely deployed in operation. With the maturing of ML algorithms and the increasing knowledge in the accelerator community, it is now essential to investigate how to provide new ML methods as ready-to-use solutions in the toolbox of accelerators and how the individual ML methods can work in orchestration to further increase their performances. This aim to increase the standardization and reusability of the methods is also reflected in the recent development of software tools and packages in accelerators, including the general-purpose optimization packages like Xopt [15] and Optimas [16], and facility-agnostic online tuning frameworks like Badger [17] and GeOFF [18].

This dissertation provides a holistic study of the applications of ML/AI methods for electron linear accelerators, ranging from efficient simulation and modeling, laser and electron bunch shaping, to online accelerator tuning. In addition to the performance of the methods, this dissertation also discusses the practical details of the deployment and reviews the potential of scalability for further accelerator operations.

Although the studies presented here all focused on linear electron accelerators, the same techniques can easily be transferred to comparable tasks in storage rings, colliders, and novel accelerator types like laser-plasma accelerators.

1.1. Terminology

In the context of applying ML algorithms to particle accelerators, several terminologies are closely connected and often used ambiguously in different scientific communities. Some clarifications are provided below for their meaning throughout the dissertation.

- **Optimization** is the process of finding the best set of parameters that maximize (or minimize) a specific objective function. This can be done either through real-time interactions with the accelerator, or independent of accelerator operation using the historical data or in a physics-based simulation model.

- **Control** is the process of maintaining the system in a desired state under internal changes or external disturbances, often achieved via a feedback controller. One example is keeping the beam at a certain position over a period of time. The term control is mostly used when an explicit time dependency is present in the system.

Task	Methods & concepts	Application examples
Modeling, virtual diagnostic	• Neural network • Differentiable simulation	• Phase space prediction [19, 20, 21] • Beam reconstruction [22, 23] • Accelerator design [24]
Anomaly detection	• Autoencoder • Recurrent network	• Faulty diagnostic [25, 26] • RF failure, interlock [27]
Optimization	• Bayesian optimization • NN-assisted opt.	• Radiation intensity [28, 29, 30] • Injection efficiency [31, 32] • Emittance, beam energy [33, 34]
Control	• Reinforcement learning • Extremum seeking • Model-based control	• Trajectory control [35, 36] • Microbunching instability [37, 38] • Power supplies [39]

Table 1.1.: Possible applications of machine learning for particle accelerator operation

- **Tuning** is a common type of task performed at accelerators, where the parameters are continuously adjusted in real-time to maintain or improve certain aspects of the performance. Tuning often contains both aspects in optimization and control, i.e. both for finding the optimal condition initially and maintaining it throughout operation.
- **Online** means that the algorithm is running and interacting with the accelerator in real-time during the operation. An online ML algorithm can adapt and make decisions on the fly when receiving new data.
- **Offline** means that the process is done with respect to a fixed batch of data, without requiring real-time interactions with the accelerator. Examples of offline operations are the training process prior to the deployment, and slow post-processing of the gathered data.

1.2. Machine Learning Enabled Future Accelerator Operation Scheme

As the complexity of the system increases and beam quality requirements become ever more stringent, the operation of future accelerators will certainly become more autonomous. Some successful applications are listed in Table 1.1. The integration of ML algorithms will bring a paradigm shift in the operation of future accelerators. Instead of spending time repeatedly adjusting the individual settings to achieve the target beam parameters, future accelerator operators will be able to set higher-level objectives, such as beam phase space or radiation properties. Intelligent ML algorithms will then adjust the accelerator to the desired state safely and efficiently. This requires an orchestration of various ML methods. For example, the data-driven models and virtual diagnostics would provide

high-resolution noise-free signals that can guide the optimization algorithm or controller and avoid exploring unsafe settings. On a broader scope, the overall operation schedule can be optimized to minimize the downtime further. Models that are trained to predict anomalies and forecast component failures can both avoid interlock during operation and make informed predictions on maintenance requirements.

1.3. Contributions of this Dissertation

Among these possible application scenarios, this dissertation focuses on the integration of ML-based methods into the life-cycle of electron linear accelerators, from simulated optimization of working points to online tuning during operation. An overview of the covered topics is illustrated in Fig. 1.2.

In the design stage, machine learning algorithms like Bayesian optimization (BO) can greatly reduce the resources and time required to find an accelerator configuration to produce the target beam. The simulation data, as well as measurement data, can be further used to train a surrogate model, often using a NN, to map the accelerator settings to the output beam parameters. Once trained, the surrogate model is usually computationally cheap to evaluate and accurate in predictions. During operation, it can be used as a virtual diagnostic, providing non-destructive high-resolution information on the beam. The emerging differentiable programming technique [40, 41] will also become an essential tool for accelerator simulation and modeling. It allows both efficient simulated optimization using gradient information in the design stage and fast prediction like a data-driven surrogate model in an online setting. The measured beam parameters can in turn be used to calibrate and identify the mismatch between the model and the real-world accelerator [42].

During operation, machine learning methods can be applied to detailed beam control and enlarge the achievable range of beam charge and emittance. One particular scenario is the photo-injector laser shaping using programmable devices with a high number of actuators, which allows precise control of the initial bunch distribution. Most importantly, the autonomous accelerator operation relies on ML-based controllers for automatic tuning to reach the target beam parameters given by the operators, and maintain the peak performance in the presence of system drifts. They obtain measurements from diagnostic devices and additional information from fast-executing models. Based on the data received, they make real-time intelligent decisions about the new accelerator settings, either by setting the actuators like magnet strengths directly or by setting some target parameter for other low-level controllers in subsystems.

This dissertation is structured in the following way:

Chapter 2 This chapter introduces basic accelerator theory concepts relevant to this dissertation, including beam particle dynamics and accelerator-based radiation generation. The accelerators studied in this dissertation are also introduced.

Chapter 3 This chapter introduces the concepts of the ML methods investigated in this dissertation, including NN as a basic building block, and BO and RL as two of the most promising optimization and control methods. It also contains a short review

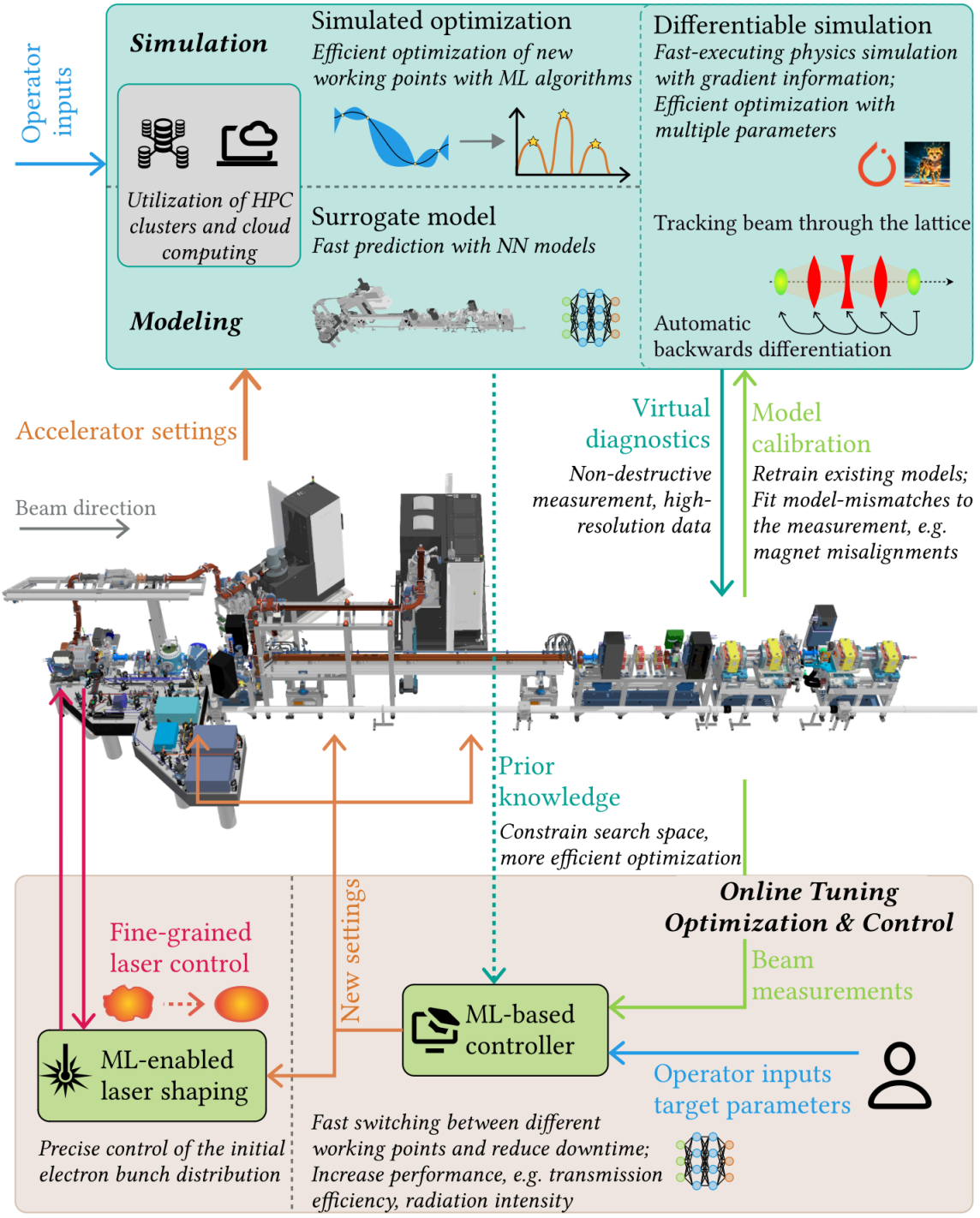


Figure 1.2.: Overview of applications of ML methods in the operation scheme of electron linear accelerators.

of the existing ML applications in the particle accelerator field with a focus on the online tuning aspect.

Chapter 4 This chapter showcases how ML methods can be applied to accelerator design and optimization in a simulated study. Using the FLUTE lattice as an example, three applications are presented: NN-based surrogate modeling of lattice sections for fast inference and virtual diagnostics, parallelized BO for guided simulation optimization, and utilization of a novel differentiable simulation code Cheetah for more efficient optimization.

Chapter 5 This chapter demonstrates the photo-injector laser shaping as an example of ML methods increasing the accelerator operation range. Spatial light modulators (SLMs) are deployed for photo-injector laser shaping at the accelerator FLUTE, allowing a versatile control of the laser pulse and the resulting initial electron bunch distribution. A proof-of-principle study for a transverse setup with a test laser is shown, using a NN-assisted control for generating higher-quality laser profiles. The laser modulation concept is subsequently tested with the FLUTE drive laser. The practical aspects of SLM-based laser shaping and the possibility of having full 3D modulation of the FLUTE electron bunch are discussed.

Chapter 6 This chapter contains the simulation and experimental results on applying BO and RL to online-tuning tasks at multiple accelerator facilities, including the European XFEL, FLUTE, and ARES. For each method, its advantages and shortcomings in different task scenarios are discussed. Various extensions and adaptions to the algorithms beyond their normal settings are presented. Based on these results, guidelines are proposed to combine the strengths of existing methods and improve the scalability and robustness of ML-based controller for future accelerator operations.

1.4. Collaborators

Some topics of this dissertation were developed jointly with Jan Kaiser, including the differentiable simulation code *Cheetah* in Section 4.3, designing the RL framework, agent training, and conducting experiments at the ARES accelerator in Section 6.3. In Chapter 5, the optical setups and experimental studies are carried out with the help of Matthias Nabinger and Carl Sax.

2. Beam Dynamics and Linear Particle Accelerators

This chapter starts with an introduction to the fundamentals of beam dynamics in particle accelerators, a more detailed treatment can be found in textbooks [43, 44, 45]. The coherent synchrotron radiation is introduced as an important source of light generated at particle accelerators. The chapter concludes with an overview of the accelerator facilities FLUTE, ARES, and European XFEL, where experimental studies are conducted in the scope of this dissertation.

2.1. Beam Dynamics in Particle Accelerators

A particle moving in an electromagnetic field with charge q experiences the Lorentz force

$$\boldsymbol{F} = q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}), \tag{2.1}$$

where $\boldsymbol{E}$ is the electric field and $\boldsymbol{B}$ is the magnetic field. They are related to a magnetic vector potential $\boldsymbol{A}$ and an electric scalar potential ϕ

$$\begin{aligned} \boldsymbol{B} &= \nabla \times \boldsymbol{A} \\ \boldsymbol{E} &= -\nabla\phi - \partial\boldsymbol{A}/\partial t \end{aligned} \tag{2.2}$$

For relativistic particles moving at the speed of light $|\boldsymbol{v}| \approx c$, the magnetic fields have a much stronger impact on the particle motion than what could be technically achieved by using electric fields. As a result, magnetic fields are mostly used to deflect and focus the particles along the trajectories in accelerators. Nevertheless, the electric fields are essential for particles to gain energies, i.e. the acceleration process.

The energy and momentum of a particle are given by

$$\begin{aligned} E &= \gamma m c^2 \\ \boldsymbol{p} &= \boldsymbol{\beta}\gamma m c, \end{aligned} \tag{2.3}$$

where m is the rest mass, $\boldsymbol{\beta} = \boldsymbol{v}/c$ is the relativistic velocity, and $\gamma = 1/\sqrt{1 - |\boldsymbol{\beta}|^2}$ is the relativistic Lorentz factor. In the case of a constant vertical magnetic field $\boldsymbol{B} = (0, B, 0)^\top$, for example produced by a dipole magnet, the particle moving with the velocity $\boldsymbol{v} = (0, 0, v)^\top$ experiences a centripetal force which bends its path in a circular trajectory

$$\frac{\gamma m v^2}{\rho} = qvB \implies \frac{1}{\rho} = \frac{q}{p}B. \tag{2.4}$$

The bending radius ρ increases linearly with the particle momentum and the quantity $B\rho$ is often referred to as *beam rigidity*.

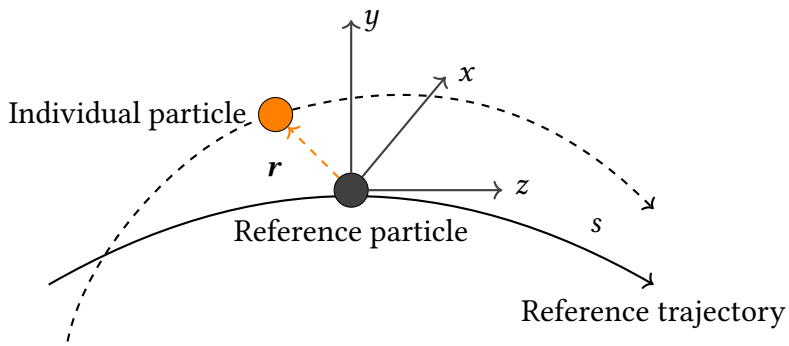


Figure 2.1.: The curvelinear coordinate system used in beam dynamics. The position vector $\boldsymbol{r}$ of an individual particle is defined as the offset to the reference particle on the ideal trajectory at any given longitudinal position s. The unit vector $\boldsymbol{e}_z$ is parallel to the trajectory, $\boldsymbol{e}_x$, and $\boldsymbol{e}_y$ are orthogonal to the trajectory.

2.1.1. Multipole Expansion of Transverse Magnetic Fields

The magnet used for beam transportation in particle accelerators ideally has a static field $\boldsymbol{B} = (B_x, B_y, B_z)^\top$ which does not depend on the longitudinal coordinate, i.e. B_z is constant. In such a case, the transverse fields can be described as the superposition of a set of multipole fields

$$B_y + iB_x = \sum_{n=1}^{\infty} C_n (x + iy)^{n-1}, \tag{2.5}$$

where $C_n \in \mathbb{C}$ are complex coefficients. The index n denotes the *order* of the fields and can be attributed to physical magnets. The dipole field has $n = 1$, the quadrupole field has $n = 2$, and the sextupole field has $n = 3$. A pure multipole magnet has $C_n \neq 0$ for only one order n and is assumed to have a vanishing longitudinal field $B_z = 0$. The solenoid magnet is a special case with $C_n = 0, \forall n$ and $B_z \neq 0$. By introducing a reference field and a reference radius, the multipole expansion can be expressed

$$B_y + iB_x = B_{\text{ref}} \sum_{n=1}^{\infty} (b_n + ia_n) \left(\frac{x + iy}{R_{\text{ref}}} \right)^{n-1}, \tag{2.6}$$

with dimensionless real-numbered coefficients b_n and a_n. As seen in Eq. (2.4), the net effect of a magnetic field on the motion of a charged particle is scaled with the particle charge q and momentum p. It is thus advantageous to define normalized multipole strengths

$$k_n = \frac{q}{p} \frac{\partial^n B_y}{\partial x^n} = n! \frac{B_{\text{ref}}}{R_{\text{ref}}^n} b_{n+1}. \tag{2.7}$$

Expanding the scaled magnetic field around $x = 0$ gives

$$\begin{aligned} \frac{q}{p}B_y &= \frac{1}{0!}\frac{q}{p}B_y \quad + \frac{1}{1!}\frac{q}{p}\frac{\partial B_y}{\partial x}x \quad + \frac{1}{2!}\frac{q}{p}\frac{\partial^2 B_y}{\partial x^2}x^2 \quad + \dots \\ &= \underbrace{k_0}_{\text{dipole}} \quad + \underbrace{k_1 x}_{\text{quadrupole}} \quad + \underbrace{\frac{1}{2}k_2 x^2}_{\text{sextupole}} \quad + \dots \end{aligned} \tag{2.8}$$

Each of these orders represents a special functionality that is desired for the beam dynamics in particle accelerators. For example, the dipoles are used for steering and quadrupoles are used for focusing the beam.

2.1.2. Hamiltonian of Charged Particles

Due to the magnets present in the particle accelerators, the beam transport path is not always a straight line, making the equations of motion in Cartesian coordinates more complicated. It is more convenient to use the *curvelinear coordinate system* instead, also known as the Frenet-Serret system, which is illustrated in Fig. 2.1. The coordinates are locally defined relative to a given distance s along the reference trajectory, i.e. an ideal design path, in an accelerator beam line. The longitudinal $\boldsymbol{e}_z$ unit vector is chosen to be tangential to the trajectory at position s. The horizontal $\boldsymbol{e}_x$ and vertical $\boldsymbol{e}_y$ coordinates are chosen so that they are perpendicular to the longitudinal axis, and together they form a right-handed orthogonal basis $\boldsymbol{e}_z = \boldsymbol{e}_x \times \boldsymbol{e}_y$. Throughout the dissertation, this curvelinear coordinate system is used.

The Hamiltonian of a single relativistic particle moving in electromagnetic field is given by

$$\mathcal{H}_{\text{Cartesian}} = c\sqrt{(\boldsymbol{P} - q\boldsymbol{A})^2 + m^2c^2} + e\phi \tag{2.9}$$

in the Cartesian coordinate system, where $\boldsymbol{P} = \boldsymbol{p} + q\boldsymbol{A}$ denotes the canonical momentum. In particle accelerators, it is usually considered that the scalar potential vanishes $\phi = 0$ and the vector potential only has a longitudinal component $A_x = A_y = 0$, which means the magnetic fields only acts in the transverse plane. Expressed in the curvelinear system as defined above with a curvature of the reference trajectory $h = 1/\rho$, the Hamiltonian becomes

$$\mathcal{H}_{\text{Curvelinear}} = c\sqrt{(\frac{p_z}{1+hx} - qA_z)^2 + p_x^2 + p_y^2 + m^2c^2}, \tag{2.10}$$

The corresponding canonical variables are

$$\boldsymbol{x} = (x, p_x, y, p_y, t, -\mathcal{H}), \tag{2.11}$$

also known as the *phase-space* coordinates. In the limit of $\rho \to \infty$, i.e. the trajectory becomes a straight line, $h \to 0$ and the Hamiltonian reduces to the one in Eq. (2.9) with $\phi = 0$.

Further canonical transformations are commonly performed so that the transverse momenta are also scaled to the *reference particle* moving along the reference trajectory with momentum $p_0 = \beta_0\gamma_0 mc$

$$\begin{aligned}\tilde{p}_x &= \frac{p_x}{p_0} &&= \frac{\beta_x \gamma mc}{p_0} \\ \tilde{p}_y &= \frac{p_y}{p_0} &&= \frac{\beta_y \gamma mc}{p_0},\end{aligned} \tag{2.12}$$

where $\beta_x = \boldsymbol{\beta} \cdot \boldsymbol{e_x}$ is the horizontal velocity and β_y is the vertical velocity. The longitudinal coordinates can also be redefined as

$$\begin{aligned}\tilde{z} &= \frac{z}{\beta_0} - ct \\ \delta &= \frac{E}{p_0 c} - \frac{1}{\beta_0}.\end{aligned} \tag{2.13}$$

The new position $\tilde{z}$ describes the longitudinal displacement of an individual particle with respect to the reference particle. For a particle arriving at the position s earlier than the reference particle $t < 0$, the position is positive $\tilde{z} > 0$. The new momentum δ describes the energy deviation relative to the nominal energy. For the reference particle, and $E = \gamma_0 mc$ and $\delta = 0$.

This leads to a new set of canonical coordinates

$$\tilde{\boldsymbol{x}} = (x, \tilde{p}_x, y, \tilde{p}_y, \tilde{z}, \delta)^\top, \tag{2.14}$$

with the position along the beamline s being the independent variable. The corresponding Hamiltonian becomes

$$\mathcal{H} = \frac{\delta}{\beta_0} - (1 + hx)\sqrt{(\delta + \frac{1}{\beta_0})^2 - \tilde{p}_x^2 - \tilde{p}_y^2 - \frac{1}{\beta_0^2\gamma_0^2}} - (1 + hx)\frac{q}{p_0}A_z \tag{2.15}$$

2.1.3. Transfer Maps

In the following, the tildes are omitted in the normalized phase space coordinates in Eq. (2.14) $\boldsymbol{x} = (x, p_x, y, p_y, z, \delta)^\top$ for the sake of simplicity. For the most simple case of a drift space, where curvature $h = 0$ and the external electromagnetic field vanishes $\boldsymbol{A} = 0$, the Hamiltonian above becomes

$$\mathcal{H}_{\text{Drift}} = \frac{\delta}{\beta_0} - \sqrt{(\delta + \frac{1}{\beta_0})^2 - p_x^2 - p_y^2 - \frac{1}{\beta_0^2\gamma_0^2}}. \tag{2.16}$$

For relativistic particles moving in a particle accelerator, the transverse momenta are much smaller than the longitudinal momenta $p_{x,y} \ll 1$. Using the *paraxial approximation*, the Hamiltonian can be expanded in lower orders of $p_{x,y}$

$$\mathcal{H}_{\text{Drift}} = \underbrace{-1 + \frac{p_x^2}{2} + \frac{p_y^2}{2} + \frac{\delta^2}{2\beta_0^2\gamma_0^2}}_{\mathcal{H}^{(2)}_{\text{Drift}}} + O(\boldsymbol{x}^3) \tag{2.17}$$

where $O(\boldsymbol{x}^3)$ denotes terms that contain dynamical variables to the third order and higher. The equation of motions can be derived from the second-order Hamiltonian for the transverse variables $\{x, p_x, y, p_y\}$

$$\begin{aligned} \frac{\mathrm{d}p_{x,y}}{\mathrm{d}s} &= -\frac{\partial \mathcal{H}^{(2)}_{\text{Drift}}}{\partial x} = 0 \\ \frac{\mathrm{d}x, y}{\mathrm{d}s} &= \frac{\partial \mathcal{H}^{(2)}_{\text{Drift}}}{\partial p_{x,y}} = p_{x,y}, \end{aligned} \tag{2.18}$$

and the longitudinal variables

$$\begin{aligned} \frac{\mathrm{d}\delta}{\mathrm{d}s} &= -\frac{\partial \mathcal{H}^{(2)}_{\text{Drift}}}{\partial z} = 0 \\ \frac{\mathrm{d}z}{\mathrm{d}s} &= \frac{\partial \mathcal{H}^{(2)}_{\text{Drift}}}{\partial \delta} = \frac{1}{\beta_0^2 \gamma_0^2}. \end{aligned} \tag{2.19}$$

These linear equations can be analytically solved. For a drift space with path length L and the initial particle $\boldsymbol{x}(s_0)$, the phase space vector at the exit of a drift space $s_1 = s_0 + L$ is given by

$$\boldsymbol{x}(s_1) = \boldsymbol{R}_{\text{Drift}} \cdot \boldsymbol{x}(s_0), \tag{2.20}$$

where $\boldsymbol{R}$ is the *transfer matrix* of a drift space

$$\boldsymbol{R}_{\text{Drift}} = \begin{pmatrix} 1 & L & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & L & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{L}{\beta_0^2 \gamma_0^2} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.21}$$

Note that in this paraxial approximation, the normalized transverse momenta become

$$p_x = \frac{\mathrm{d}x}{\mathrm{d}s} = x', \tag{2.22}$$

which represents the geometric angle of the transverse motion of a particle.

Similar calculations can be performed for any given magnets. In general, the propagation of particles along an accelerator beam line can be written as

$$\boldsymbol{x}(s_1) = \boldsymbol{M}\left(\boldsymbol{x}(s_0)\right), \tag{2.23}$$

with $\boldsymbol{M}$ being a vector-valued function of the phase space coordinates. The transfer map can be expanded as described above

$$x_i(s_1) = \sum_j R_{ij} x_j(s_0) + \sum_{j,k} T_{ijk} x_j(s_0) x_k(s_0) + O(\boldsymbol{x}^3) \tag{2.24}$$

where R_{ij} denotes a component of the linear transfer matrix $\boldsymbol{R} \in \mathbb{R}^{6\times6}$ as described before and T_{ijk} is the component of a second-order transfer matrix $\boldsymbol{T} \in \mathbb{R}^{6\times6\times6}$. If only the linear order is considered, the transfer matrices can be grouped by matrix multiplication

$$\begin{aligned}\boldsymbol{x}(s_2) &= \boldsymbol{R}(s_1, s_2) \cdot \boldsymbol{R}(s_0, s_1) \cdot \boldsymbol{x}(s_0) \\ &= \boldsymbol{R}(s_0, s_2) \cdot \boldsymbol{x}(s_0),\end{aligned} \tag{2.25}$$

with the combined transfer matrix $R_{ik}(s_0, s_2) = \sum_j R_{ij}(s_1, s_2)R_{jk}(s_0, s_1)$.

In this dissertation, this idea of matrix based particle tracking is utilized to build a fast-executing differentiable simulation code *Cheetah* [42], which can be applied both to simulated optimization (see Section 4.3) and building fast RL training environments for online accelerator tuning (see Section 6.3).

2.1.4. Collective Effects

Until this point, all the calculations consider only the particles' motion under some external electromagnetic field, for example, provided by the magnets in an accelerator. Nevertheless, the motion of a particle bunch cannot be solely considered as a collection of individual particles, as the particles are charged and will interact with each other. Such effects are known as the *collective effects* [46]. As the dynamics of the particle beam is influenced by the charge density of the beam itself, collective effects cannot be modeled statically. One example is the *space charge* effect, coming from the electric fields of particles inside a bunch, which deteriorates the bunch profile quality and increases the beam sizes or beam emittances. For general bunch shapes other than uniform or Gaussian bunches, the forces from the space charge effects cannot be analytically solved. Simulating the impact of space charge usually involves discretizing the space, numerically solving the electric field at a given point, and applying the space charge forces as sequential kicks along the tracking. In general, these collective effects can only be approximated, and the accuracy is usually limited by computing resources and execution time. They become non-negligible and contribute largely to discrepancies between simulations and real-world conditions when accelerators operate in high bunch current and low energy regimes, which is common for linear accelerators.

Coherent synchrotron radiation (CSR) is another collective effect that is of particular interest. It is due to (part of) the electron bunch emitting coherently as a single particle. CSR is sometimes undesired as it may interact with the electron bunch again and degrade the beam quality, especially when the beam is fully compressed. However, this dissertation focuses on the beneficial aspect of CSR, which is using the emitted radiation as a source of intensive light.

2.2. Accelerator-based Radiation Generation

The instantaneous synchrotron radiation power emitted by a single electron moving on a circular trajectory with radius ρ is given by

$$I_\gamma = \frac{e^2 c}{6\pi\epsilon_0}\frac{\beta^4\gamma^4}{\rho^2}, \tag{2.26}$$

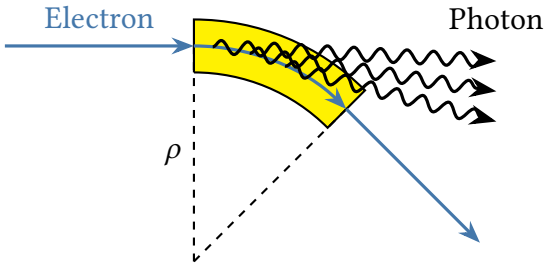


Figure 2.2.: Illustration of the synchrotron radiation emitted by an electron passing through a bending magnet.

where e is the unit charge, ϵ_0 is the vacuum dielectric constant [44]. Note that the radiation power scales with $I_\gamma \propto \gamma^4$ and quickly becomes intense for ultra-relativistic electrons.

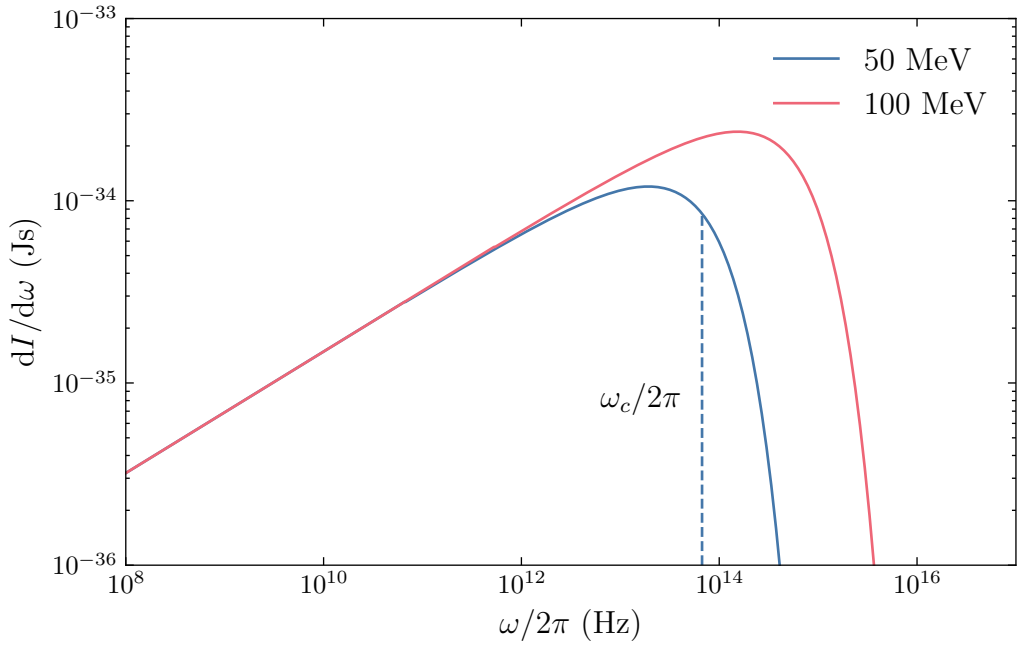


Figure 2.3.: Synchrotron radiation energy spectrum emitted by a single electron with different traveling in a circular trajectory with radius $\rho = 1\,\mathrm{m}$ according to Eq. (2.27). The vertical dashed line marks the critical frequency ω_c of the 50 MeV electron, dividing the spectrum into two equal parts of integrated intensities. The power spectra are spatially integrated over solid angle Ω.

The emitted radiation covers wide spectra with the spectral density being

$$\frac{\mathrm{d}^2 I_0}{\mathrm{d}\Omega \mathrm{d}\omega} = \frac{3e^2\gamma^2}{16\pi^2\epsilon_0 c}\left(\frac{\omega}{\omega_c}\right)^2 (1+\gamma^2\theta^2)^2 \left(K_{2/3}^2(\xi) + \frac{\gamma^2\theta^2}{1+\gamma^2\theta^2}K_{1/3}^2(\xi)\right) \tag{2.27}$$

where $\omega_c = 3\gamma^3 c/2\rho$ is the *critical frequency*, $\xi = \omega(1+\gamma^2\theta^2)^{3/2}\big/2\omega_c$, and K_ν are the modified Bessel functions. Fig. 2.3 illustrates the angle-integrated single electron synchrotron

radiation energy spectrum. The spectral intensity peaks at the order of ω_c, which divides the total spectrum into two halves of equal intensities when integrated over frequencies. For ultra-relativistic electrons $\gamma \gg 100$, the radiation is predominantly emitted in the forward cone with an opening angle $\theta_c \sim 1/\gamma$.

The total synchrotron radiation emitted by an electron bunch is

$$\frac{\mathrm{d}^2 I}{\mathrm{d}\Omega \mathrm{d}\omega} = [\underbrace{N_\mathrm{e}}_{\text{Incoherent}} + \underbrace{N_\mathrm{e}(N_\mathrm{e}-1)F(\omega)}_{\text{Coherent}}]\frac{\mathrm{d}^2 I_0}{\mathrm{d}\Omega \mathrm{d}\omega}, \tag{2.28}$$

where N_e is the number of electrons in a bunch, $F(\omega) \in [0, 1]$ is the form factor. The first part is the incoherent synchrotron radiation (ISR), scaling linearly with N_e. The second part is the CSR, which scales quadratically. Due to the large number of electrons in a bunch, the coherent radiation is enhanced by orders of magnitude when the form factor F is close to 1.

In the ultra-relativistic case, the form factor can be reduced to only the longitudinal component $F(\omega) \sim F_l(\omega)$. For a bunch with longitudinal Gaussian distributed charge density, the form factor can be analytically calculated

$$F_l(\omega) = \exp\left(-\omega^2\sigma^2\right), \tag{2.29}$$

where σ is the root mean square (RMS) bunch length. The calculation for arbitrary bunch distribution is further discussed in Section 4.2. This shows that the form factor exponentially decreases to zero for frequencies that are larger than the inverse bunch length.

Fig. 2.4 depicts the effect of CSR for longitudinal Gaussian bunches with 10 pC bunch charge and 50 MeV bunch energy. Note that the bunch length is expressed in σ_t (s) instead of σ_z (m). For relativistic particles, they are simply related by the speed of light

$$\sigma_t = \sigma_z / c. \tag{2.30}$$

The emitted intensity is clearly dominated by the coherent component from low-frequency to a cut-off frequency, which is inversely proportional to the bunch length $\omega/2\pi \sim 1/\sigma_t$. For shorter bunch lengths, the CSR spectrum extends to higher frequencies, and the overall radiation intensity is increased.

In addition to broadband CSR radiation, strong narrowband radiation can be generated with the help of undulators. They consist of alternating magnetic fields which deflect the electron bunch on an oscillating trajectory and the emitted radiation with the correct frequency component will be amplified by constructive interference.

Nowadays, many particle accelerators are built as dedicated *light sources* to utilize the synchrotron radiation effects. Storage rings have the advantage that they provide overall high radiation power due to the high repetition rate, and they can serve dozens of beamlines simultaneously. Linear accelerators are often limited in their repetition rates, commonly between 1 - 100 Hz, except for superconducting radio frequency (RF) guns, which can go up to the MHz range. However, they can generate ultra-short electron bunches and strong coherent radiation, emitting intensive light pulses down to attosecond ranges [47, 48]. They often have multiple operation modes and can provide tunable radiation pulses that are tailored to the experimental requirements.

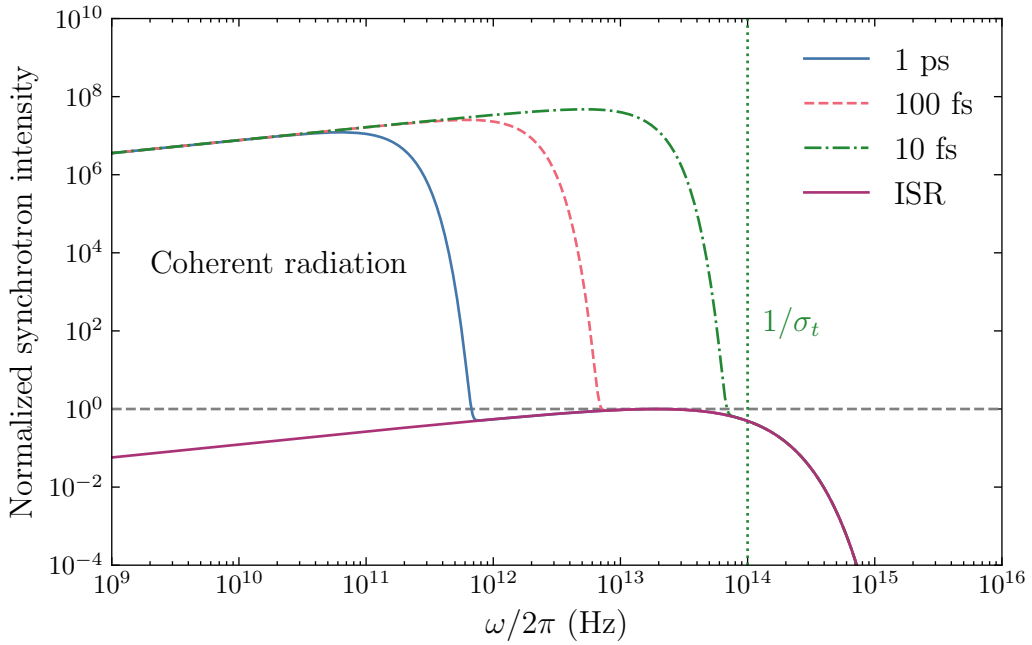


Figure 2.4.: Coherent synchrotron radiation spectrum for Gaussian bunches with different bunch lengths. The radiation intensities are normalized with respect to the maximum of the incoherent spectrum. All the bunches have 10 pC bunch charge and 50 MeV bunch energy.

2.3. Linear Particle Accelerators

The first component in an electron linear accelerator is the particle source. One common choice is the RF photo-injector, also known as the RF photocathode gun. The electrons are released via the photo-electric effect by laser pulses and metal or semiconductor cathodes. Pulsed electron bunches with high charge and low emittance can be generated in such RF photo-injectors. By changing the laser properties and the RF phase and amplitude, the initial bunch charge as well as the distribution can be tuned. The emitted electrons are rapidly accelerated to a few MeV by the RF cavities to reduce the space charge effect. In most cases, the strong defocusing effect and rapid emittance growth by repelling space charge effects [46] of the low energy electrons are further compensated by a solenoid magnet positioned after or even surrounding the RF gun [49].

The electrons are further accelerated in an array of periodic RF cavities, where the phase of the RF wave is synchronized with the bunches. The relativistic electrons pass through the cavities and get accelerated up to the desired energy. The high energy electrons can be directly utilized for experiments or radiation generation, for example using bending magnets or undulator magnets.

As shown above, very short electron bunch lengths are required to generate intensive radiation. In linear accelerators, such compression is performed using a *bunch compressor*. Fig. 2.5 shows the working principle of a standard bunch compressor consisting of four dipole magnets, also known as the *magnetic chicane*. The dipole magnets operate at the same magnetic strength, while the second and third magnets have an opposite sign than the

other two. The particles with different momenta will be deflected with different bending radii and therefore have different path lengths through the bunch compressor. A particle with a higher momentum than the reference particle p_0 will have a shorter path and move to the head of the bunch, and vice versa. As a result, the bunch length can be reduced if it is initially *chirped*. This means that the momentum offset δ and longitudinal offset z are correlated so that the high-energy particles are initially at the tail of the bunch, and low-energy particles at the head of the bunch.

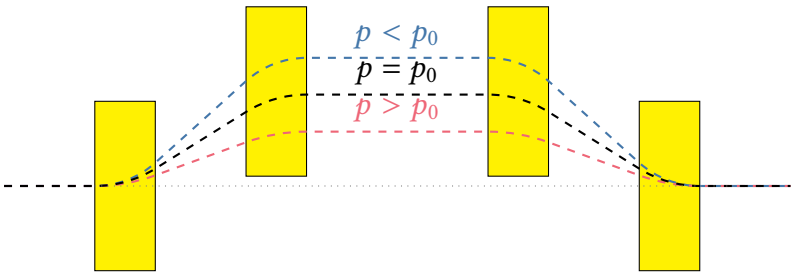


Figure 2.5.: Working principle of a bunch compressor consisting of four dipole magnets. The three dashed lines denote the trajectories of particles with momenta equal to the reference momentum p_0 (black), lower (blue), or higher (red) than p_0 respectively.

In linear accelerators, such longitudinal correlation can be created by an accelerating cavity. When passing through an ideal RF cavity with sinusoidal voltage, the particle with initial energy deviation $\delta(s_0)$ will have the final energy deviation

$$\delta(s_1) = \delta(s_0) + \frac{qV_{\mathrm{RF}}}{E_0} \sin\left(2\pi f_{\mathrm{RF}} \frac{z(s_0)}{c} + \phi_{\mathrm{RF}}\right), \tag{2.31}$$

where E_0 is the reference energy, V_{RF} is the voltage amplitude, f_{RF} is the RF frequency, and ϕ_{RF} is the phase offset. The chirp becomes more linear when the cavity is operated near zero-crossing $\phi_{\mathrm{RF}} = 0$, which inevitably reduces the energy gain of the particles in the cavity. While in small accelerators the main linac is often both used for accelerating and providing chirp, larger accelerators have additional dedicated cavities, providing more precise chirp and bunch length control.

2.4. Accelerators Studied in this Dissertation

2.4.1. FLUTE

The far-infrared linac and test experiment (FLUTE) [50] is an accelerator test facility at KIT to study the generation and diagnostics of ultra-short electron bunches and broadband THz radiations. It is designed to operate with a wide range of parameters, generating electron bunches with charges from pC to nC, bunch energy up to 90 MeV with an upgrade of RF systems (in 2024), and bunch lengths down to a few femtoseconds. The usual operation parameters of FLUTE and other accelerators are listed in Table 2.1.

Table 2.1.: Typical operation parameters of the studied accelerators FLUTE, ARES, and EuXFEL

Typical parameters	FLUTE	ARES	EuXFEL
Beam energy (MeV)	40 - 90	100 - 155	8 - 17.5 $\times 10^3$
Bunch charge (pC)	1 - 1000	3×10^{-3} - 100	20 - 1000
Bunch length (fs)	1 - 300	0.2 - 10	3 - 150
Repetition rate (Hz)	1 - 50	1 - 50	10

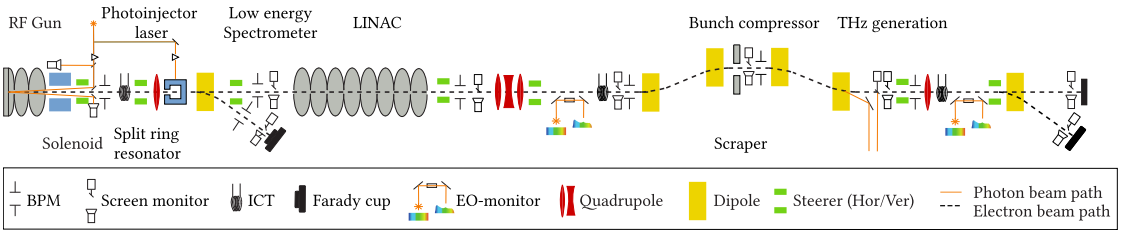


Figure 2.6.: Schematics overview of the FLUTE components. Figure adapted from [51].

Figure 2.6 shows the schematic layout of the FLUTE accelerator [51]. The electrons are generated at a copper cathode and accelerated up to 7 MeV by the RF gun. A solenoid magnet is positioned after the gun to focus the electron bunch. This section from the RF gun to the entrance of the linac is referred to as the *low energy section*. Afterward, a 3.15 m long linac accelerates the bunch to its final energy. In the simulation studies presented in this dissertation, the beam energy is limited to $\sim$ 50 MeV. Thanks to the latest RF system upgrade, the top energy can be increased to $\sim$ 90 MeV. The bunch is then longitudinally compressed using a bunch compressor consisting of four dipole magnets. At the last dipole, strong CSR will be emitted by the compressed bunch and further used for experiments.

2.4.2. ARES

The accelerator research experiment at SINBAD (ARES) [52] is an accelerator facility located at DESY to produce and study sub-femtosecond electron bunches, conduct accelerator components development, and provide electrons for medical experiments. Figure 2.7 shows a simplified schematic of the ARES layout. It shares a similar design as the FLUTE accelerator and therefore serves as an ideal test bed for investigating the transferability of the developed machine learning (ML) methods. The ARES contains a photo-injector and two independently driven traveling wave structures, providing electrons with bunch charges from 3 fC to 100 pC, energies up to 155 MeV, bunch length down to a few femtoseconds, and repetition rate from 1 - 50 Hz. In Section 6.3, the section named *Experimental Area (EA)* is investigated to compare the performance of various ML-based algorithms for online-tuning tasks.

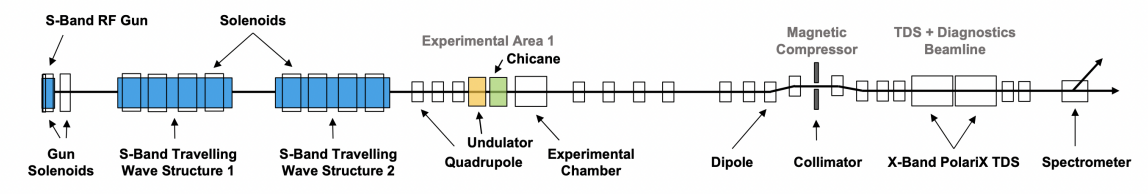


Figure 2.7.: Schematic layout of the ARES Accelerator. Figure from [53]

2.4.3. European XFEL

The European X-Ray Free-Electron Laser (EuXFEL) [6] is a hard X-ray free electron laser (FEL) in Hamburg. A simplified layout of the European X-Ray Free-Electron Laser (EuXFEL) is given in Fig. 2.8. The 1.7 km long superconducting linear accelerator can accelerate the electron bunches up to 17.5 GeV. It can accelerate a train of up to 2700 electron bunches with bunch charges from 20 pC to 1 nC and a maximal repetition rate of 10 Hz with 0.6 ms pulse duration, corresponding to a peak repetition rate of 4.5 MHz for the X-ray pulses. The high-energy electron bunches are subsequently guided into the beamlines equipped with arrays of undulators and emit intensive coherent radiations with the self-amplified spontaneous emission (SASE) mechanism, with photon energies ranging from 7 - 14 keV. In Section 6.1, Bayesian optimization is applied to maximizing the FEL intensity at the SASE1 beamline of the European XFEL.

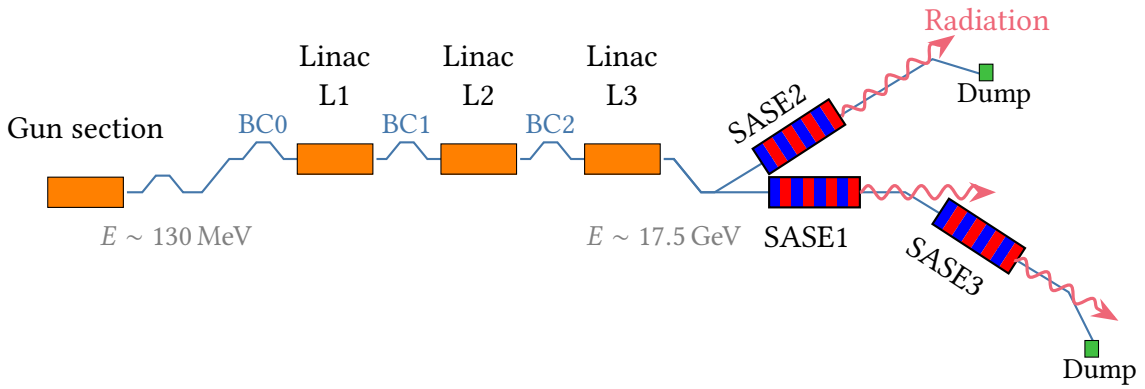


Figure 2.8.: Simplified layout of the EuXFEL. The blue line denotes the electron path. The bunch goes through a series of accelerating structures and bunch compressors. It consists of three beamlines with undulators that can generate intense coherent radiation using the SASE mechanism. Figure adapted from [54].

3. Machine Learning Methods

This chapter aims to provide some basic concepts in machine learning (ML), which are required to follow the discussions in this dissertation. It starts with an introduction to the general concepts neural network (NN) and gradient-based optimizations. The chapter focuses then on Bayesian optimization (BO) and reinforcement learning (RL), two methods for online accelerator tuning that are investigated in detail in this dissertation. A brief overview of the successful applications of BO and RL at particle accelerators is provided as well. For more mathematical treatments, readers are referred to the textbooks for the individual textbooks respectively [55, 56, 57].

3.1. Neural Networks

The neural network is one of the most fundamental building blocks in modern machine learning methods. They are universal function approximators that describe a general nonlinear function

$$f(\boldsymbol{x}|\boldsymbol{w}) = \boldsymbol{y}, \tag{3.1}$$

with parameters $\boldsymbol{w}$, which maps the vector-valued inputs $\boldsymbol{x}$ to outputs $\boldsymbol{y}$.

Fig. 3.1 illustrates the structure of a neural network in its most simple form, consisting of only one layer with a single neuron, also known as the perceptron. It takes an n-dimensional input $(x_1, \ldots, x_n)$ and transforms it by a weighted sum $(w_1^{(1)}, \ldots, w_n^{(1)})$ and a bias term b

$$t^{(1)} = \sum_{i=1}^{n} w_i^{(1)} x_i + b. \tag{3.2}$$

The superscripts here denote the number of layers. For the sake of simplicity, the bias can be absorbed into the weight vector $w_0 \coloneqq b$ and the input vector $\boldsymbol{x}$ is augmented with an additional $x_0 = 1$, so that the weighted sum becomes simply

$$t^{(1)} = \boldsymbol{w}^{(1)\top} \boldsymbol{x}. \tag{3.3}$$

The transformed value is further processed by a nonlinear *activation function* $g(\cdot)$ to generate the output

$$a^{(1)} = g(\boldsymbol{w}^{(1)\top} \boldsymbol{x}). \tag{3.4}$$

In a feed-forward fully connected neural network (NN), also known as the multilayer perceptron (MLP), the neurons are aligned in layers. The output of j-th layer $\boldsymbol{a}^{(j)}$ is passed as the input of the next layer $j + 1$, until reaching the output layer. This process is called the *forward propagation*. When NN is trained and used to make predictions, it is often referred to as *inference*. By addressing the propagation as tensor multiplication and

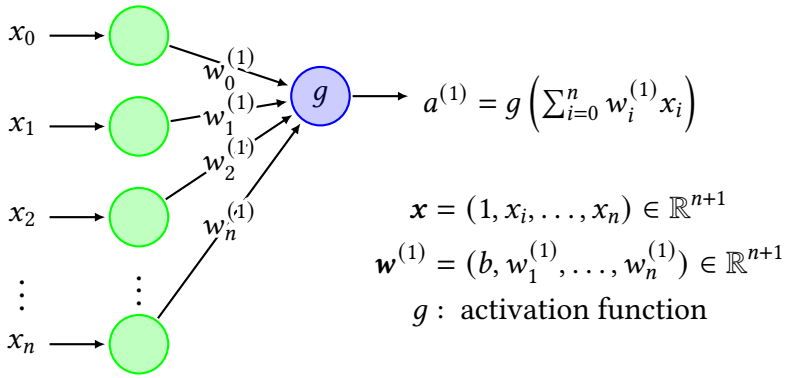


Figure 3.1.: Structure of a one-layer neural network, also known as the *perceptron*. The input $\boldsymbol{x}$ is multiplied with $\boldsymbol{w}$, which includes the weights $(w_1, \ldots, w_n)$ augmented by a scalar-valued bias b. The sum is transformed by a nonlinear activation function g to produce the output $a^{(1)}$.

utilizing specialized processors like graphical processing units (GPUs), the calculation can be highly parallelized and the inference with NN becomes very efficient.

Within one layer, usually, the same activation functions are used for the sake of parallelism. The nonlinear activation functions are essential in the neural network for its approximation capabilities. Without them, the represented function mapping becomes linear and can be reduced to a simple matrix multiplication.

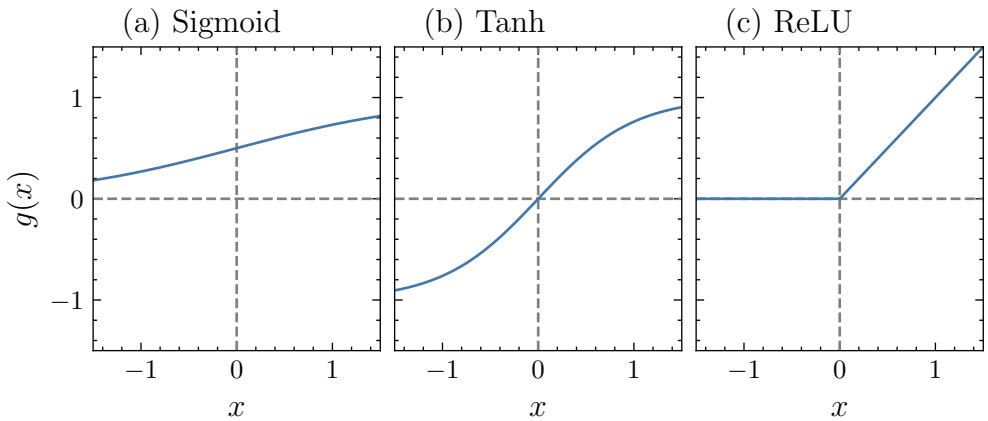


Figure 3.2.: Commonly used activation functions in neural networks, including (a) sigmoid, (b) hyperbolic tangent, and (c) ReLU function.

Various activation functions are designed for different purposes and applications. They are often composites of simple functions where the derivatives can be analytically calcu-

lated. Some commonly used activation functions are introduced below and visualized in Fig. 3.2.

$$\begin{aligned} g_{\text{sigmoid}}(x) &= \frac{1}{1+e^{-x}}, \\ g'_{\text{sigmoid}}(x) &= g(x)(1-g(x)) \end{aligned} \tag{3.5}$$

The sigmoid function is smooth and constrained in $[0, 1]$, which makes it ideal for binary classification tasks. Another popular choice of activation function is the hyperbolic tangent function, which is constrained in the range of $[-1, 1]$.

$$\begin{aligned} g_{\tanh}(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}}, \\ g'_{\tanh}(x) &= 1 - g(x)^2 \end{aligned} \tag{3.6}$$

In comparison to sigmoid, tanh has the advantage that it is zero-centered and the derivative is four times larger than the one of sigmoid at $x = 0$, allowing the training process to be faster. Nevertheless, it still saturates at large input amplitudes and the gradient could vanish for deep neural networks. Rectified linear unit (ReLU) is a widely used non-bounded activation function

$$\begin{aligned} g_{\text{ReLU}}(x) &= \max(0, x), \\ g'_{\text{ReLU}}(x) &= \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases} \end{aligned} \tag{3.7}$$

As it does not contain exponential terms, ReLU is much simpler to implement in hardware and requires considerably lower computation resources. Although it can struggle to predict highly nonlinear features when using a small-sized NN, this is not a problem in larger NNs. One downside of ReLU is its flat region for negative input values, which can potentially stop the progress in NN training. This can be mitigated by several variants, including Leaky ReLU $g(x) = \alpha x$ for $x < 0$, exponential linear unit (ELU) $g(x) = \alpha(e^x - 1)$ with a free parameter α controlling the behavior for $x < 0$, and Softplus $\frac{1}{\beta}\ln(1 + \exp(\beta x))$ with a parameter β which constrains the output to be always positive.

It is favorable to scale the inputs and outputs of the neural networks according to the active regions of the activation function to fully utilize their nonlinearity. Common practices are min-max normalizing the input to $[0, 1]$ or standardizing the input data with zero mean and unit standard deviation.

3.1.1. Automatic Differentiation and Gradient Descent Optimization

The network weights $\boldsymbol{w}$ are usually randomly initialized and need to be adapted so that they can approximate the behavior of an unknown mapping. This process is known as the *training* of NNs. It becomes a standard optimization problem with a dataset of training inputs X and target outputs $\hat{Y}$

$$\min_{\boldsymbol{w}} \sum_{\boldsymbol{x} \in X,\, \hat{\boldsymbol{y}} \in Y} L(\boldsymbol{y} = f(\boldsymbol{x}|\boldsymbol{w}), \hat{\boldsymbol{y}}), \tag{3.8}$$

where $\hat{\boldsymbol{y}}$ denotes the target output, also known as the training *labels*, and L is a custom scalar-valued *loss function*, quantifying the discrepancy between the NN prediction and the

target output. Commonly used loss functions are the mean squared error (MSE) and mean aboslute error (MAE) for regression tasks, and the cross-entropy function for classification tasks. Almost any numerical optimization algorithms can be used to fit the NN parameters, such as particle swarm optimization, genetic algorithm (GA), simulated annealing, and L-BFGS [58]. Nevertheless, due to the large number of parameters to be optimized and the fact that the computations in the network are composites of elementary functions, gradient descent algorithms are very effective and have become de facto the standard approach when it comes to neural network training.

In many cases, the stochastic gradient descent (SGD) is used for optimizing the parameters, which means that instead of the gradient of the total loss in the dataset, only the gradient with respect to a single data pair or a *mini-batch* of data is calculated. This increases the frequency of gradient updates and accelerates the overall training process. In each step, the parameters $\boldsymbol{w}$ are updated using the gradient values

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}} L(\boldsymbol{x}, \hat{\boldsymbol{y}}, \boldsymbol{w}), \tag{3.9}$$

where α is the *learning rate* that scales the amplitude of each update and $\nabla_{\boldsymbol{w}}$ denotes the partial derivative of the loss function with respect to the parameters $\boldsymbol{w}$.

Modern machine learning software libraries, such as PyTorch [59, 60], TensorFlow [61], and JAX [62] support a feature called automatic differentiation (AD), which allows the derivatives of a function to be automatically obtained. When training NNs, a computation graph is built to keep track of the transformations being applied to obtain the output during the forward propagation, i.e. passing the input training data to obtain the output values. Afterward, the partial derivative of the loss function with respect to an individual weight parameter can be calculated based on the gradient of the next layer by applying the chain rule iteratively. As the gradient flows backward through the network, this step is also known as the *backpropagation*. This backward propagation can also be repeated to obtain Hessians if higher-order derivative information is required for the gradient-based algorithms.

Multiple variants of the basic SGD algorithm are designed to improve its speed and convergence properties. For example the introduction of a momentum term to accelerate the convergence when gradients are in the same direction over multiple update steps, the adaptive learning rate based on past gradients in RMSprop, and the Adam optimizer which combines both ideas [63].

Moreover, the usage of gradient descent methods is not only limited to neural network training but more generally in scenarios where the gradient information can be readily accessed, such as when simulation tools are built with ML libraries. In Section 4.3, *Cheetah* is introduced as a novel differentiable accelerator simulation code built with PyTorch, fully utilizing the advantages of AD with modern gradient descent algorithms.

When using neural networks to approximate the function mappings, the final performance not only depends on the optimized weights but also on the set of *hyperparameters*. They are additional parameters that are usually fixed before the training process and govern the overall behavior of the algorithm. Hyperparameters can be numerical values like the number of neurons and number of layers in a NN, size of the mini-batches, initial learning rate, and the number of epochs, i.e. the number of complete passes through the

training dataset. They could also be categorical values such as choice of activation function and optimizer algorithms.

3.2. Bayesian Optimization

While neural networks are powerful tools that can approximate arbitrarily complex functions, they require a large number of high-quality training samples. Such amount of data is often not available, especially in real-world accelerators, where beam time is scarce.

Bayesian optimization (BO) is a class of numerical optimization algorithms that can globally optimize an unknown objective function efficiently. Recently it has been widely applied in the particle accelerator community for both online and simulated optimization tasks. This section introduces the basic concepts of BO algorithms used in this dissertation, more details on the mathematical formulation can be found in the textbook [56]. An introduction to BO as well as its application cases for particle accelerators are also provided in the recent review paper [64].

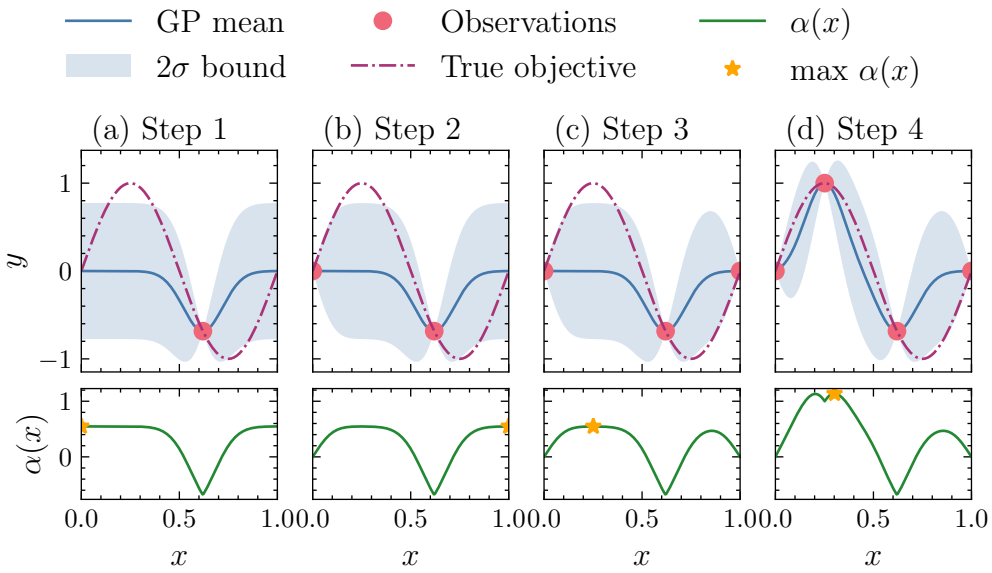


Figure 3.3.: Visualization of Bayesian optimization steps. In each step, a Gaussian process (GP) is trained on previous observations (red dots) as a statistical model for an unknown objective function (red dash-dot line). The GP model predicts the function value (blue line) with uncertainty (blue shaded region). Based on the GP posterior predictions, an acquisition function (green line) is calculated and the next point to sample (yellow star) is chosen by maximizing the acquisition function.

Here, a general optimization problem is considered

$$\max_{\boldsymbol{x}\in X} f(\boldsymbol{x}), \tag{3.10}$$

where an unknown black-box objective f is to be maximized within an input domain $\mathcal{X} \subseteq \mathbb{R}^d$. This means that the objective can only be evaluated but the analytical form of it is unknown, and no gradient information is available. Although not considered here, constraints can be included so that the domain of possible input parameters is further limited.

At each step, the evaluation of the objective will return a noisy signal

$$y = f(\boldsymbol{x}) + \epsilon, \tag{3.11}$$

with a Gaussian noise following an independent, identically distributed Gaussian distribution $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. In most cases, this noise is assumed to be homoscedastic, which means that it does not depend on the input parameters $\boldsymbol{x}$. Additionally, the function evaluation is assumed to be expensive, which can be due to high computational or monetary costs. The goal is to find the optimal set of parameters with the least amount of steps.

The working principle of BO is illustrated in Fig. 3.3 for a one-dimensional optimization task. In each step, a surrogate model of the objective is built using the observation data. While the surrogate model can have arbitrary structure, in most cases a GP model [56] is used. Based on the predictions of the surrogate model, an acquisition function $\alpha(\cdot)$ can be defined to determine the parameter setting to sample. Then, the function is evaluated at the setting that maximizes the acquisition, and the observations are added back to the dataset. These steps are repeated until the optimization criteria are met.

3.2.1. Gaussian Process Modeling

GP is a non-parametric way to model unknown functions. It is used in BO as a sample efficient probabilistic surrogate of the objective function, providing both the estimate of objective value at a given point and its uncertainty. A GP describes a distribution of possible functions

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')), \tag{3.12}$$

with the *prior mean* function $m(\cdot)$ and *covariance* (or *kernel*) function $k(\cdot, \cdot)$ defined as

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}\left[f(\boldsymbol{x})\right] \\ k(\boldsymbol{x}, \boldsymbol{x}')) &= \mathbb{E}\left[(f(\boldsymbol{x}) - m(\boldsymbol{x})(f(\boldsymbol{x}') - m(\boldsymbol{x}')\right]. \end{aligned} \tag{3.13}$$

For convenience of calculation, or when the landscape of the objective function is unknown, the prior mean is often set to be $m(\boldsymbol{x}) \equiv 0$. However, this is not always the case, as discussed later in Section 4.4.

At each input position $\boldsymbol{x}^*$, the function value is a Gaussian distributed random variable $p(y|\boldsymbol{x}^*)$, as visualized in Fig. 3.4(a). Samples of functions can be drawn from the GP by evaluating at a set of input points X^*, where the function values jointly form a multivariate Gaussian distribution. After observing n data points $D = \{X, Y\}$, the model can make *posterior* predictions $p(y|\boldsymbol{x}^*, D)$ conditioned on the available data, which are still Gaussian distributed. In general, the posterior distribution for n^* test points X^* in the noise-free case $\sigma_\epsilon = 0$ is given by $\mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^{*2})$, where

$$\begin{aligned} \boldsymbol{\mu}^* &= K(X^*, X)K(X, X)^{-1}Y, \\ \boldsymbol{\sigma}^{*2} &= K(X^*, X^*) - K(X^*, X)K(X, X)^{-1}K(X, X^*). \end{aligned} \tag{3.14}$$

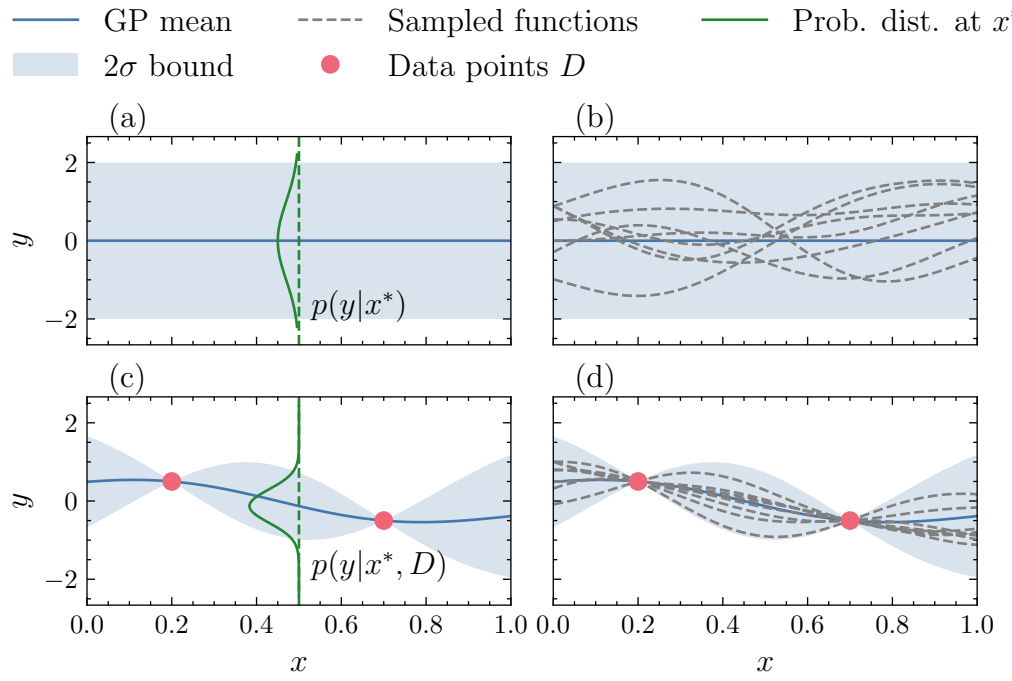


Figure 3.4.: GP prior and posterior visualizations. (a) The prior distribution of GP with zero prior mean. The function value at an arbitrary point is distributed according to a Gaussian distribution. (b) Function samples drawn from the prior distribution. (c) Posterior predictions of GP can be calculated by conditioning on the observed data points D. (d) Function samples drawn from the posterior distribution.

$K(X, X)$ is an $n \times n$ covariance matrix between the dataset, $K(X^*, X)$ is an $n^* \times n$ covariance matrix between the test points and the dataset, and $K(X^*, X^*)$ is an $n^* \times n^*$ covariance matrix between the test points. Similarly, as for the prior distribution, functions can also be sampled from the posterior distribution as shown in Fig. 3.4(d).

The behavior of GP model is governed by the choice of the covariance function, which quantifies the similarity between two points. Based on the assumption that the objective f is continuous, the data points that are close to another point $\boldsymbol{x}$ are expected to have similar output values to $f(\boldsymbol{x})$. In many cases, stationary kernels are used, which means that the correlation only depends on the relative displacement between two points $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x} - \boldsymbol{x}')$, and not on their absolute positions.

One of the most commonly used covariance functions is the radial basis function (RBF), resembling the form of a Gaussian distribution. The RBF is defined as

$$k_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2}\frac{||x - x'||^2}{\ell^2}\right), \tag{3.15}$$

where ℓ is the lengthscale hyperparameter. More generally, the objective can have different sensitivity to individual input dimensions, leading to a vector of lengthscales $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_d)$

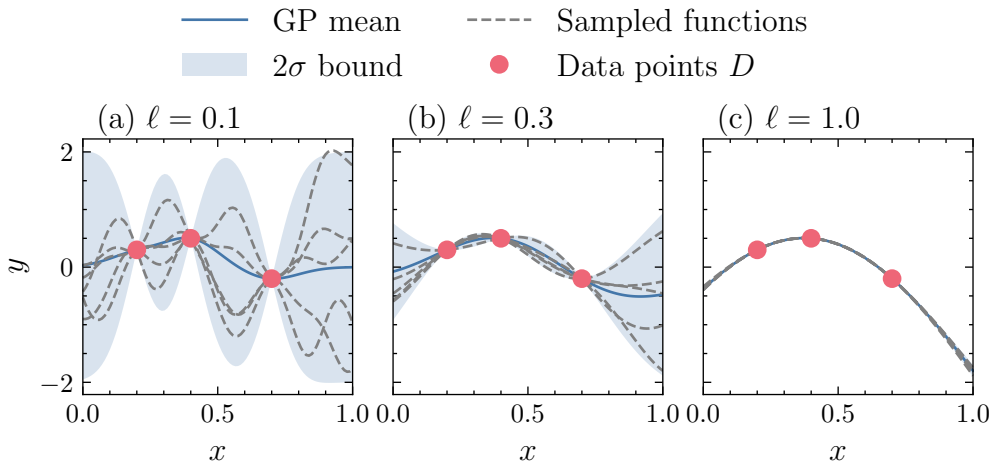


Figure 3.5.: Effects of the lengthscale hyperparameter on the GP model.

with distinct values, also known as the *automatic relevance determination*. The RBF can therefore be extended to

$$\begin{aligned} k_{\text{RBF, ARD}}(\boldsymbol{x}, \boldsymbol{x}') &= \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^\top \operatorname{diag}(\boldsymbol{\ell})^{-2}(\boldsymbol{x}-\boldsymbol{x}')\right) \\ &= \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\left(\frac{x_i - x_i'}{\ell_i}\right)^2\right), \end{aligned} \tag{3.16}$$

where $\operatorname{diag}(\boldsymbol{\ell})$ denotes a $d \times d$ diagonal matrix with $\boldsymbol{\ell}$ as entries. Each ℓ_i roughly corresponds to the distance along one input axis, at which the two data points become uncorrelated, and the function values can change significantly. The effect of the lengthscale hyperparameter is visualized in Fig. 3.5. For small values of ℓ, the function varies rapidly over a short interval. When the lengthscale becomes larger, the function becomes smoother and varies slowly over a long range.

The RBF kernel makes smooth predictions, i.e. the represented functions are indefinitely mean-square differentiable. In reality, however, this is often too stringent of a constraint on the modeled physical objectives. A generalization of RBF is the Matérn kernel. It is defined as

$$k_{\text{Matérn}}(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\frac{||\boldsymbol{x}-\boldsymbol{x}'||}{l}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{||\boldsymbol{x}-\boldsymbol{x}'||}{l}\right), \tag{3.17}$$

with positive hyperparameters ν and ℓ, and K_ν is the modified Bessel function. The values for ν are commonly chosen to be half-integer, e.g. $\nu = 1.5$ leads to once mean-square differentiable functions, and $\nu = 2.5$ leads to twice mean-square differentiable functions.

The complete covariance function can also be a composite of multiple basic kernels by multiplication or addition. In particular, the covariance function used in GP is often expressed as

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 k_0(\boldsymbol{x}_p, \boldsymbol{x}_q) + \sigma_{\text{n}}^2 \delta_{pq}, \tag{3.18}$$

where k_0 is a basic kernel function like RBF and Matérn, $\boldsymbol{x}_p$ and $\boldsymbol{x}_q$ are two arbitrary points, and δ_{pq} is a Kronecker delta ($\delta_{pq} = 1 \Leftrightarrow p = q$, $\delta_{pq} = 0$ otherwise). It contains further

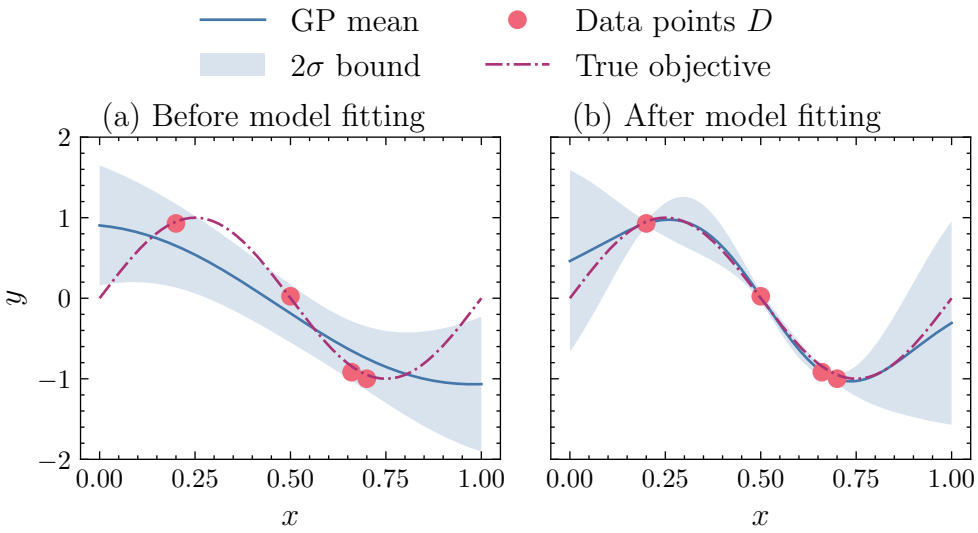


Figure 3.6.: Maximum log-likelihood fitting of the GP hyperparameters

hyperparameters that can be varied, the signal variance σ_f^2 and the noise variance σ_n^2. The diagonal term containing the noise variance is used to model the independent white noise in the measurement. The signal variance σ_f^2 scales the absolute amplitude of the covariance function.

The hyperparameters of the GP model need to be adapted so that the resulting model can best represent the observed data points (X, Y). The marginal likelihood of the model over the latent function values $\boldsymbol{f}$ is

$$p(Y|X) = \int \underbrace{p(Y|\boldsymbol{f}, X)}_{\text{likelihood}} \underbrace{p(\boldsymbol{f}|X)}_{\text{prior}} \mathrm{d}\boldsymbol{f}, \tag{3.19}$$

which equals the integration of the likelihood times the prior integrated over the parameter space of $\boldsymbol{f}$. Under the GP assumption, the prior $p(\boldsymbol{f}|X)$ is Gaussian, and its log-likelihood function can be calculated analytically. The marginal log likelihood (MLL) of the GP model conditioned on the hyperparameters $\boldsymbol{\theta}$ with n data points (X, Y) can be expressed as

$$\log p(Y|X, \boldsymbol{\theta}) = -\frac{1}{2} Y^T K_Y^{-1} Y - \frac{1}{2} \log |K_Y| - \frac{n}{2} \log 2\pi, \tag{3.20}$$

where $K_Y = K(X, X) + \sigma_n^2 I$ is the covariance matrix for the noisy target Y, the model hyperparameters $\boldsymbol{\theta}$ are contained in K_y, and n is the number of training samples $n = \text{len}(X)$. The first term with the training data quantifies the goodness of the data fit, the second term contains only the model parameters and acts as a penalization for model complexity, and the third term is a normalization constant. Taking the lengthscales ℓ as an example, smaller values of $\boldsymbol{\ell}$ lead to a more flexible model but a higher penalty in the model complexity. Therefore, maximizing the MLL results in a GP model that sufficiently represents the training data and contains the simplest model. One way to determine the GP hyperparameters is to estimate them in advance using historical data [31], but this

limits the model's ability to adapt to new conditions that are not described by the historical data. Alternatively, the GP model can be re-trained on the dataset at every step during optimization, at the cost of increasing computational requirements during application time. The maximization of the MLL typically takes 0.1 - 10 seconds on a conventional CPU, but the time scales $O(n^3)$ with the number of training points.

3.2.2. Acquisition Function

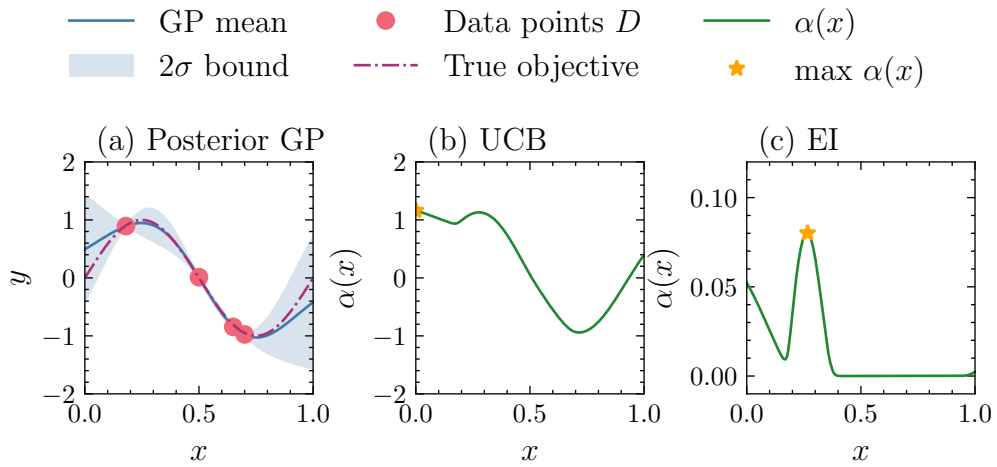


Figure 3.7.: Acquisition functions are constructed using the GP posterior predictions. (a) shows the GP model with the observed data. (b) shows the upper confidence bound (UCB) and (c) shows the expected improvement (EI) acquisition function.

With a GP model inferring the posterior distribution of the objective function, an acquisition function α is built to determine the next sample point to evaluate, so that the number of required physical observations is effectively reduced. Concretely, the objective f is sampled at $\arg\max_{\boldsymbol{x}} \alpha(\boldsymbol{x}|D)$ at each step. This is visualized in Fig. 3.7. Custom acquisition functions can be constructed to meet specific task requirements. In general, to find the global optimum of the unknown objective, they all balance the exploration and exploitation of the search space. This can be explained in the following by the two commonly used acquisition functions: the upper confidence bound (UCB) and the expected improvement (EI). UCB explicitly controls the exploration-exploitation trade-off with a parameter β_{UCB}

$$\alpha_{\text{UCB}}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \beta_{\text{UCB}}\sigma(\boldsymbol{x}), \tag{3.21}$$

where $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are the GP posterior mean and standard deviation. For high β_{UCB} values, the contribution of the variance term becomes large and points with high uncertainty are sampled, which leads to more exploration. On the contrary, small β_{UCB} values lead to more exploitation of observed peaks, as they have a higher posterior mean. An empirical choice of the parameter is $\beta_{\text{UCB}} = 2$, which corresponds to the usual 95% confidence bound for Gaussian distributed values. Nevertheless, β_{UCB} can also increase along with the evaluation steps to ensure that BO converges to the global optimum [65, 66].

The EI calculates the expected value of the improvement of a proposed point $\boldsymbol{x}$ over the best-observed value f_{best} [67]

$$\begin{aligned} \alpha_{\text{EI}}(\boldsymbol{x}) &= \mathbb{E}[\max(\mu(\boldsymbol{x}) - (f_{\text{best}} + \xi), 0)] \\ &= (\mu(x) - (f_{\text{best}} + \xi))\Phi(Z) + \sigma(\boldsymbol{x})\phi(z) \end{aligned} \tag{3.22}$$

with the parameter z describing the normalized improvement

$$z = \frac{\mu(\boldsymbol{x}) - (f_{\text{best}} + \xi)}{\sigma(\boldsymbol{x})}, \tag{3.23}$$

where ϕ is the probability density function (PDF) and Φ is the cumulative density function (CDF) of the standard normal distribution. The exploration-exploitation trade-off in EI can be controlled by introducing a positive parameter ξ [68]. In general, a large ξ value acts as if the previous best value is higher, steering the BO to perform more exploration.

For many optimization tasks, UCB and EI both behave robustly and lead to comparable performance. However, numerical instabilities can arise when optimizing the EI acquisition when a large region of the parameter space has acquisition values that are close to zero. This is also reflected in Fig. 3.7(c), where the EI is very flat for $x \in [0.5, 1.0]$.

3.2.3. Tailoring Bayesian Optimization Methods for Particle Accelerators

Although the vanilla version BO algorithm can already handle a wide range of optimization tasks, adapting individual components of the algorithms to the individual tasks will further increase its performance and allow it to meet certain real-world requirements. Specifically in the domain of particle accelerator tuning, the beamtime for tuning needs to be minimized, certain safety constraints must be respected, and prior knowledge of the accelerator system can be included. Below, some customizations of the BO algorithm with their application for the accelerator are introduced. A more detailed treatment can be found in [64].

The modifications of BO can be mainly separated into the GP modeling techniques and the design of acquisition functions. Due to the specific layout of each accelerator, settings like magnet strengths are often grouped in so-called families and changed simultaneously during operation, such correlation information can be input into the kernel function. For example in the free electron laser (FEL) intensity tuning task [28] and beam loss minimization task in a storage ring [69], the correlations between quadrupole strengths are calculated in the form of Hessian matrices and used to speed up the BO. In addition, the objective function is not always a static function of the tuning parameters like magnet strengths. In reality, they could depend on contextual variables like the bunch current or vary in time due to systematic drifts. Such effects can be explicitly modeled as extra dimensions in the GP models, also known as adaptive BO (ABO) [70] or contextual BO (C-BO) [71] respectively. C-BO has been applied to optimize the injection rate into the storage ring Karlsruhe research accelerator (KARA), keeping track of the optimum setting varying over the amount of accumulated current [31]. ABO has been applied to beam trajectory tuning in the APS linac for an upstream beam that is drifting over time [72, 73]. Another important feature of the accelerators is that data can be obtained at different fidelities. In simulation, this means that multiple simulation models are available with

different physical effects included. In online tuning, this means that beam measurements with different diagnostic devices or data acquisition configurations. Generally, such a multi-fidelity setup involves a trade-off of spending more time to obtain more accurate data, or saving resources but obtaining only low-quality data. The data coming from different sources can be combined and explicitly modeled in the GP model with an extra fidelity parameter. This has been demonstrated in the design optimization in laser-plasma accelerator simulations [74, 16]. Lastly, non-zero prior mean functions can be used in the GP modeling instead of the uninformative zero prior mean. This is further illustrated in Section 4.4 for the simulation of FLUTE.

The acquisition function can be tailored to change the behavior of BO for accelerator tasks. One example is to use only the GP posterior uncertainty as acquisition $\alpha(x) = \sigma(x)$ to only focus on the exploration of the parameter space. This has been applied for efficient online characterization in accelerator experiments [75]. For accelerator tuning, smaller step sizes in parameter changes are often preferred to maintain an overall stable system and reduce the settling time of these parameters via individual feedback controllers. This can be achieved by setting a hard limit on the step size or by biasing the acquisition function towards the current setting [75, 76]. Accelerators often have safety constraints that should not be violated during the optimization process. The constraints can be explicitly predicted using GP models in the SafeOpt algorithms so that the optimization is only performed in the safe region. In the context of accelerator tuning, this has been demonstrated by FEL tuning while maintaining a threshold radiation intensity [77], beam loss optimization while not triggering any interlock systems [30], and timing jitter minimization while maintaining the laser lock in the laser synchronization system [78]. In the multi-fidelity case, the acquisition function also needs to be modified. This has been demonstrated in the laser-plasma accelerator simulations, where a low-fidelity fast-executing simulation is used to support the optimization for a computationally intensive high-fidelity simulation [16]. The acquisition function can also be extended to solve the multiobjective optimization task, finding a Pareto-front of non-dominated points with competing objectives. This has been demonstrated in the photo-injector tuning to find the beam with minimal beam emittances, beam sizes, and energy spread [79]. Another example is to find the maximum bunch charge with minimal energy spread in a laser-plasma accelerator [80].

3.3. Introduction to Reinforcement Learning

Reinforcement learning (RL) is a machine learning (ML) paradigm, where the algorithm learns to make decisions to maximize a reward signal. The learner, referred to as the *agent*, can make decisions and interact with an *environment*. Similar to a numerical optimization algorithm like BO, it also faces the exploration-exploitation trade-off. The agent needs to focus on previous effective actions to gain more rewards but also needs to try out new actions in the hope of more future rewards. In the last decade, reinforcement learning (RL) has been successfully applied to a wide range of tasks, from complex games like Atari [81], chess, and go [82, 83], to real-world control tasks such as tokamak fusion control [84] and drone racing [85].

This section gives a brief introduction to RL, providing the basic terminologies that are required to understand the work in this dissertation. More details on RL can be found in [57], where most of the treatment below is based on.

3.3.1. Reinforcement Learning Problem Formulation

The successful application of RL depends heavily on the formulation of the task. In general, the RL problems are formulated as Markov decision processs (MDPs). A MDP is a formalization of sequential decision-making, where the actions can also have a long-term influence on the subsequent situations and therefore the future rewards. The interactions take place in discrete time steps $t = 0, 1, 2, \ldots$. At each time step t, the agent receives a representation of the environment's current state $S_t \in \mathcal{S}$. Based on this observation, the agent takes an action $A_t \in \mathcal{A}$. Note that in general, the possible action space can be dependent on the specific state $\mathcal{A} = \mathcal{A}(s)$. The environment enters into the next state S_{t+1} and the agent receives a scalar-valued reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$. The sequence of such interactions is called a *trajectory*, which looks like

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots \tag{3.24}$$

When the agent takes a certain action, the next environment state does not need to be deterministic. It can be generally described with a probability function

$$\begin{aligned} &p : (\mathcal{S} \times \mathcal{R}) \times (\mathcal{S} \times \mathcal{A}) \rightarrow [0, 1] \\ &p\left(s', r | s, a\right) = \Pr\left\{S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\right\}, \end{aligned} \tag{3.25}$$

where s', s, r, a are specific instances of the states S, rewards R, and actions A. This denotes the conditional probability of the state going into a new state s' and returning the reward r if the agent takes action a at a system state s. The transition function p is also called the *dynamics* of the MDP, with

$$\sum_{s'} \sum_{r} p\left(s', r | s, a\right) = 1, \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \tag{3.26}$$

This means that the possible next states S_{t+1} and rewards R_{t+1} depend only on the immediate preceding state S_t and action A_t, but not on the history, i.e. earlier states and actions. This is called the *Markov property*. In the context of RL, this implies that the agent should be able to make a decision based only on the current information of the environment. At any given time step t, the agent is then tasked to maximize the cumulative rewards that it receives in the future. This is also called the *return*

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T, \tag{3.27}$$

where T is the time step for a terminal state. This sequence of interactions is also called an *episode*. In the real world, the task can also be continuous and does not have a specific terminal state. In such a case, the return might become infinite. A discount factor $\gamma \in [0, 1]$

can be introduced to mitigate the infinity and simplify the mathematical derivations. The return in Eq. (3.27) can be generalized to the discounted return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{3.28}$$

Here, the γ effectively determines the horizon the RL agent looks into the future. For $\gamma = 0$, the RL agent only looks at the immediate return, behaving like a numerical optimizer. For γ close to 1, the delayed reward also becomes relevant, and the agent can learn to trade off immediate rewards for higher future rewards, planning far ahead in time steps. Overall, a MDP can be defined using a five-tuple

$$(S, A, R, p, \gamma), \tag{3.29}$$

that contains the state, action, reward, dynamics, and the discount factor.

Nevertheless, such an idealized formalism may not be always directly applicable to a real-world task. One issue is that the agent sometimes cannot fully observe the environment's state. Such a problem is called the partially observable Markov decision process (POMDP). In each time step, the agent only receives an *observation* $O_t \in \mathcal{O}$, instead of the underlying state S_t. For example, this can be due to the limited diagnostics available in the real-world system. The agent needs to infer the latent variables and try to make the optimal decision based on the partial observations. In this case, the Markov property can still be restored in the agent's view if certain histories are included in the state definition, that are sufficient for the agent to infer the latent states. The consideration of partially observable environments becomes essential when training RL agents for real-world particle accelerator tuning, as will be discussed in Section 6.3. For simplicity reasons, the following treatment will still assume that the state is fully observable $O_t = S_t$.

3.3.2. Policy and Value Functions

The way that the RL agent behaves is called the *policy*. Formally, it is defined as the probability distribution $\pi(a|s), \forall a \in \mathcal{A}, s \in \mathcal{S}$, i.e. how likely the agent will select an action a when being in the state s. RL algorithms specify how the policy π should be changed based on the agent's experience to optimize the return.

Most of the RL algorithms make use of value functions to quantify how valuable a given state is, or how valuable it is for the agent to perform a certain action in that state. As the future rewards depend on the agent's behavior, such value functions also depend on the policy π that the agent takes. Specifically, the *state-value* function for an agent following a policy π is defined by

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t | S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \quad \forall s \in \mathcal{S}, \tag{3.30}$$

which is the expected return it receives by following the policy π. To help the agent choose a specific action, the *action-value* function can be defined similarly by

$$q_\pi(s,a) = \mathbb{E}_\pi\left[G_t|S_t=s,A_t=a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle|S_t=s,A_t=a\right], \forall s\in\mathcal{S},\forall a\in\mathcal{A}, \tag{3.31}$$

which is the expected return of the agent taking a specific action a in the state s, and following the policy π thereafter. It can be identified that the two definitions of value functions are related by

$$v_\pi(s) = \sum_a \pi(s|a) q_\pi(s,a) \tag{3.32}$$

for any given policy π.

Based on the definition of return G given in Eq. (3.28), it satisfies the recursive relation that

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3}\ldots \\ &= R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3}\ldots\right) \\ &= R_{t+1} + \gamma G_{t+1}. \end{aligned} \tag{3.33}$$

Making use of this relation, the value function can also be obtained recursively by taking the value functions of the next states

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\left[G_t|S_t=s\right] \\ &= \mathbb{E}_\pi\left[R_{t+1}+\gamma G_{t+1}|S_t=s\right] \\ &= \sum_{a\in\mathcal{A}}\pi(a|s)\sum_{s'\in\mathcal{S},r\in\mathcal{R}} p\left(s',r|s,a\right)\left[r+\gamma v_\pi(s')\right], \quad \forall s\in\mathcal{S}. \end{aligned} \tag{3.34}$$

This is called the Bellman equation for the state-value function v_π. It is a sum over all the possible actions a, next states s', and the rewards r, weighted by the probability of taking the action $\pi(a|s)$ and transitioning into the next state and reward $p\left(s',r|s,a\right)$. Similarly, the action-value function can also be recursively calculated

$$q_\pi(s,a) = \sum_{s'\in\mathcal{S},r\in\mathcal{R}} p\left(s',r|s,a\right)\left[r+\gamma\sum_{a'\in\mathcal{A}}\pi(s',a')q_\pi(s',a')\right], \quad \forall s\in\mathcal{S},\forall a\in\mathcal{A}, \tag{3.35}$$

summing over the possible next states s', rewards r, and actions to be taken in the next state a'.

The goal of the RL is to improve the policy. In MDPs, there is at least one policy π_* so that its expected return is greater or equal to that of any other policies π' for all states. The state-value function corresponding to such an optimal policy is defined as

$$v_*(s) = \max_\pi v_\pi(s), \ \forall s\in\mathcal{S}, \tag{3.36}$$

which is called the optimal state-value function. Similarly, the optimal action-value function can be defined as

$$q_*(s) = \max_\pi q_\pi(s,a), \ \forall s\in\mathcal{S},\forall a\in\mathcal{A}. \tag{3.37}$$

Inserting the optimal state-value function into the Bellman equation in Eq. (3.34) removes the dependency on a specific policy

$$v_*(s) = \max_a \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p\left(s', r|s, a\right) \left[r + \gamma v_*(s')\right], \quad \forall s \in \mathcal{S}. \tag{3.38}$$

This is called the Bellman optimality equation. The corresponding optimality equation for the action-value function is

$$q_*(s) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p\left(s', r|s, a\right) \left[r + \gamma \max_{a'} q_*(s', a')\right], \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \tag{3.39}$$

3.3.3. Basic Learning Concepts

For simple MDPs with accurately known dynamics p and a finite number of discrete states and actions, the optimal value functions v_* or q_* can be calculated exactly, for example using exhaustive searches and dynamic programming.

Dynamic programming and many modern RL algorithms utilize the value functions to guide the search for good policies. In general, the problem can be split into two parts. First, the state-value function v_π needs to be computed for a policy π. This is called the policy evaluation step. This can be done iteratively using the Bellman equation in Eq. (3.34)

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p\left(s', r|s, a\right) \left[r + \gamma v_k(s')\right], \tag{3.40}$$

where k denotes the index of the updated value function. For $k \to \infty$, v_k converges to the state-value function v_π. The second part is the policy improvement, where the obtained state-value function can be used to improve the policy. A new greedy policy with respect to the value function can be obtained

$$\pi'(s) = \arg\max_a \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p\left(s', r|s, a\right) \left[r + \gamma v_\pi(s')\right]. \tag{3.41}$$

This is strictly better than the original policy except when the original policy is optimal. By repeating these two steps, they will converge to the optimal value function v_* and the optimal policy π_*. In general, these two steps can also be interleaved, without the need to perform a full sweep in the policy evaluation step. This approach is called generalized policy iteration and is shown in Fig. 3.8.

One limitation of dynamic programming is that the system dynamics p needs to be known. If that is not the case, Monte Carlo sampling can be used to estimate the value function $v_\pi(s)$ with $V(s)$, by averaging over the sampled returns. Nevertheless, the Monte Carlo method requires a large number of samples and becomes very inefficient when the episodes are long. Temporal-difference (TD) learning mitigates this issue by bootstrapping, which is performing updates iteratively based on the estimated value functions for the next states. In its simplest case, the estimated state-value function is updated with

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right], \tag{3.42}$$

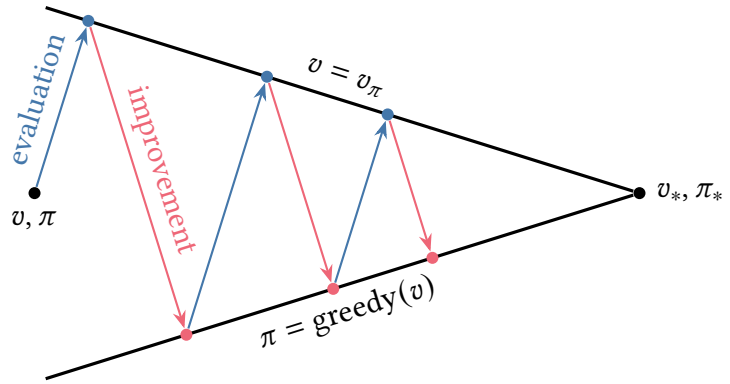


Figure 3.8.: Simplified view of the generalized policy iteration in RL. It consists of two steps, policy evaluation (blue) and policy improvement (red). From an arbitrary value function v and policy π, the policy evaluation step updates the value function to be consistent with the current policy v_π, the policy improvement step makes the policy greedy with respect to the current value function. These two steps will converge to the optimal policy π_* and the optimal value function v_*. Figure adapted from [57].

considering only one-step transition from S_t to S_{t+1} and receiving reward R_{t+1}. Here, α is the learning rate. The estimated action-value function can be updated similarly

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right], \tag{3.43}$$

and the policy can be made greedy with respect to the $Q(S_t, A_t)$. The idea of TD-learning and Monte Carlo can also be combined, resulting in the generalized TD(λ) algorithm. The corresponding λ-return used for the updates is defined as

$$\begin{aligned} G_t^\lambda &= (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t \\ G_{t:t+n} &= R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n V(S_{t+n}), \end{aligned} \tag{3.44}$$

where $\lambda \in [0, 1]$ is a decay factor, and T is the final time step if the episode terminates. The term $G_{t:t+n}$ denotes the estimated return looking n-steps ahead and using the approximate value function at the new state S_{t+n} for bootstrapping. In the case of $\lambda = 1$, the first summation goes to 0 and only the actual return G_t. This is equivalent to the Monte Carlo algorithm. For the other limit $\lambda = 0$, the second term is 0 and the first summation reduces to $G_{t:t+1}$, corresponding to the one-step TD method. The term λ characterizes how far in the future steps the algorithm will consider.

Lastly, in real-world tasks such as accelerator tuning, the state space is often continuous. The estimated value functions can no longer be stored as a look-up table but need to be parameterized, i.e. $\hat{v}_(s, \boldsymbol{w})$ and $\hat{q}_(s, a, \boldsymbol{w})$. Now it is common to use NNs to approximate the value functions, due to their capability to model arbitrary functions. This also marks the beginning of the deep RL field.

3.3.4. Policy Gradient Methods

The above methods rely on learning the value functions v_π and q_π. The behavior policy is obtained by choosing greedy actions with respect to the action-value functions. Alternatively, one could directly learn a parameterized policy $\pi(a|s, \boldsymbol{\theta}) = \Pr\{A_t = a|S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$. This also allows an effortless extension to the continuous action space case. In the case of policy learning, a value function can still exist to aid policy learning but is not mandatory. When both the policy and value approximations are being learned, the method is referred to as *actor-critic* methods, where "actor" denotes the policy and "critic" denotes the value function. The policy gradient methods learn policy parameters based on the gradient of a given performance measure $\mathcal{J}(\boldsymbol{\theta})$. The parameters are updated using gradient ascent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla \hat{\mathcal{J}}(\boldsymbol{\theta}_t), \tag{3.45}$$

where α is the learning rate and $\nabla \hat{\mathcal{J}}(\boldsymbol{\theta}_t)$ is the estimated gradient of the performance measure.

In the episodic case, the performance measure can be simply defined as

$$\mathcal{J}(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(s_0), \tag{3.46}$$

where s_0 is the episode's initial state. According to the policy gradient theorem, the gradient of performance measure can be expressed by other known quantities

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) \tag{3.47}$$

where $\mu(s)$ denotes the normalized fraction of time spent in state s by following the policy π. In its generalized form, the gradient can be expressed as

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) \propto \mathbb{E}\left[\Phi_t \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a_t|s_t)\right], \tag{3.48}$$

where Φ_t can take different forms [86], such as the return G_t, the action-value function $q_\pi(s_t, a_t)$. One choice is particularly of interest, which is the advantage function

$$\mathcal{A}_\pi(s_t, a_t) = q_\pi(s_t, a_t) - v_\pi(s_t). \tag{3.49}$$

By subtracting the state-value function, the variance in the estimated gradient updates can be reduced, leading to more stable policy learning. The advantage function is now widely used in the actor-critic algorithms for policy updates.

3.3.5. Modern RL Algorithms

These RL concepts introduced above, such as the value function, advantage function, TD, and policy gradient, are the fundamental building blocks of modern RL algorithms. A spectrum of RL algorithms have been developed in the last decade, aiming to provide higher sample efficiency, better policy performance, and more robust training progress. This section briefly introduces some modern RL algorithms that have been used in this dissertation to train RL agents for the particle accelerator tasks.

3.3.5.1. Trust Region Policy Optimization

One issue of the standard policy gradient method is that the training is not guaranteed to be stable. Due to the complexity of the RL problem, sometimes the small updates in the policy parameters can also lead to drastic performance drops. This issue is addressed in the trust region policy optimization (TRPO) algorithm [87]. The TRPO aims to maximize the following surrogate objective

$$\mathcal{L}_{\theta_k}(\theta) = \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\underbrace{\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}}_{r(\theta,\theta_k)}\mathcal{A}_{\pi_{\theta_k}}(s,a)\right], \tag{3.50}$$

where $r(\theta, \theta_k)$ is the probability ratio between the new and old policy, acting as a scaling factor in the importance sampling method. $\mathcal{A}_{\pi_{\theta_k}}(s,a)$ is the advantage function using the old policy. This measures how good a new policy π_θ performs compared to the old policy π_{θ_k}, averaged over data generated using the π_{θ_k}. The parameter is updated using

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}(\theta) \;\text{ subject to } \mathbb{E}\left[D_{\mathrm{KL}}(\pi_\theta||\pi_{\theta_k})\right] \leq \delta, \tag{3.51}$$

i.e. maximizing the surrogate objective under the constraint that the Kullback–Leibler (KL) divergence is below a certain threshold δ. In practice, the true KL divergence is usually approximated using the average one over the states visited by the old policy

$$D_{\mathrm{KL}}(\pi_\theta||\pi_{\theta_k}) \approx \overline{D_{\mathrm{KL}}}(\pi_\theta||\pi_{\theta_k}) = \mathbb{E}_{s\sim\pi_{\theta_k}}\left[\overline{D_{\mathrm{KL}}}(\pi_\theta(\cdot|s)||\pi_{\theta_k})(\cdot|s)\right] \tag{3.52}$$

The updates can be calculated using the conjugate gradient method [88], which finds the direction of the update, and a line search, which finds the proper step size.

The advantage of TRPO is that it guarantees the policy update produces a better policy, resulting in mostly monotonic performance improvement. Nevertheless, the policy can still converge to local optima.

3.3.5.2. Proximal Policy Optimization

While TRPO has desirable guarantees in the policy training process, it is often computationally intensive to solve such a constrained optimization problem. This is mitigated in the proximal policy optimization (PPO) algorithm [89]. It maximizes the following objective

$$\mathcal{L}^{\mathrm{Clip}}_{\theta_k}(\theta) = \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\min\left(r(\theta,\theta_k)\mathcal{A}_{\theta_k}(s,a), \mathrm{clip}(r, 1-\epsilon, 1+\epsilon)\mathcal{A}_{\theta_k}(s,a)\right)\right], \tag{3.53}$$

where $\epsilon \in (0,1)$ is a hyperparameter controlling how large the policy update step can be. This circumvents the constrained optimization and KL divergence by explicitly clipping the policy updates. This clipped objective can be explained as the following: if the advantage is positive $\mathcal{A} \geq 0$, the probability of taking certain action $\pi_\theta(a|s)$ will be increased but bounded to be no larger than $(1+\epsilon)\pi_{\theta_k}(a|s)$. Similarly, if the advantage is negative $\mathcal{A} \leq 0$, the probability of taking that action will be decreased, but not smaller than $(1-\epsilon)\pi_{\theta_k}(a|s)$.

In this way, it ensures that the new policy after the update is still not far away from the old policy. The updates can be simply performed by the gradient descent algorithms, such as SGD and Adam.

Thanks to its computational efficiency, the PPO can be more easily scaled up to large-scale problems with higher dimensions of action and state spaces. In addition, the sample collection in PPO can be easily parallelized, making it much faster to train when there are enough computational resources.

3.3.5.3. Twin Delayed Deep Deterministic Policy Gradient

The TRPO and PPO are both on-policy algorithms, which means that the target policy being updated is also the behavior policy that generates the data. Another class of RL algorithms is off-policy, which means that the training data is not necessarily generated by the target policy. Off-policy algorithms are usually more sample-efficient as they can utilize all the data, instead of only the ones generated with the current policy. One popular algorithm is the twin delayed deep deterministic policy gradient (TD3) [90]. It is based on the previous deep deterministic policy gradient (DDPG) algorithm, which learns a Q-function and derives the action by maximizing the Q-function. The TD3 further improves the training process in three aspects. First, it learns two Q-functions and uses the lower one to estimate the target critic update (twin), reducing the influence of potentially overestimated Q-values. Second, it updates the actor (policy) less frequently than the critic (Q-function), allowing the critic to learn the Q-value better and stabilizing the training process. Third, it added noise to the target action to prevent the policy from exploiting sharp peaks in the learned Q-function, resulting in smoother policies.

3.3.5.4. Soft Actor Critic

The soft actor-critic (SAC) [91] combines features from TD3 and stochastic policy optimization. It uses the entropy of the policy as a regularization term and aims to optimize the following performance measure

$$\mathcal{J}(\pi) = \mathbb{E}_{(s_t,a_t)\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t(R(s_t,a_t,s_{t+1}) + \alpha H(\pi(\cdot|s_t)))\right], \tag{3.54}$$

where $\alpha > 0$ is a trade-off parameter controlling the exploration and exploitation. The action-value function can be defined according to this entropy-regulated objective

$$Q_\pi(s,a) = \mathbb{E}_{(s_t,a_t)\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t(R(s_t,a_t,s_{t+1}) + \alpha H(\pi(\cdot|s_t)))\middle| s_0 = s, a_0 = a\right]. \tag{3.55}$$

Similar to TD3, the SAC algorithm uses the double Q-functions idea and performs updates using the one with smaller values. The loss function of the SAC critic update is

$$L(\boldsymbol{w}_i, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left[(Q_{\boldsymbol{w}_i}(s,a) - y(r,s',d))\right], \tag{3.56}$$

where $\boldsymbol{w}_i$ are the parameters of the i-th Q-function and $\mathcal{D}$ is the replay buffer storing the environment transitions as tuples (s, a, r, s', d), i.e. the state, action, reward, next state, and a done flag indicating the end of the episode. The target value y is defined as

$$y(r, s', d) = r + \gamma(1 - d)\left(\min_{i=1,2} Q_{\boldsymbol{w}_i}(s', a') - \alpha \ln \pi_{\boldsymbol{\theta}}(a'|s')\right), \quad a' \sim \pi_{\boldsymbol{\theta}}(\cdot|s'), \tag{3.57}$$

where a' is the next action sampled using the policy network. The policy is then updated to maximize the following policy objective function

$$\arg\max_{\boldsymbol{\theta}} \mathcal{J}_\pi(\boldsymbol{\theta}) = \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a'\sim\pi_{\boldsymbol{\theta}}(\cdot|s)}\left[\min_{i=1,2} Q_{\boldsymbol{w}_i}(s', a') - \alpha \ln \pi_{\boldsymbol{\theta}}(a'|s')\right]\right]. \tag{3.58}$$

This is performed again using the gradient descent methods. As SAC explicitly considers the entropy of the policy, it encourages the exploration behavior during training and prevents the policy from local convergences.

3.3.6. Reinforcement Learning Applications for Accelerators

In the past few years, modern RL algorithms have been applied to train intelligent agents for online optimization and control at several particle accelerators. Many of these works were conducted concurrently with the research presented in this dissertation. In the following, a non-exhaustive overview of the RL applications at accelerators is provided. For example, policy gradient algorithms have been applied at the FERMI laboratory, a seeded-FEL facility [92]. The RL agent was used to control the mirrors to align the seed laser pulse transversely with the electron bunches, maximizing the overlap and the generated FEL intensity. RL has been further used to tune the dispersion-generating magnet and the delay time between the seed laser and the electron beam to maximize the FEL energy [93]. At CERN, normalized advantage function (NAF) was used to efficiently control the electron trajectory steering tasks at AWAKE and LINAC4 [35, 94]. At the ARES accelerator, RL agents were trained to transversely control the bunch properties [36, 95], which will be further discussed in Section 6.3. A similar beam transport tuning task was demonstrated at the CAFe II accelerator [96, 97].

Another key aspect of the RL approach is the real-time control, meaning that the RL agent can process the information and take action at a very high frequency, meeting the requirement of a dynamically changing system. At the Fermilab booster, the RL agent was trained using a surrogate model and deployed in a field-programmable gate array (FPGA) to allow real-time regulation of the magnet power supply with a repetition rate higher than 15 Hz. At the KARA storage ring, RL was studied to control the microbunching instability [37, 98]. When implemented in the heterogeneous computing board *Versal*, the RL agent can interact with the accelerator with a latency of about 2.8 µs, allowing turn-by-turn control at the KARA storage ring. It has been demonstrated that this RL approach allows the agent to effectively control both the horizontal betatron oscillation [99] and the microbunching instability [38].

Most of the algorithms discussed in this chapter are model-free, meaning they do not explicitly learn or use a model of the environment's dynamics. Although they are versatile

and can be applied to almost every task, model-free algorithms often suffer from sample efficiency. Another class of RL algorithms is model-based. They are provided with a model or learn it explicitly, and use the model to predict future states and rewards. The model-based algorithms are often more sample-efficient than the model-free ones, at the cost of increased complexity of the algorithm or upfront engineering effort. At the FERMI FEL, model-based RL was applied for the same FEL tuning task and required only a few hundred samples [100]. At the AWAKE accelerator, model-based RL using GP models was tested and also demonstrated extreme sample efficiency [101, 102]. This will be further discussed in Section 6.4.3.

4. Applying Machine Learning Methods for Accelerator Simulation

Physical simulations are the cornerstones for designing and operating particle accelerators. With the growing size of accelerator systems and ever-increasing demands on beam intensity and beam quality like small emittances and short bunch lengths, the system can no longer be accurately described by single-particle dynamics and linear approximations. Detailed beam dynamics simulations are required, including higher-order transfer maps and collective effects, which increases the computational requirement and slows down the overall simulation process. The emerging machine learning (ML) methods are promising in speeding up the design process and reducing required computation resources from various aspects. This chapter uses the linear accelerator far-infrared linac and test experiment (FLUTE) [50] as a test bench and presents several case studies on ML applications for accelerator simulation and modeling. When enough training data is available, either from simulation or measurements, a surrogate model can be trained as a computationally cheap replacement. Section 4.1 uses the ASTRA [103] simulation data to train a neural network as a surrogate model for the low-energy section of FLUTE. When such a surrogate model is trained with enough data, it can offer fast and accurate predictions after training. During the design phase, when little prior knowledge exists and optimized parameters need to be found for a new accelerator configuration, Bayesian optimization (BO) can be used as an efficient global optimizer. In Section 4.2, a parallel version of BO is implemented to optimize the coherent synchrotron radiation produced at FLUTE. The simulated optimization can become more efficient if the gradient information is available. To that extent, a novel differentiable beam dynamics simulation code Cheetah [42] is developed. In Section 4.3, *Cheetah* is applied to the FLUTE tuning task using the gradient-descent algorithms. Lastly, Section 4.4 presents a proof-of-principle study on integrating Bayesian optimization and differentiable simulation to achieve more sample-efficient optimization in the presence of complicated physics effects, such as space charge. Parts of the results presented in this chapter have previously been published in [42, 104, 105].

4.1. Surrogate Modeling of FLUTE

In particle accelerator simulations, it is often crucial to account for collective effects as they limit the ultimate performance when accelerators are operating toward their limits in terms of bunch charge, bunch lengths, and emittances [46]. They are non-static effects that come from the interaction between particles themselves. The electromagnetic fields of the charged particles alter their environment, which in turn affects the dynamics of the particles themselves. As a result, modeling collective effects typically requires numerical

approximations, making the computing process much more demanding in comparison to the single-particle beam dynamics. For example, a detailed space charge particle tracking simulation can take minutes or hours to run and thus makes simulated optimization of high-dimensional parameters extremely time-consuming. This leads to the ever-increasing requirement of computation time and resources for designing modern linear accelerators with versatile operation modes, especially when the parameter optimization needs to be repeated for various conditions. To reduce the computation cost and accelerate the design process, it is beneficial to fully utilize the obtained results when calculating new working points.

Table 4.1.: Surrogate model methods and applications

Methods and structure	• Neural network [106] • Gaussian process [56] • Random forest [107]
Applications in accelerator	• Fast exploration in simulation [108, 33] • Training environment for reinforcement learning controller [39] • Replace parts of the simulations [36, 109] • Virtual diagnostics [19, 20, 110] • Assist other algorithms in online-tuning [111, 112, 113]

A surrogate model can be used to approximate the output values and provide rapid evaluations with low computational costs, in replacement of the time-consuming simulations. Surrogate modeling is a data-driven technique to map the inputs to outputs when the true relationship is either unknown or computationally expensive. Table 4.1 lists some of the methods used for building surrogate models and the application cases in particle accelerators. Common methods for building a data-driven surrogate model include Gaussian process (GP) regression [56], random forests [107], and deep neural network (NN) [106]. Among these methods, deep NN has the advantage that it can approximate arbitrary non-linear functions at the cost of a large number of required training samples. It is therefore preferred when sufficient training samples can be gathered, for example, in the case of simulation.

Once a surrogate model is trained, it can be queried at low costs for future usage, allowing a more efficient exploration of the parameter space in the design stage [33, 108]. The surrogate models can be also integrated into physical simulations to model components and sections of the accelerator [109] or to approximate the computationally intensive effects like space charge [42]. As they are fast to execute, they are also used as training environments for the reinforcement learning (RL) algorithms, which require up to millions of interactions with the environments [39].

During accelerator operation, certain beam information can only be measured destructively or take a long time, for example measuring the beam transverse phase space information using quadrupole scans and diagnostic screens. Surrogate models could be used online as *virtual diagnostics*, providing valuable information about the beam in a non-destructive way, e.g. the longitudinal phase space of the electron bunches [19, 20],

and the transverse emittance [110]. They can also be combined with other algorithms or controllers to speed up the online tuning of accelerators, such as extremum seeking [111, 21], feed-forward controllers [113], and BO.

In this section, a NN surrogate model is developed to predict the output beam parameters at the end of the low-energy section of FLUTE. The layout of the section is shown in Fig. 2.6. It consists of an RF photo-injector which generates the electron bunch and brings the energy up to 7 MeV, a solenoid magnet to focus the beam and compensate for the space charge effect, screens for transverse beam images, and a spectrometer dipole to diagnose the beam energy.

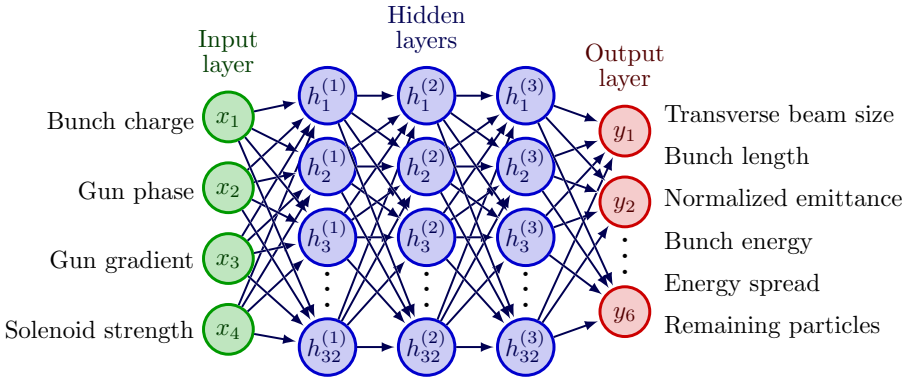


Figure 4.1.: Structure of the NN surrogate model for FLUTE low-energy section

4.1.1. Training the Surrogate Model

The input layer of the surrogate model NN consists of 4 neurons representing the adjustable parameters, i.e. the bunch charge, the gun RF phase, the maximal gun RF gradient, and the solenoid magnetic field. The bunch charge can be controlled by changing the photo-injector laser intensity, while other parameters are directly adjustable. These are the most important parameters which govern the behavior of the beam in the low-energy section. The output layer returns 6 scalar values representing the bunch properties, namely the horizontal beam size σ_x, bunch length σ_z, mean energy E, relative energy spread σ_E, normalized transverse emittance ϵ_x, and percentage of the remaining particles N_{remain}. As FLUTE usually operates with a round beam, the vertical beam size $\sigma_y = \sigma_x$ is not considered here. These bunch properties can also be measured using the diagnostic devices [114] so that the model can be further retrained and fine-tuned to match the measurement results.

A fully connected feed-forward NN with 3 hidden layers is used, as shown in Fig. 4.1. Each hidden layer has 32 neurons and the hyperbolic tangent function (tanh) as the activation function. The size of the network is chosen to sufficiently approximate the transfer map from the photocathode to the beginning of the linac. At the same time, the NN is not too large to fully memorize the dataset, also referred to as *over-fitting*, which can lead to poor generalization capability to unseen scenarios.

The training dataset consists of 10 000 samples randomly selected from the parameter space in Table 4.2. Each parameter configuration is simulated using an ASTRA tracking

Table 4.2.: Input parameter ranges used to generate training data for the surrogate model. The RF phases are given in ASTRA's convention, i.e. the global absolute phase offset with respect to the start of the tracking.

Input Parameters	Range
Charge	1 - 30 pC
Gun RF phase	175 - 235 deg
Gun RF gradient	50 - 100 MV/m
Solenoid field	0.08 - 0.2 T

Table 4.3.: Neural network training parameters. The loss function and optimizer are common choices. The batch size, learning rate, and epoch number are tuned via hyperparameter scan across multiple training trials.

Hyperparameters	Value
Loss function	MSE
Batch size	64
Optimizer	Adam
Learning rate	0.001
Epoch	200

simulation with the space charge effect. Before feeding into the NN, the input and output parameters are min-max normalized, i.e. mapped to [0,1] intervals, to speed up the training process. The output normalization also ensures that each predicted bunch property is equally weighted in the loss calculation. Table 4.3 lists the hyperparameters used in the training process. Adam optimizer is used, with a batch size of 64 and an initial learning rate of 0.001. The loss function to be minimized is the mean squared error (MSE) between the predicted and the target bunch properties. Training is performed for 200 epochs to prevent overfitting on the dataset.

The trained model is then evaluated on test data, consisting of 1000 random parameter settings on which the NN is not trained. The prediction error is shown in Fig. 4.2, where almost all the predicted values show good agreement with the target outputs. The prediction of the mean energy E is slightly lower than the target values, but the overall discrepancy is still well under 0.1 MeV. After training, the surrogate model can predict the beam properties very fast. As an example, Fig. 4.3 shows a 2D subspace of the 4D input parameter space of the transverse beam size σ_x depending on the solenoid magnetic field and the bunch charge Q. The left plot is generated by ASTRA simulations with a 10×10 grid, which takes ~5 h if executed sequentially. The right plot is a 50×50 grid predicted by the trained NN surrogate, which only takes milliseconds as the prediction can be easily parallelized. As shown in the figure, the NN-predicted beam sizes have a similar structure as the ASTRA simulation results, e.g. the solenoid field required to minimize the beam size at the screen is slightly increasing with the bunch charge due to space charge effects. Deviations from the ideal field strength lead to larger σ_z due to under- or over-focusing

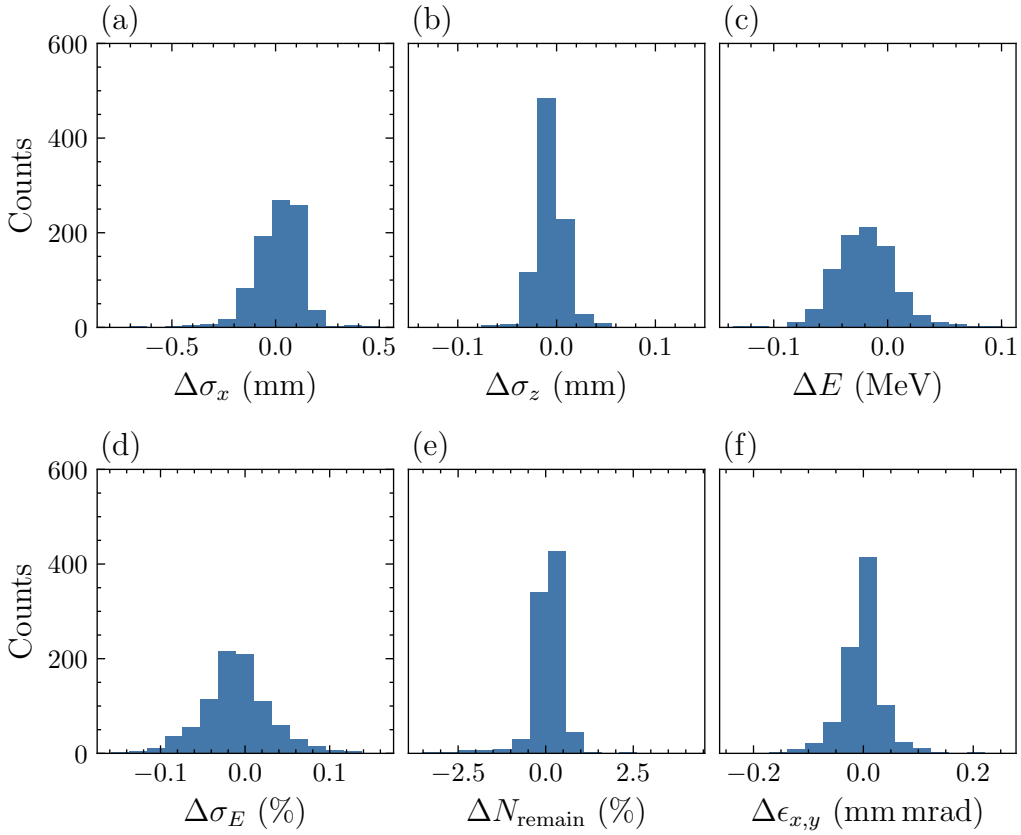


Figure 4.2.: Prediction error on the test data for each output variable: (a) transverse beam size, (b) bunch length, (c) mean energy, (d) relative energy spread, (e) remaining particles, and (f) transverse emittance.

of the bunch. The minimum beam sizes, marked as white stars, are very comparable, with $\sigma_x = 0.166\,\text{mm}$ for the ASTRA simulation and $\sigma_x = 0.164\,\text{mm}$ for the NN prediction, corresponding to a relative error of 1.2 %. While this level of accuracy is sufficient for the proposed applications at FLUTE, the model's performance can be further improved if more accurate predictions are required. Possible methods include adding more training data, tailoring the model structure to the accelerator section, and retraining the model on real-world measurements.

4.1.2. Applications of the Surrogate Model

As the surrogate model is trained to approximate the dynamics of the underlying physical system, it can provide insights into the importance of the input parameters with respect to the output beam properties. The Shapley additive explanations (SHAP) method [115] can be used to calculate the feature importance of the input parameters for arbitrary nonlinear mappings. Such methods are very useful in complex systems, as the commonly used Pearson correlation coefficients can only work with linear dependencies. Figure 4.4 shows the SHAP values for the transverse beam size σ_x prediction. The results on other output beam parameters and more details on the method can be found in Appendix A.1.

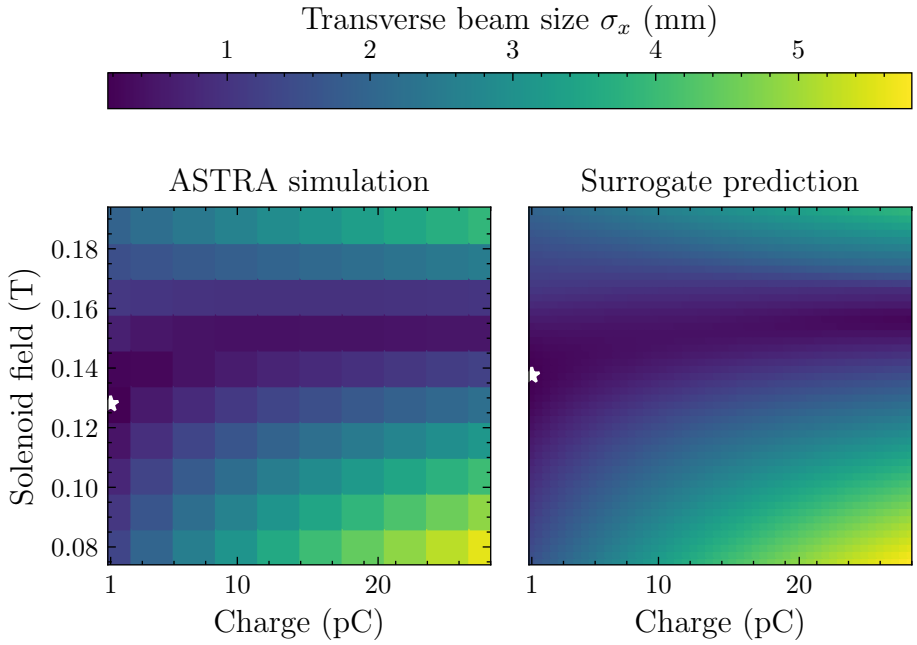


Figure 4.3.: Beam size prediction by the surrogate model (left) and the ASTRA simulation results (right). The minimal beam sizes are marked with white stars for the respective method. The results are obtained by varying the solenoid magnetic field and the bunch charge while keeping the gun RF parameters fixed. The NN predictions take less than 1 ms on a common laptop. The ASTRA simulations each take about 3 min on the same device and can be parallelized using multiple CPU cores.

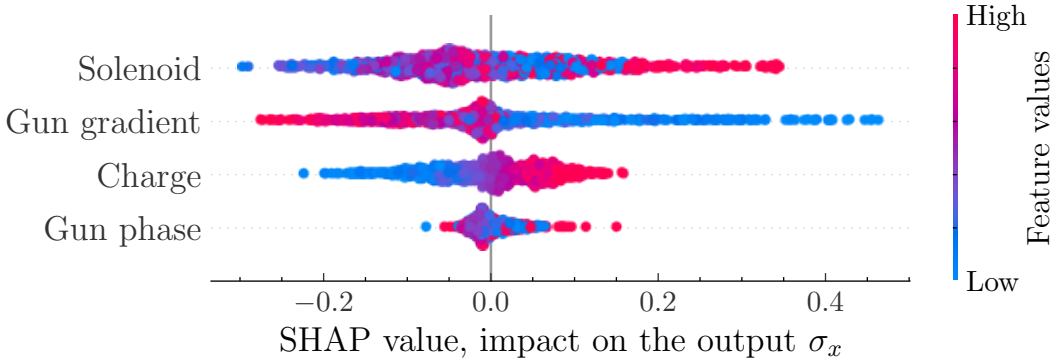


Figure 4.4.: Feature importance study of the surrogate model for the transverse beam size σ_x prediction using the kernel SHAP method [115]. The SHAP values are calculated for individual input parameters and denote their attribution for predicting the output σ_x. For more details on the method and the results of other output beam parameters, see Appendix A.1.

For each input parameter, its SHAP values denote the attribution of the input parameter on the output, evaluated over a set of test points with different values of the other input

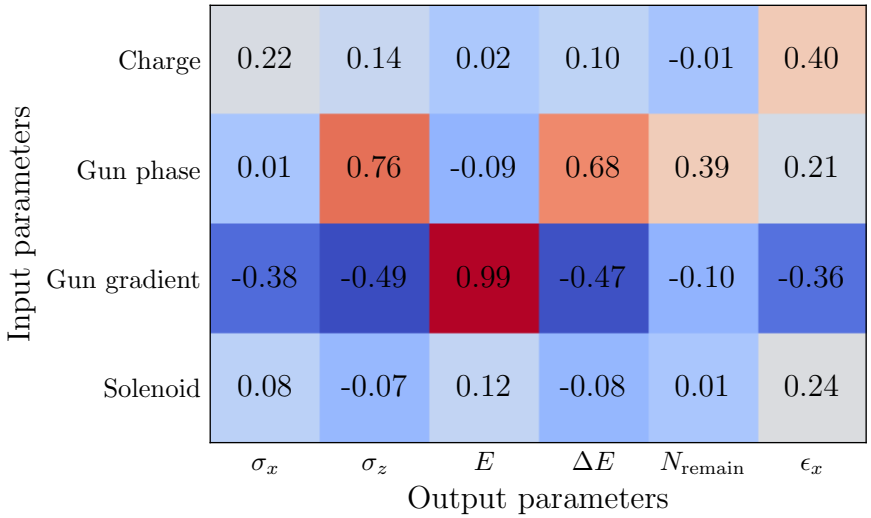


Figure 4.5.: Pearson correlation coefficients between the input parameters (x-axis) and the output beam parameters (y-axis) at the FLUTE low energy section.

parameters. In this plot, the input parameters are ordered by the amount of impact they have on the prediction. The points are color-coded based on the values of the corresponding input parameters. The solenoid field strength has overall the most significant impact on the beam size σ_x. This effect is expected as the solenoid magnet is designed to transversely focus the beam. It is worth noting that there are regions with overlapping red and blue points, meaning that different solenoid fields can lead to the same beam sizes. This is because, with the same solenoid field strength, the beam can be optimally focused (negative impact on σ_x), or over-focused when the beam energy is lower (positive impact on σ_x). The second most important parameter is the gun RF gradient, which determines the energy gain of the bunch and thus the space charge effect. Its impact is easier to identify, higher radio frequency (RF) (red) leads to higher beam energy and a more focused beam (negative impact on σ_x) at the end of the section. The gun phase and the bunch charge have a smaller impact, but correlations can still be identified.

As a comparison, the Pearson correlation coefficients are calculated between the input parameters and the output beam parameters, using the same test data points. The results are shown in Fig. 4.5. For features with a clear linear or monotonic dependency, such as the beam energy and the gun RF gradient, the correlation coefficient is very high as expected. Nevertheless, it fails to explain the non-linear dependencies, especially in the case where the input parameter has a different impact when other inputs are varied. Looking at the solenoid strength and the beam size σ_x, for example, the correlation is close to zero and appears insignificant, which is contradictory to the results obtained by the SHAP method. Such commonly used linear correlation methods can be thus misleading when it comes to explaining the data and unveiling the dependencies in a non-linear system.

Although in this simple case of a photo-injector gun section, these correlations are already well known, it is expected that methods like SHAP can provide valuable insights in more complex nonlinear systems, especially when the beam dynamics are not fully

understood. In such a case, these methods can be used to guide the physicists both in further simulation studies and in the actual operation of the accelerator.

4.1.2.1. Virtual Diagnostics

One of the use cases of a trained surrogate model is to provide a virtual diagnostic for the accelerator operation. Thanks to its fast inference time, the surrogate model can be used to predict the beam parameters shot-to-shot in a non-destructive manner. At the FLUTE accelerator, for example, the beam energy E is measured destructively with a spectrometer dipole magnet. The resulting bending angle is proportional to the bunch energy for a given magnetic field. By varying the dipole magnet strength, the beam can be deflected and observed on a Yttrium Aluminum Garnet (YAG) screen in the spectrometer arm, where the energy can be determined from the bunch position. Even with an automated procedure, an energy measurement could take up to several minutes, as the dipole needs to be cycled to avoid the remnant field due to magnetic hysteresis affecting the normal operation. In this case, the predictions provided by the surrogate model can be used to guide the operation and speed up the machine setup process.

Figure 4.6 shows the comparison of the surrogate model predictions to the energy measurements for a photo-injector gun RF power scan [116]. The upper plot shows the beam energy over the gun power and the lower plot shows the differences with respect to the measured values. For each data point, an ASTRA tracking simulation and surrogate model prediction are performed with the corresponding accelerator parameters. The computation time of the surrogate model is within a millisecond, which is negligible compared to $\sim$ 1 h of required ASTRA simulation time and several hours of beam time needed for the measurement. Both the surrogate model predictions and the ASTRA simulation results correspond well to the measurement data with a deviation of under 0.1 MeV. The surrogate model predictions follow more closely the values of the ASTRA outputs, as the model is trained on the simulation data. There are also systematic errors between the simulation results and the measurement, which can be explained by the calibration in the power supply and non-ideal magnetic and RF fields in the real-world setup.

It needs to be noted that the NN model was only trained on the data in the ranges shown in Table 4.2. In particular with a minimum gun gradient of 50 MV/m, which corresponds to a gun power of ~3.8 MW in real-world measurement. This means that the model extrapolated to an unseen parameter range for the first few points in the plot, showing the capability of the model to predict the beam properties in a wider range of operation conditions.

In addition, Fig. 4.7 shows the same comparison for the RF phase scan, where the surrogate model also shows good agreement with the measurement results. Similar to the case for the energy scan measurement, there are systematic errors between the simulation and measurement results. It is worth noting that the ASTRA tracking simulation failed for the RF phase value at 178 deg, due to the loss of the reference particle. In the real world, part of the beam was emitted from the gun and could be measured. The trained NN still predicted a value by extrapolation, even if it is at the edge of the training range. Nevertheless, the difference between the NN prediction and measurement for that RF

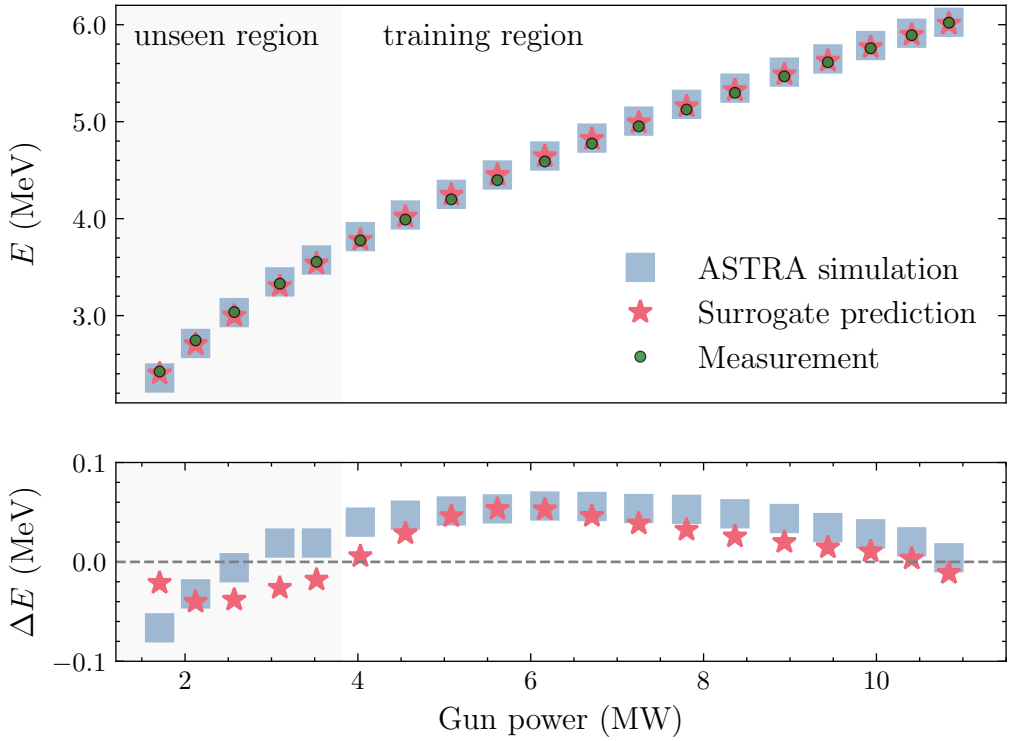


Figure 4.6.: Energy measurements (green dots) for a FLUTE photo-injector RF power scan compared to ASTRA simulations (blue squares) and surrogate model predictions (red stars). The lower plot uses the measurement values as a reference and shows the remaining differences between the simulation results and the NN predicted values. Each measurement is averaged from 20 shots with a standard error of about 0.1 %. The shaded region denotes settings with low RF powers that are unseen during the NN training.

phase is larger than the ones within the training region. Such cases can be identified if the surrogate model also provides the uncertainty estimate along with its prediction. This can be achieved for example using GP or an ensemble of NNs [117].

The NN model can be further improved by retraining on the measurement data, which could not only reduce the discrepancy between simulation and measurement but also mitigate the long-term drifts of the accelerator components.

4.1.2.2. Fast Evaluation for other Algorithms

Apart from providing virtual diagnostics, the surrogate model can also be used to aid other algorithms. One example is the accelerator tuning with RL [35, 36, 39, 95]. The training process of the RL agents is notoriously time-consuming, often taking millions of interactions with the environments. Therefore, it is infeasible to train RL on computationally expensive simulations or directly on the accelerator with a low repetition rate, which will take about a year of continuous beam time at a real-world accelerator [36]. The surrogate model can be used as a training environment for RL, allowing the agent to learn

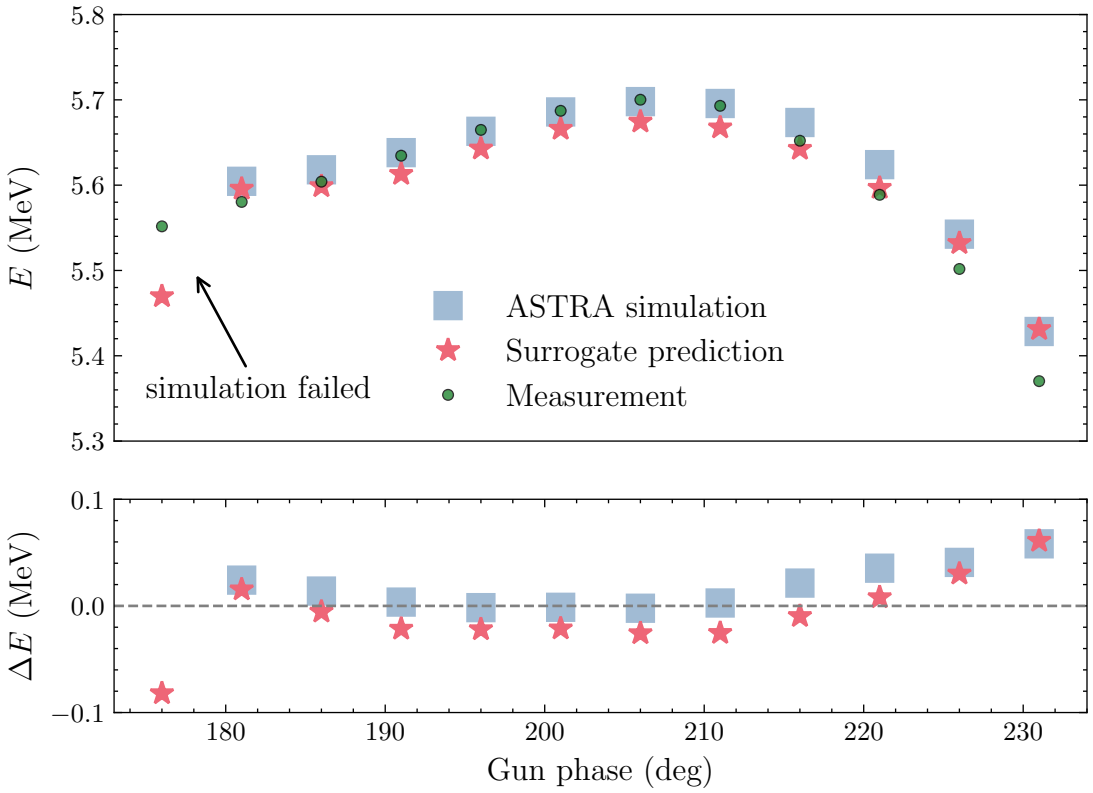


Figure 4.7.: Energy measurements (green dots) for a photo-injector RF phase scan of the gun, compared to ASTRA simulations (blue squares) and surrogate model predictions (red stars). The lower plot shows the differences between the predicted values and the measurement values. Each measurement is averaged from 20 shots with a standard error of about 0.1 %. Phase values are given as in the ASTRA convention. For the RF phase at 178 deg, the ASTRA simulation failed because of the loss of the reference particle.

the policy and minimize the beam time needed to retrain on the accelerator. Although the surrogate model has certain discrepancies compared to the actual accelerator, robust RL agents can still be trained on it that can be further adapted to the real-world accelerator. This approach minimizes the beam time needed for the deployment of an RL agent at the accelerator. It has been demonstrated in several cases where the RL is trained on simplified simulation and transferred to the real-world accelerator subsequently [36, 96]. This will be further discussed in Chapter 6.

Last but not least, a NN surrogate model can be used in combination with other optimization algorithms, such as BO, to speed up the optimization process. This can be achieved via different approaches, such as constraining the parameter space to explore, providing correlation information of the input parameters, or acting as the prior mean module for the GP function [42, 112]. This will be further discussed in Section 4.4.

4.2. Simulated Optimization for Intense THz Radiation

In the design phase of a new particle accelerator, beam parameters need to be optimized or reach certain threshold values by changing the lattice parameters, such as the magnet strengths and the lengths of the components. These objectives are often conflicting and depend non-linearly on the tuning parameters. The accelerator optimization also needs to be repeated for different working points, i.e. beam parameters need to be optimized at a fixed final energy or fixed bunch charge. Sometimes the optimization for one accelerator section needs to be run again due to the change in the upstream beam condition. In these scenarios, there is usually limited or no simulation data available for training an informative surrogate model. Physical prior knowledge is available and can identify a subset of tuning parameters that most significantly affect the desired output beam characteristics. At the same time, the task is inherently complicated so no analytical solution can be directly derived. Consequently, the best accelerator parameters need to be determined efficiently using some numerical optimization algorithms. Methods like line search and Nelder-Mead simplex are effective but prone to local optima so they need to be repeated with multiple initial conditions. Grid scan and random search are valid methods for finding the global optimum in the accelerator settings, but their sample requirements scale exponentially with the number of tuning parameters and quickly become infeasible when a large accelerator section is considered. ML algorithms, especially the BO, are promising candidates to solve such global black-box optimization tasks efficiently.

This section uses the FLUTE coherent synchrotron radiation (CSR) optimization task as an example to demonstrate how such ML-based algorithms can be applied to aid the simulated optimization of particle accelerators. At the FLUTE accelerator, one of the scientific goals is to generate and study intense radiation in the THz range. This requires optimal beam compression, which is achieved by creating an energy chirp, i.e. longitudinal position-dependent beam energy, and setting the bunch compressor strength accordingly. As the electron bunches at FLUTE are low-energy ($E < 50\,\mathrm{MeV}$) and can have high bunch charges, the beam dynamics are strongly influenced by collective effects and the accelerator settings need to be jointly optimized.

4.2.1. Calculation of the Coherent Synchrotron Radiation Generation at FLUTE

The emitted synchrotron radiation from an electron bunch can be calculated based on Eq. (2.28), where the amount of coherent emission is governed by the form factor $F(\omega)$. In the relativistic case, only the longitudinal form factor $F_l(\omega)$ is considered. For lower frequency regions where the form factor is close to 1, the coherent radiation scales quadratically with the number of electrons in the bunch, leading to a significant enhancement of the radiation intensity. For an arbitrary electron bunch, the longitudinal form factor F_l can be calculated

$$F_l(\omega) = \left| \int_{t=-\infty}^{\infty} \varrho(z) e^{(\mathrm{i}\omega z/c)} \mathrm{d}z \right|^2 , \tag{4.1}$$

where $\varrho(z)$ is the normalized longitudinal charge density of the bunch. By reducing the bunch length σ_z, the form factor F_l can be extended to higher frequencies, leading to an increase in the overall CSR intensity. Therefore, a natural objective for the optimization task is to minimize the root mean square (RMS) bunch length σ_z.

Since the longitudinal bunch distribution can not be simplified as a Gaussian, the RMS bunch length does not always correspond to the actual form factor, especially when substructures are present due to collective effects. As an alternative, the peak electric field of the THz pulse can be used as an objective function, which leads to a more direct optimization of the THz pulse despite requiring more processing steps of the output particle distribution. Here, the CSR pulse emitted by an arbitrarily shaped bunch is calculated following the semi-analytic approach introduced in [118, 119]. The E-field is given by

$$\begin{aligned} \boldsymbol{E}(t) &= N_\mathrm{e} \int_{-\infty}^{\infty} \boldsymbol{E}_0(\tau)\varrho(t-\tau)\,\mathrm{d}\tau \\ &= N_\mathrm{e}\frac{1}{\pi}\mathrm{Re}\int_0^{\infty} \tilde{\boldsymbol{E}}_0(\omega)\,\tilde{\varrho}(\omega)\,e^{\mathrm{i}\omega t}\,\mathrm{d}\omega\,, \end{aligned} \tag{4.2}$$

where N_e is the number of electrons and $\boldsymbol{E}_0$ is the single-particle electric field. The equation is expressed in the time domain as the bunch is relativistic, and $\varrho(t)$ denotes the normalized charge density of the electron bunch in the time domain. For simplicity, the higher order terms in the originally proposed spline interpolation are dropped and ϱ is approximated using a stepwise linear interpolation

$$\varrho_j(t) \sim \left(\kappa_j(t-t_{j-1}) + c_{0,j}\right)\left(\Theta(t-t_{j-1}) - \Theta(t-t_j)\right), \tag{4.3}$$

where j denotes the index of the interpolated segments, Θ is the Heaviside step function, $c_{0,j}$ is a constant offset, and κ_j is the slope of the charge density in the j-th segment

$$\kappa_j = \frac{\varrho(t_j) - \varrho(t_{j-1})}{t_j - t_{j-1}}. \tag{4.4}$$

With the linear interpolation, the Fourier-transformed charge density $\tilde{\varrho}(\omega)$ is obtained analytically via

$$\tilde{\varrho}(\omega) = \sum_{j=1}^{N} \frac{\kappa_j}{\mathrm{i}\omega}\left(e^{\mathrm{i}\omega t_j} - e^{\mathrm{i}\omega t_{j-1}}\right). \tag{4.5}$$

In the frequency domain, the single-particle electric field is given by [44]

$$\tilde{\boldsymbol{E}}_0(\omega) = \frac{\sqrt{3}e}{4\pi\epsilon_0\, c\, R}\gamma\left(1+\gamma^2\theta^2\right)\frac{\omega}{\omega_c}\left[K_{2/3}(\xi)\boldsymbol{u}_\sigma + \mathrm{i}\frac{\gamma\theta K_{1/3}(\xi)}{\sqrt{1+\gamma^2\theta^2}}\boldsymbol{u}_\pi\right], \tag{4.6}$$

where $\xi = \omega(1+\gamma^2\theta^2)^{3/2}\Big/2\omega_c$ as defined in Eq. (2.27), K_ν is the modified Bessel function, θ is the observation angle, and R is the distance from the observation point to the source of the emission. The electric field contains two polarization components, where $\boldsymbol{u}_\sigma$ and $\boldsymbol{u}_\pi$ denote the unit vectors for the σ- and π-polarization components respectively.

In the following studies, the case $R = 1\,\mathrm{m}$ and $\theta = 0$ is considered, which means only the σ-polarization contributes to the electric field of the pulse. Inserting Eq. (4.5) and Eq. (4.6) into Eq. (4.2) gives

$$E_\sigma(t) = \frac{-3N_e e}{8\pi\epsilon_0 cR}\frac{\gamma(1+\gamma^2\theta^2)}{\omega_c} \times \sum_{j=1}^{N}\kappa_j\left[\cosh\left(\frac{2}{3}\operatorname{arcsinh}\left(\frac{t-t_j}{\tau_c}\right)\right) - \cosh\left(\frac{2}{3}\operatorname{arcsinh}\left(\frac{t-t_{j-1}}{\tau_c}\right)\right)\right] \quad (4.7)$$

and $|\boldsymbol{E}(t)| = E_\sigma(t)$ for $\theta = 0$.

Two objective functions are considered and maximized for the THz radiation generation at FLUTE based on the equations above.

$$\begin{aligned} \text{Bunch length}: \ f &= -\sigma_z, \\ \text{Max E-field}: \ f &= E_{\max} = \max_t |\boldsymbol{E}(t)|\,. \end{aligned} \quad (4.8)$$

The FLUTE accelerator is implemented as a start-to-end model using simulation codes ASTRA [103] and OCELOT [120]. The electron bunch is first created by ASTRA and tracked from the photocathode to the entrance of the bunch compressor, including the space charge effect along the track. The ASTRA output particle distribution is then fed into an OCELOT model, which tracks the bunch further until the end of the magnetic chicane and, in addition, includes the CSR effects. Two different simulation codes are required here because ASTRA has the capability of handling real field maps in the photocathode gun and OCELOT contains the CSR calculations. The resulting beam distribution is used to calculate the respective objective function f. For the simulation presented below, 10 000 macro particles were generated and tracked in ASTRA and OCELOT, where each simulation took about 10 min on a single CPU core.

4.2.2. Parallelized Bayesian Optimization

Many derivative-free algorithms can be used to find out the best accelerator parameters in such an optimization task, such as random search, iterative line scans, and genetic algorithms. However, these methods suffer from local convergence and low sample efficiency. They scale poorly when the number of tuning parameters increases. BO is a promising sample-efficient algorithm for black-box optimizations. In its vanilla formulation, as introduced in Section 3.2, the next point to evaluate is selected by maximizing an acquisition function $\alpha(x)$, which is built on a GP model and aims to balance the exploration and exploitation behavior. In the simulation case, the evaluation can often be run in parallel when computational resources are available, e.g. multiple CPU cores or using a computing cluster. It is desirable if the algorithm can provide multiple points to speed up the overall optimization process. In BO, this is referred to as parallel or batch BO. Taking the expected improvement (EI) acquisition as an example, the batch points can be calculated by jointly optimizing the expected value of improvement [64]. Although this works for a small batch size, the computation time for the joint optimization scales exponentially with the batch

size. Therefore, this is undesirable in a scenario where the function evaluation itself is not very time-consuming (~ 10 min) and rather large batch sizes are possible, as the acquisition function optimization would take a non-negligible fraction of overall computation time. When the batch size becomes too large, the benefit of using a GP model to propose valuable settings decreases. If the extra computation resources can be afforded, one could consider a hybrid GP-genetic algorithm method [121] or a common evolutionary algorithm.

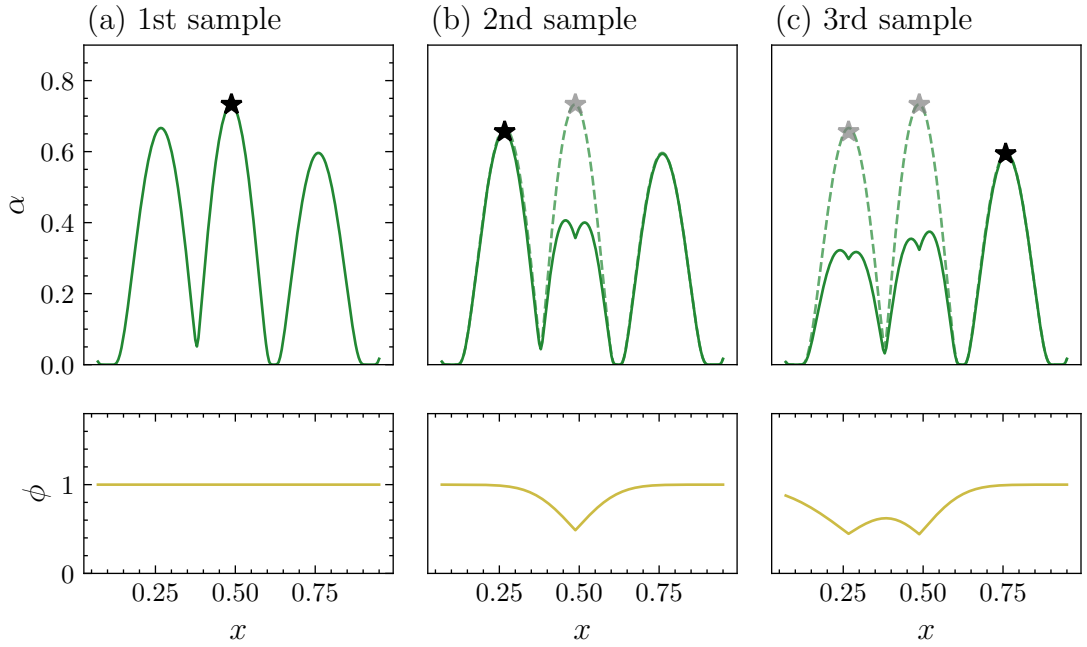


Figure 4.8.: Penalized acquisition function for parallel BO. After selecting the first point, a subsequent point to evaluate (black star) is selected by maximizing the penalized acquisition function (green line), which is obtained by the product of the original acquisition function α (green dashed) and a penalization function φ (yellow line) centered at previously sampled points (gray star). This process is repeated until the desired amount of points is chosen.

An alternative approach for parallel BO is proposed in [122] based on local penalization of the acquisition function, as visualized in Fig. 4.8. For a positive acquisition function like expected improvement (EI), the k-th sample to evaluate in optimization step t is selected by

$$\begin{aligned} \boldsymbol{x}_{t,k} &= \arg\max_{\boldsymbol{x}\in\mathcal{X}} \{\alpha(\boldsymbol{x})\phi\} \\ &= \arg\max_{\boldsymbol{x}\in\mathcal{X}} \left\{ \alpha(\boldsymbol{x}) \prod_{j=1}^{k-1} \phi(\boldsymbol{x}, \boldsymbol{x}_{t,j}) \right\}, \end{aligned} \tag{4.9}$$

where $\phi(\boldsymbol{x}, \boldsymbol{x}_{t,j})$ are the local penalizers centered at the previously selected points $\boldsymbol{x}_{t,j}$ in the same batch. The local penalizers smoothly reduce the value of the optimal around $\boldsymbol{x}_{t,j}$, so that the batch of points is not concentrated in one region of the parameter space. In such a way, the selected batch is expected to both better explore the parameter space and

have a higher chance of finding the global maximum. Here, the penalization function is defined as

$$\begin{aligned} \phi(\boldsymbol{x}, \boldsymbol{x}_j) &= \frac{1}{2}\mathrm{erfc}(-z) \\ z &= \frac{1}{\sqrt{2\sigma^2 \boldsymbol{x}_j}} \left(L||\boldsymbol{x}_j - \boldsymbol{x}|| - f_{\mathrm{best}} + \mu(\boldsymbol{x}_j)\right), \end{aligned} \tag{4.10}$$

where erfc is the complementary error function, f_{best} is the best-observed function value, σ^2 is the posterior variance, μ is the posterior mean predicted by the GP model, and L is a Lipschitz constant. The behavior of this penalizer can be seen in the bottom plots in Fig. 4.8. Its value is close to 1 for regions far away from previously selected points $\boldsymbol{x}_j$ and goes to zero when approaching these points, effectively excluding the neighborhood of $\boldsymbol{x}_j$. The Lipschitz constant L controls the width of the exclusion region and can be defined based on the characteristic of the objective function. Using this approach, the batch of points to be evaluated in the next optimization step can be effectively selected, with the computation time scaling linearly with the batch size.

4.2.3. Optimization Settings

The electron bunch must be optimally compressed and have high bunch energy to generate intensive CSR radiation. The final bunch energy depends on the RF settings at the photoinjector gun and the linac. The beam compression in the bunch compressor requires an energy chirp, which is created by the off-crest acceleration at RF fields with the photoinjector and the linac at FLUTE. The final bunch length σ_z is limited by the non-linearity in the energy chirp, which is in turn attributed to the non-linearity in the RF fields and the collective effects. Moving away from the on-crest phase reduces the non-linearity in the sinusoidal accelerating field that the bunch experiences at the cost of lower energy gain and higher energy spread. At the same time, this leads to a stronger space charge effect and a lower single-particle radiation power. Such a trade-off in the parameter settings makes the THz optimization a non-trivial task. Consequently, the optimization task takes six accelerator settings as the input parameters, because they have the most influence on the THz radiation generation. The ranges of the input parameters are listed in Table 4.4.

Table 4.4.: Input parameters for the simulated optimization of THz pulse generation at FLUTE. The RF phases are given in ASTRA's convention, i.e. the global absolute phase offset with respect to the start of the tracking.

Parameter	Range
Gun RF phase	180 - 220 deg
Gun RF amplitude	100 - 120 MV/m
Linac RF phase	150 - 190 deg
Linac RF amplitude	9 - 12 MV/m
Solenoid strength	0.07 - 0.15 T
Bunch compressor angle	0.12 - 0.13 rad

The parallel BO algorithm was run on a computing cluster with 20 parallel evaluations and a maximum of 50 optimization steps. A single optimization run took a total of about 6 hours. The majority of the time was spent on the simulation, with the inference time of BO being negligible in comparison. For each optimization configuration, 20 random sample points are used to initialize the GP model. The GP hyperparameters like the variance σ^2 and the kernel lengthscales $\boldsymbol{\ell}$, are fitted dynamically during the optimizations.

4.2.4. Optimization Results

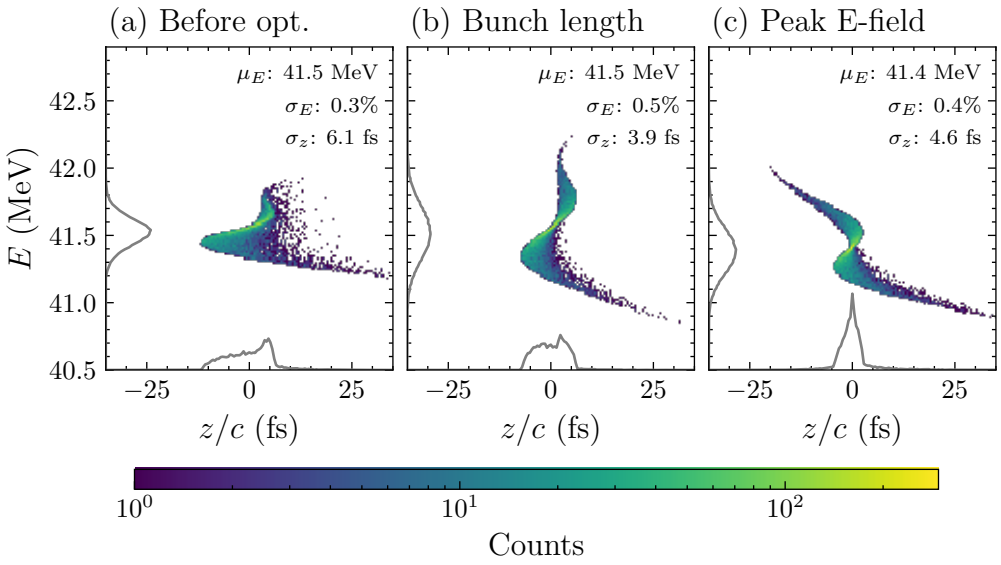


Figure 4.9.: Longitudinal phase spaces of the optimized 1 pC bunches for (a) the previous design settings in the design stage, (b) the BO result when using the bunch length as the objective, and (c) the BO result when using the peak electric field as the objective.

First, the low charge case with 1 pC bunch charge is considered. The initial particle distribution is generated by a Gaussian laser pulse with a transverse size of 0.25 mm and a pulse length of 700 fs. The longitudinal phase spaces of the electron bunches from the optimization results are shown in Fig. 4.9, with the result from the design stage settings shown in (a) as a comparison. The one-dimensional projections of the beam distributions are shown in the gray curves. All the settings produced bunches with similar final energy at about 41 MeV. The bunch length optimization further compressed the bunch down to $\sigma_z = 3.9$ fs. On the other side, optimization for peak electric field leads to a higher peak current where a large portion of the charge is centered in the bunch, even though with a larger RMS bunch length of 4.6 fs.

The corresponding longitudinal form factors $F_l(\omega)$ are calculated and shown in Fig. 4.10 (a). The form factor is close to one for lower frequencies and decreases rapidly above the critical frequency $\omega_c \sim 1.4 \times 10^{14}\,\mathrm{s}^{-1}$ (corresponding to about 22 THz). The angular integrated synchrotron radiation intensity spectra are shown in Fig. 4.10 (b). The spectrum

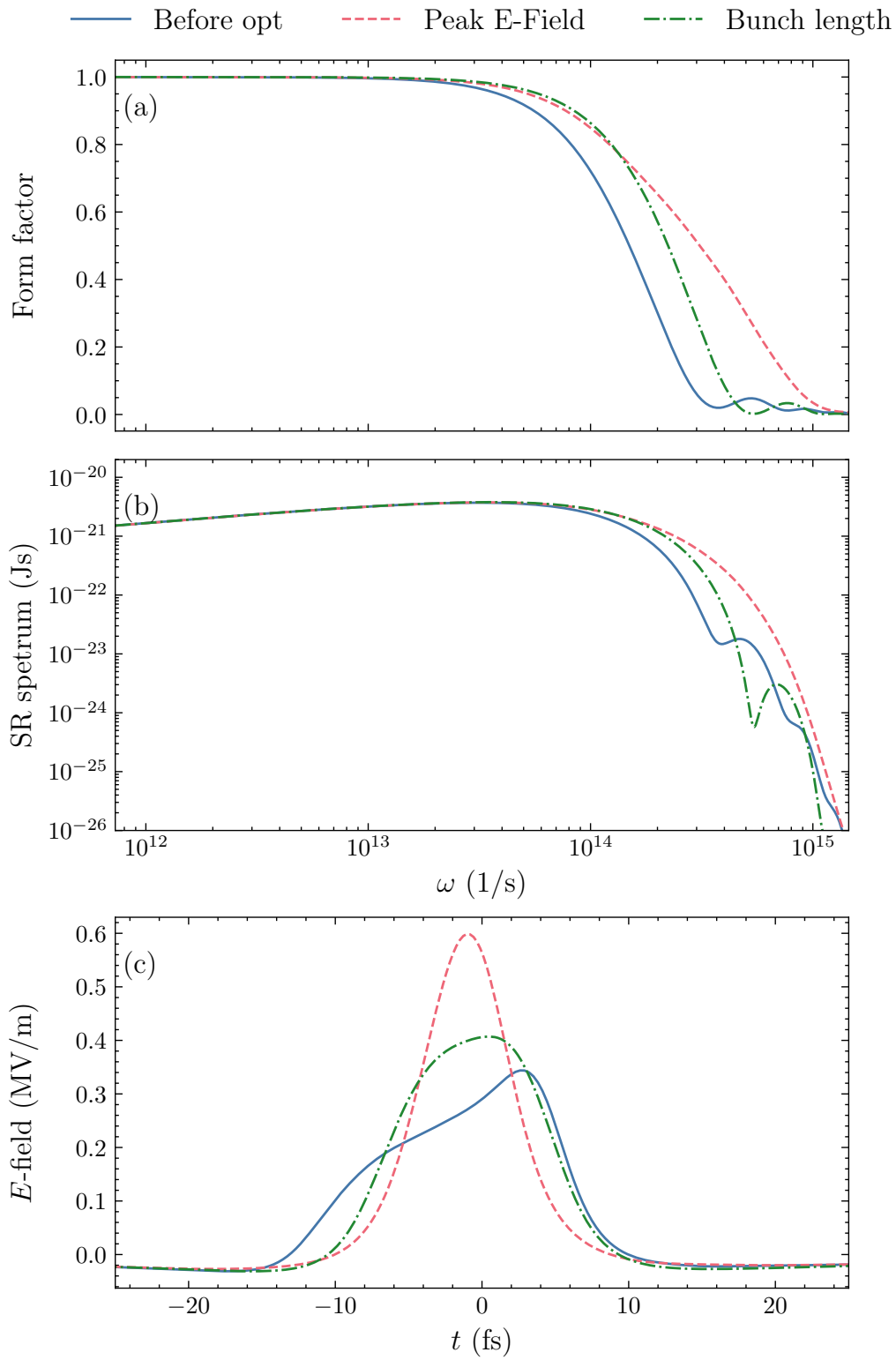


Figure 4.10.: Optimization results for the 1 pC bunch: the design stage settings (before optimization), BO optimization result when using the bunch length as an objective, and the BO optimization result when using the peak electric field as an objective. (a) shows the longitudinal form factors for three bunches. (b) shows the angular integrated synchrotron radiation spectra, and (c) shows the temporal electric field profiles of the generated THz pulses.

of the $E_{\max}$-optimization extends to higher frequencies and has a smooth curve, while the spectra for other settings show visible side bumps in the high-frequency region due to the substructures in the bunch with heterogeneous charge densities. Lastly, the electric fields of the THz pulses are calculated and shown in Fig. 4.10 (c). The electric field optimization reaches a peak electric field of 0.6 MV/m, which is higher than the design settings with 0.35 MV/m by a factor of ~ 1.7 and the bunch length optimization with 0.4 MV/m. For the design settings, the THz pulse shape follows roughly the shape of the bunch current profile. Decreasing the bunch length further smooths the bunch structure. Afterward, the resulting THz pulse approaches a Gaussian shape with a full width at half maximum (FWHM) of about $1/\omega_c$ (corresponding to about 7 fs).

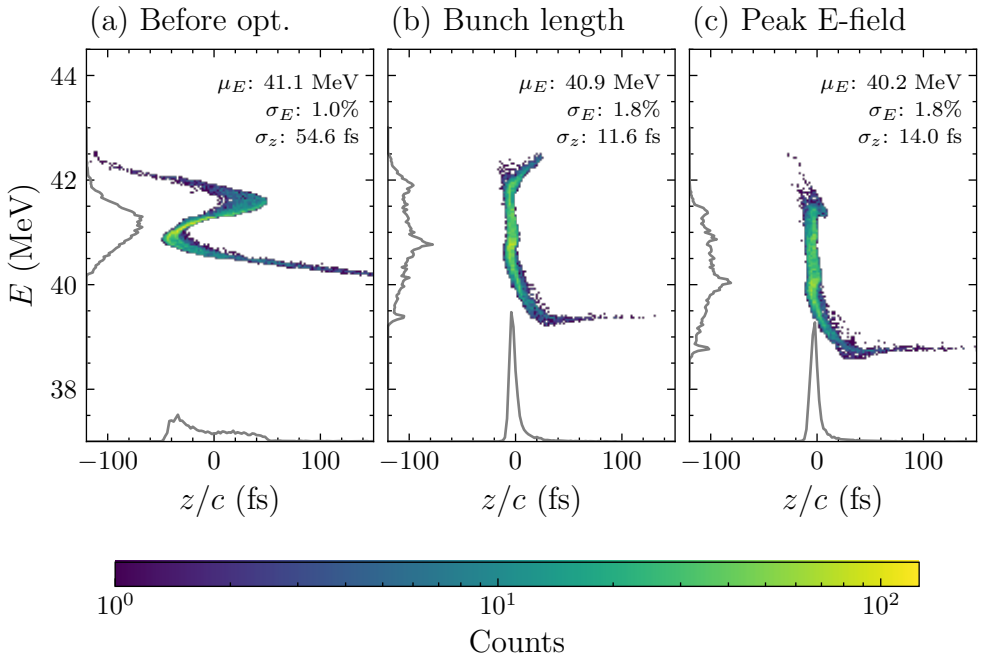


Figure 4.11.: Longitudinal phase spaces of the optimized 100 pC bunches for (a) the previous design settings in the design stage, (b) the BO result when using the bunch length as the objective, and (c) the BO result when using the peak electric field as the objective.

The same optimization is run for a high bunch charge case with 100 pC. The initial bunch distribution is generated from a Gaussian laser pulse with a transverse size of 0.25 mm and a pulse length of 2 ps. The optimized bunch length $\sigma_z = 11.6$ fs is smaller than the design stage value of 54.6 fs by a factor of ~ 4.7. Both bunches from BO optimizations have a higher energy spread, which is a side effect of moving away from the nominal linac phase for a more linear accelerating gradient and thus a better compression. In comparison to the 1 pC case, the final bunches for the 100 pC case are still longer than $1/\omega_c$. Therefore, minimizing the bunch length is still a valid strategy to maximize the electric field of the CSR pulse, and both objectives are similar. Reducing the bunch length σ_z still efficiently extends the form factor to higher frequencies and increases the overall CSR pulse intensity, as shown in Fig. 4.12. Both BO results show similar spectrum intensities, which are more

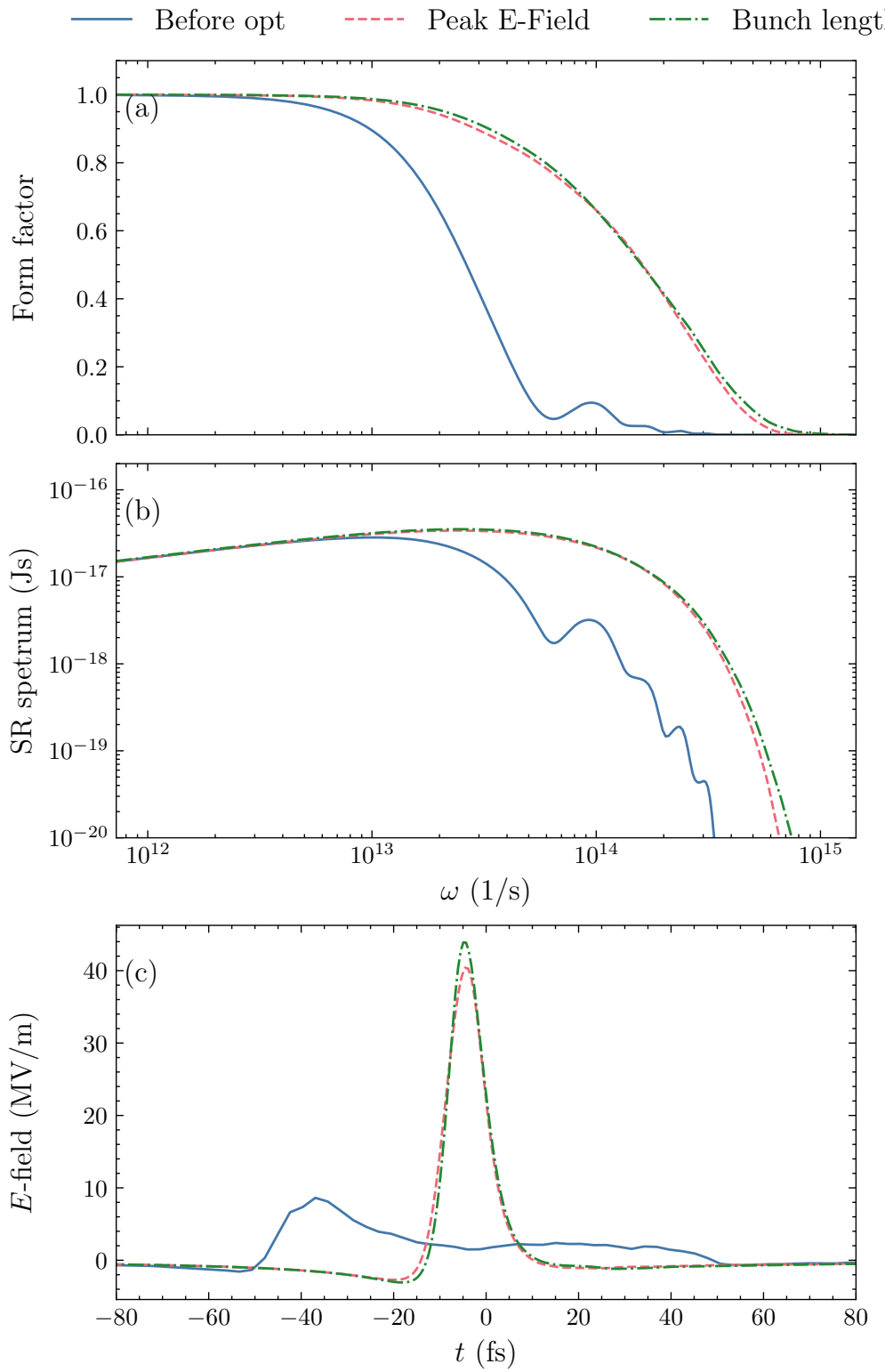


Figure 4.12.: Optimization results for the 100 pC bunch: the design stage settings (before optimization), BO optimization result when using the bunch length σ_z as an objective, and the BO optimization result when using the peak electric field as an objective. (a) shows the longitudinal form factors for three bunches. (b) shows the angular integrated synchrotron radiation spectra, and (c) shows the temporal electric field profiles of the generated THz pulses.

intense than the one from the design settings. A peak E-field of about 43 MV/m is reached, which is higher than the previous design value at 8.4 MV/m by a factor of 5. The generated THz pulse can also be optically focused and further enhanced by two orders of magnitudes, reaching a peak field in the order of ~ GV/m.

Using the same method, the radiation generated at FLUTE with other mechanisms like coherent transition radiation (CTR) and edge radiation (ER) can be optimized in the same way, simply by replacing the CSR calculation with the respective radiation calculations. Moreover, this optimization setup can be readily extended to a multiobjective task. This is achieved by using multiobjective acquisition functions such as the expected hypervolume improvement as demonstrated in [79]. In such a task, a Pareto front is to be determined with multiple competing objectives, such as the highest final beam energy with the highest radiation intensity. In the design phase, this helps unveil the possible ranges of beam parameters. The obtained Pareto front contains more information than the single-objective optimization result and allows physicists to fix the layout and parameters of the accelerator a posterior, depending on a specific trade-off between the objectives [74]. In online operations, efficient multiobjective optimizations can find different operation modes with focuses on different beam parameters, for example, trading off the maximal bunch charge and the minimal energy spread that can be achieved in a laser-plasma accelerator [80].

4.3. Differentiable Beam Dynamics Simulation

As shown above, ML-based algorithms like BO are very effective in finding the optimum in a black-box optimization problem for accelerator simulations. However, as the number of input parameters increases, the parameter space to be optimized scales exponentially, and the required number of steps will increase accordingly. This inevitably requires more computational resources and slows down the overall design process of future large accelerators with many tuning parameters. One possibility to mitigate this issue is by using gradient-based optimization algorithms, which often outperform gradient-free algorithms in terms of sample efficiency. Recent results in ML show that they are even successful in optimizing up to billions of parameters [123].

4.3.1. Simulation Code Cheetah

Conventional beam dynamics simulations only provide the output beam parameter as the result. The gradient of the output with respect to an input parameter, such as a magnet strength, can only be estimated numerically, for example using finite differences. This scales poorly with the number of input parameters and requires a large number of simulations to be run every time. This limits in practice the number of input parameters that can be optimized simultaneously using gradient-based optimizers. The emerging ML libraries allow an alternative technique called *automatic differentiation*. It decomposes all the computations in a function mapping into atomic operations, such as addition and multiplication, and uses the chain rule to automatically obtain the partial derivatives up to arbitrary orders. Using these novel software libraries, a new generation of simulation codes

can be built that automatically deliver the derivative information together with the tracking results. This allows the gradient-based optimization to be applied for high-dimensional optimization tasks in particle accelerator simulations. To this end, I developed together with Jan Kaiser the differentiable simulation code *Cheetah* [42]. It is one of the first beam dynamics simulation packages that supports full differentiability, dedicated to enabling seamless integration of ML methods into particle accelerators. Several other differentiable codes were developed around the same time with specialized features, including Bmad-X [124] with supports backend-agnostic implementations of particle tracking routines, a differentiable space charge simulation [125] based on truncated power series algebra (TPSA) which supports forward-mode differentiation, a differentiable beam dynamics package JuTrack [126] implemented using the Julia programming language supporting TPSA, and a differentiable synchrotron light simulation [127].

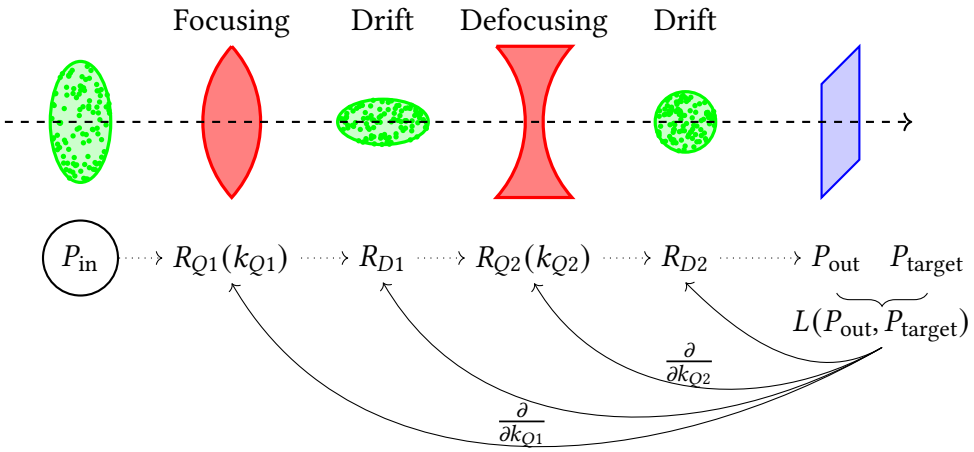


Figure 4.13.: Differentiable simulation of a FODO lattice in Cheetah. An incoming beam P_{in} (denoted by the green ellipse) is sequentially tracked through the FODO lattice consisting of two quadrupoles $\{Q1, Q2\}$ (in red) and drift spaces $\{D1, D2\}$ with linear transfer matrices $\{R_{Q1}, R_{D1}, R_{Q2}, R_{D2}\}$ respectively. The final beam P_{out} is compared with the desired target beam P_{target} to calculate a loss function L. The quadrupole strengths $\{k_{Q1}, k_{Q2}\}$ can be optimized using gradient descent, where the gradients are obtained by backpropagation.

Cheetah benefits from the optimized tensor computations implemented in PyTorch and allows high-speed tracking of the particles, reaching more than 3 orders of magnitude speed-up compared to other commonly used simulation tools. In Cheetah, each particle is represented using a 7-dimensional vector[1]

$$\boldsymbol{x}_{\text{Cheetah}} = (x, p_x, y, p_y, z, \delta, 1), \tag{4.11}$$

with the first six dimensions being the canonical phase space coordinates $\boldsymbol{x}$ as defined in Eq. (2.11). The 6D phase space vector is expanded at the end, analogous to an affine

[1] The description here is based on the version *v0.7*, while the explanation in [42] refers to an earlier version *v0.6*.

space, allowing a coherent representation of transfer maps for real-world effects such as magnet misalignment and thin-lens magnets. The macroparticles are combined in a so-called `ParticleBeam` P with the shape $(N_{\text{particles}}, 7)$ to represent the electron beam.

The accelerator components are implemented in the `Element` classes in Cheetah, such as magnets, RF cavities, drift spaces, and diagnostic devices. By default, the elements compute linear beam dynamics using an implementation of the linear transfer map $R_{\text{Cheetah}} \in \mathbb{R}^{7\times7}$

$$R_{\text{Cheetah}} = \left(\begin{array}{c|c} R_0 & \vdots \\ \hline 0 \cdots 0 & 1 \end{array}\right), \tag{4.12}$$

with $R_0 \in \mathbb{R}^{6\times6}$ being the standard transfer matrix based on [128]. When tracking a beam P_{in} through an element with a transfer matrix R, the outgoing beam is

$$P_{\text{out}} = P_{\text{in}} R^{\intercal}. \tag{4.13}$$

This process is visualized in the schematic in Fig. 4.13. For a FODO section with two quadrupole magnets and drift spaces, the outgoing beam is

$$P_{\text{out}} = P_{\text{in}} R_{Q1}^{\intercal} R_{D1}^{\intercal} R_{Q2}^{\intercal} R_{D2}^{\intercal}, \tag{4.14}$$

where the transfer maps $\{R_{Q1}, R_{Q2}\}$ depend on the quadrupole strengths $\{k_{Q1}, k_{Q2}\}$ respectively. If a specific beam distribution P_{target} at the final position is desired, a loss function L can be defined to quantify the difference between P_{out} and P_{target}. Using the automatic differentiation (AD) functionalities in PyTorch, the partial derivatives to the quadrupole strengths $\{\frac{\partial L}{\partial k_{Q1}}, \frac{\partial L}{\partial k_{Q2}}\}$ can be directly obtained.

4.3.2. Differentiable Modeling of the THz CSR Generation at FLUTE

To apply gradient-based algorithms, the optimization task needs to be end-to-end differentiable. The high-energy section of the FLUTE lattice is implemented in Cheetah, allowing differentiable particle tracking from the entrance of the linac until the end of the bunch compressor. The remaining part lies in the calculation of the charge density profile ρ from the macroparticle phase space coordinates. In the calculations presented in Section 4.2, this step was performed using the histogram method, rendering the process non-differentiable.

An alternative approach that retains differentiability is the KDE. The estimated density at a given position is calculated by summing the kernel values of all the samples, often using a Gaussian kernel. Figure 4.14 illustrates this approach using the longitudinal phase space of the 100 pC bunch. The KDE can smoothly approximate the structure of the charge density profile. The corresponding discrete histogramming method is also shown below for comparison. This method is detailed further in Appendix A.2.

Using KDE for the charge density estimation, the electric field of the emitted CSR pulse can be calculated in a differentiable way. To ensure the stability of the gradient-based optimization, the input parameters are scaled and normalized to the same range. As the gradient descent is an unconstrained algorithm, the physical constraints on the parameters can be incorporated via transformations by nonlinear activation functions. For example, a parameter with a lower bound can be transformed by rectified linear unit

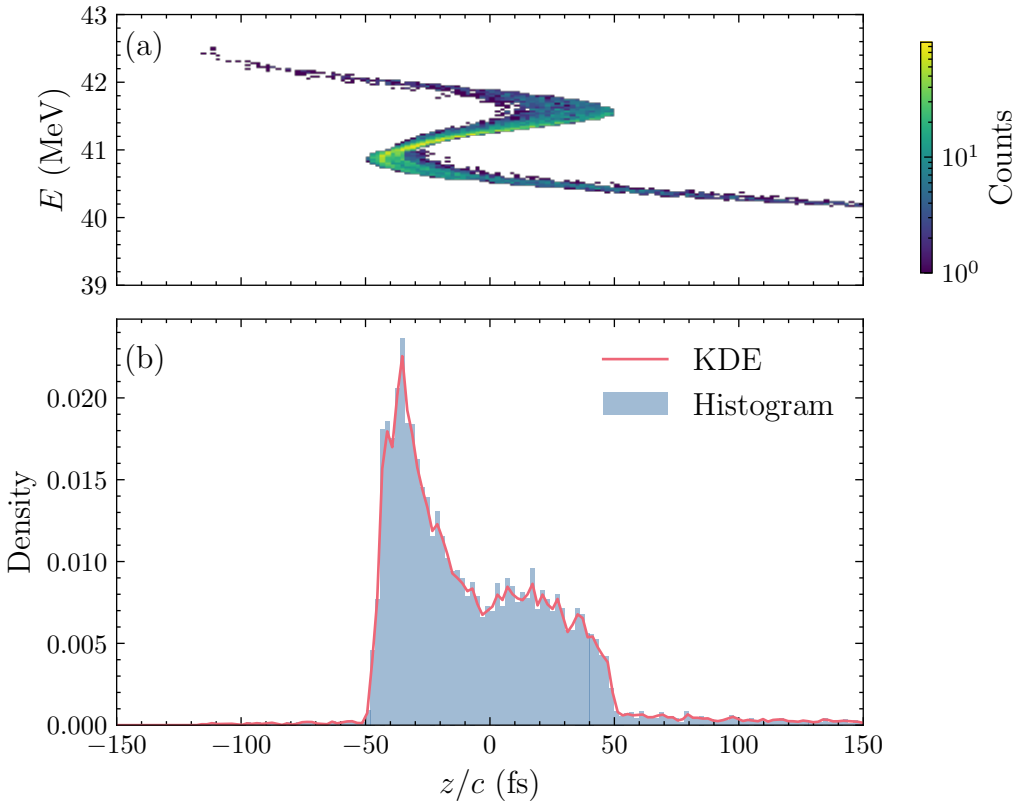


Figure 4.14.: Kernel density estimation for the longitudinal beam profile. (a) The longitudinal phase space of an 100 pC electron bunch. (b) The charge density estimation using histogram (blue) compared to the kernel density estimation (KDE) method (red).

(ReLU), exponential linear unit (ELU), or Softplus functions, and sigmoid or tanh can be used to represent a parameter with both lower and upper bounds. Other general inequality constraints can also be introduced via the log barrier method [88].

Figure 4.15 shows the progress of a gradient-based optimization for the THz electric field using the linac phase, linac voltage, and the bending angle. The initial beam distribution at the entrance of the linac is taken from the ASTRA simulation result with the optimal setting, and the other parameters being optimized are randomly initialized. The optimization is performed using the Adam optimizer with an initial learning rate of 0.005 and in total 1500 steps. The electric field of the THz pulse was smoothly maximized and convergence was observed. Thanks to the fast-executing simulation and the automatic differentiation (AD), such an optimization process took only a few seconds, allowing computationally cheap evaluations of new accelerator configurations.

Recent advances in differentiable simulation also include the propagation of synchrotron light through optical elements such as lenses, mirrors, and detectors [127]. Further combination of such a differentiable optics simulation with the Cheetah beam dynamics simulation will allow also end-to-end differentiable modeling of the accelerator, from an initial electron bunch to the observed light in either beamline experiments or diagnostic

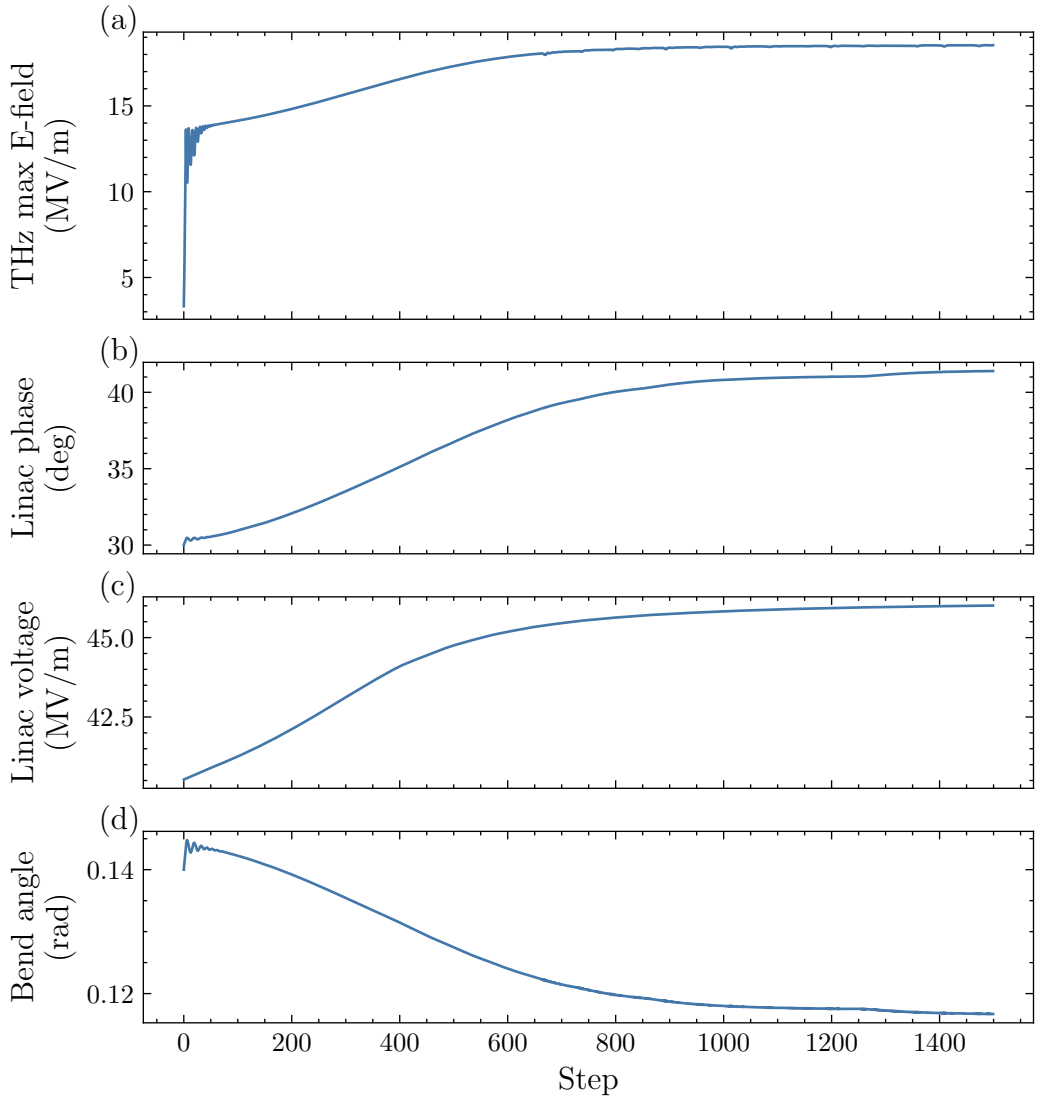


Figure 4.15.: Gradient-based optimization of the THz radiation using the differentiable simulation code Cheetah. The maximum electric field of the THz pulse is to be maximized using the linac RF phase, voltage, and the bunch compressor bending angle.

devices. This will also open the door for more targeted optimization, with objectives tailored to the needs of beamline users such as a specific temporal profile or certain spectral component of the light.

The longitudinal phase spaces of the beam distributions at different stages are shown in Fig. 4.16. (a) shows the initial beam distribution before entering the linac. (b) shows the beam at the end of the bunch compressor before the optimization, where the beam is not optimally compressed. After running the optimization, the output beam is optimally compressed as shown in (c). It can be seen that the resulting beam phase space is much smoother than the ones obtained in Section 4.2. The reason for the discrepancy is that the FLUTE simulation model implemented in Cheetah does not include collective effects yet.

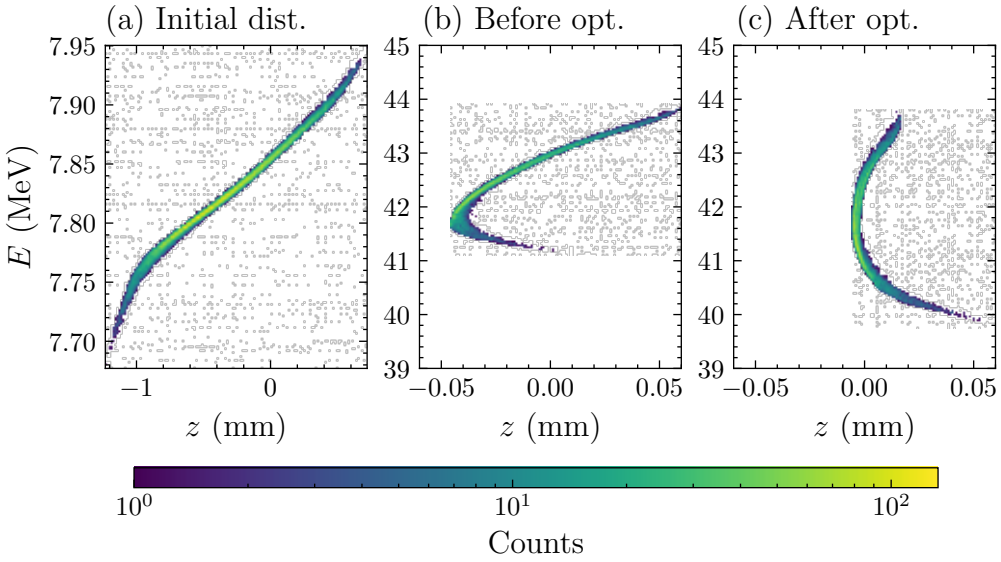


Figure 4.16.: Longitudinal phase space of the electron bunches (a) at the entrance of linac, (b) at the end of bunch compressor before the optimization, and (c) at the end of bunch compressor after the optimization.

For the 100 pC electron bunch with about 40 MeV energy studied here, the space charge and CSR effects have a strong influence on the beam dynamics.

In Fig. 4.17, the tracking is repeated in OCELOT with different physical effects activated. In (a), the bending angle of the bunch compressor is scanned while keeping the linac parameters fixed, leading to different bunch lengths σ_z when including collective effects. It can be identified that including space charge (SC) has the strongest influence. In comparison, CSR has a small impact as the beam is compressed only at the final dipole magnet, and the CSR is negligible before that. Overall, the collective effects affect the beam dynamics at FLUTE. If they are ignored, the optimization can lead to settings that cannot be achieved in the real world.

This is one of the current limitations of using gradient-based optimization with differentiable simulations. At the time of completing this dissertation, there are ongoing efforts to implement more physical effects such as space charge into Cheetah. Including these effects will increase the fidelity of the simulation and allow more accurate and reliable results using gradient-based optimization in the future. Nevertheless, as more and more effects are taken into consideration, the computation requirement inevitably grows. This also poses a challenge to the gradient calculation, both for the memory requirement for storing all the computation steps and for numerical stability when propagating the gradient back to the parameters to be tuned. The feasibility of performing gradient-based optimization in complicated simulations remains an open question and will be investigated in the future. One alternative approach is to integrate the NN surrogate model with Cheetah simulation. This has been initially demonstrated in [42], where a NN is trained to model the beam tracking through a quadrupole magnet including space charge effects. As Cheetah implements the accelerator components also in PyTorch modules, a NN model can be seamlessly integrated into the existing lattice elements. Such a combination allows a

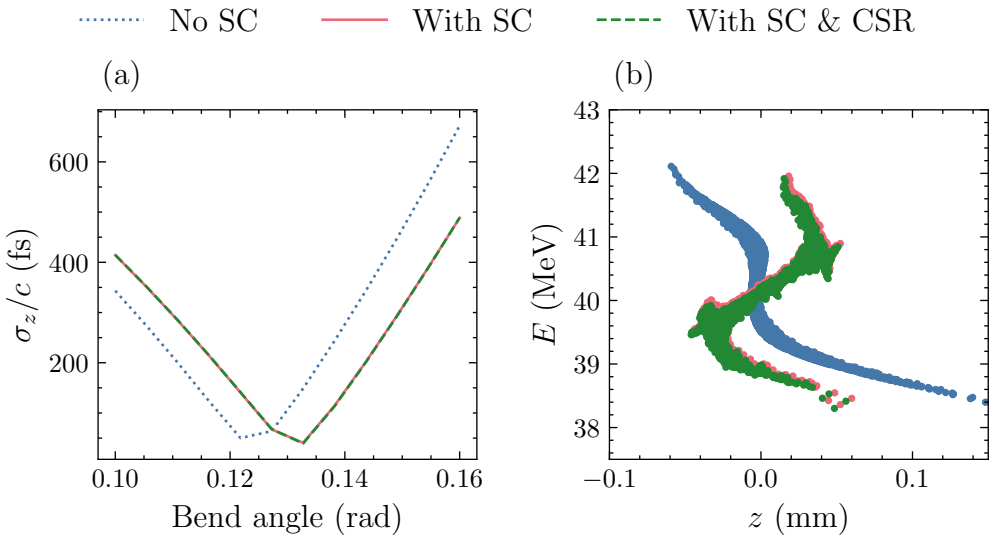


Figure 4.17.: Tracking results of a 100 pC using the same accelerator settings with different physical effects activated, no space charge (SC) (blue dotted), with space charge (SC) (red line), and with SC and CSR (green dashed). (a) Final RMS bunch lengths with a bending-angle scan. (b) Longitudinal phase spaces of the output beams.

data-driven modeling of the complex collective effects and the particle tracking remains computationally cheap.

4.4. Bayesian Optimization with Physics-informed Prior

Even though the reduced physics model in Cheetah does not fully reproduce the real-world experimental results, it can still be utilized to guide the search for the optimum setting in a computationally intensive simulation, reducing the number of evaluations using the complex simulation and lowering the overall computational cost. One example of such an approach is combining the differentiable simulation Cheetah with the BO algorithms. Available physical information on the task can be integrated into BO in various ways, such as calculating the correlation between the input parameters to better construct a kernel function [28], utilizing the results from different simulations explicitly as a multi-fidelity problem [74, 16]. In this section, the feasibility of using a fast-executing simulation like Cheetah directly as the prior mean function for BO is explored.

For the BO results presented in Section 4.2, a constant zero prior mean $m \equiv 0$ is used. In general, any function mapping from the input parameters to the objective function values can be used. This changes the posterior prediction of the GP model on test points X^* from Eq. (3.14) to

$$\boldsymbol{\mu}^* = \boldsymbol{m}(X^*) + K(X^*, X)K(X, X)^{-1}(Y - \boldsymbol{m}(X^*)), \tag{4.15}$$

where $K(\cdot, \cdot)$ is the kernel function and $\{X, Y\}$ are the data points.

When using an informed prior mean function that contains structural information on the objective function, the GP can approximate the objective function with fewer data points, and the efficiency of BO can be greatly improved. This effect is visualized in Fig. 4.18.

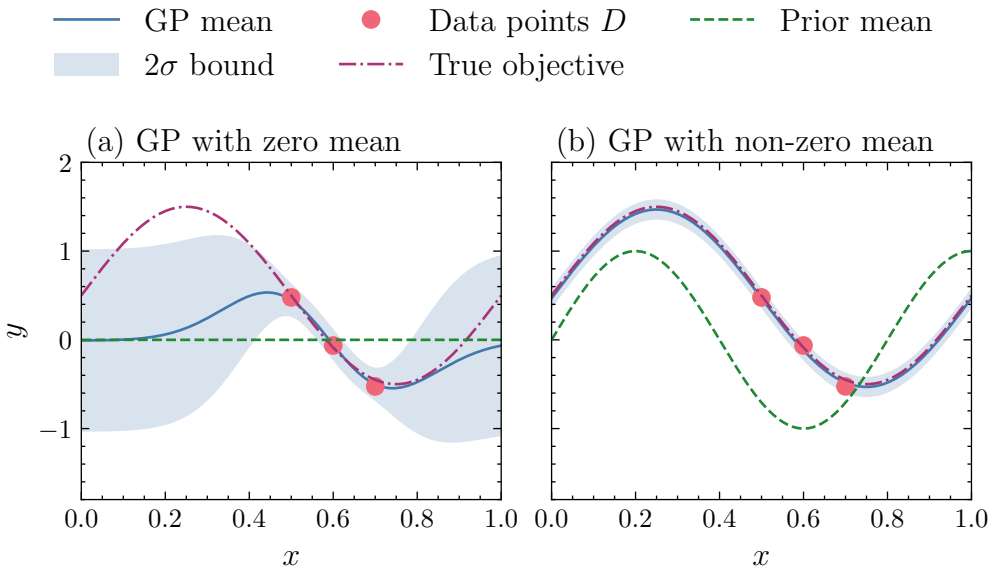


Figure 4.18.: Effect of using a non-zero prior mean for the GP modeling. Based on the three observation points (red dots), a GP model is fitted to an unknown objective function (red dash-dot line), and the posterior mean (blue line) and uncertainty (blue-shaded region) predictions are shown. (a) GP uses the constant zero mean $m \equiv 0$. (b) GP uses an informative prior mean that resembles the shape of the underlying objective.

The GP is built on three observation points (red dots) to model an underlying objective function (red dash-dot line), which has a sinusoidal form. In (a), the zero prior mean is used for the GP model. The resulting posterior prediction contains large uncertainty in regions that are distant from the observed points and the posterior mean fails to model the true objective function. In comparison, when using a sinusoidal function as the prior mean in (b), the posterior prediction contains less uncertainty and approximates the objective function much better than the zero prior mean case.

In practice, the implementation of the prior mean function for GP should be fast-executing and differentiable. As the prior mean function needs to be repeatedly evaluated both for fitting the GP model to the data and maximizing the acquisition function, a slow-executing prior mean function will take prohibitively long to compute. This makes a differentiable simulation like Cheetah an ideal choice as a physics-informed prior for BO. Alternatively, for slow simulation or real-world data where fast simulation is not available, data-driven models like NNs can be trained and used as prior mean functions [79, 112].

For the THz optimization task at FLUTE, again the 100 pC case was considered. The tracking was performed in ASTRA and OCELOT taking space charge and CSR into account. The BO using Cheetah as the prior mean function was compared to the vanilla BO using

zero prior mean and latin hypercube sampling (LHS). Fig. 4.19 shows the optimization progress of the three methods, where each algorithm was repeated 10 times with the same initial setting and different random seeds. As expected, both BO variants converged to settings with higher objective values than the LHS. BO with a physics-informed prior mean leads to faster optimization and converges to a better setting, even when the low-fidelity Cheetah simulation is mismatched to the ASTRA and OCELOT simulation including the collective effects. The final result obtained using BO prior mean is comparable to the one obtained via parallel BO in Section 4.2, which took 1000 evaluations in total. This corresponds to a tenfold saving in computation resources. If the wall time for the optimization process is to be minimized, the physics-informed BO can be run in parallel as well.

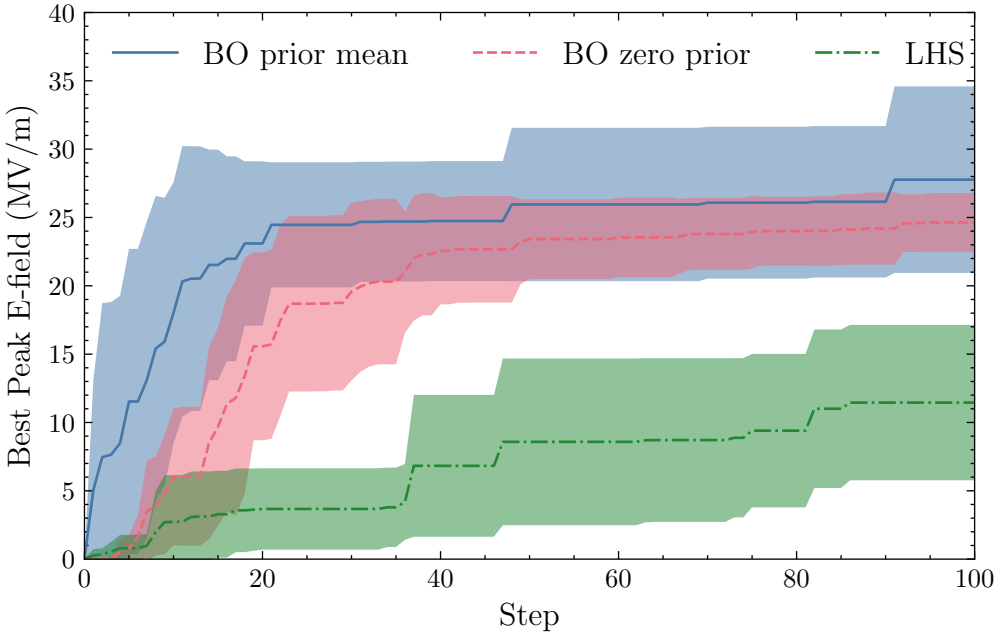


Figure 4.19.: Progress of BO using the physics-informed prior mean compared to BO with zero prior mean, and latin hypercube sampling (LHS). All optimization runs started from the same initial setting. Each algorithm was repeated 10 times with different random seeds for calculating the standard deviations, which are shown in the shaded region.

The output beam distributions obtained from the best settings of each algorithm are further investigated. Figure 4.20 shows the longitudinal phase spaces of the compressed beams after the bunch compressor. For both BO results, the electron bunches are well compressed, and only a sharp single peak can be seen in the projected longitudinal beam profile. The final bunch lengths are 14.2 fs for BO with prior mean and 12.2 fs for BO with zero prior mean. For the LHS result, the final bunch is slightly longer with $\sigma_z = 21$ fs.

This is reflected in the emitted synchrotron radiation spectra of the beam distributions, which are shown in Fig. 4.21.

The BO results show a smooth spectral power until the cutoff which is limited by the bunch length. In comparison, the spectrum intensity of the LHS result drops at a lower

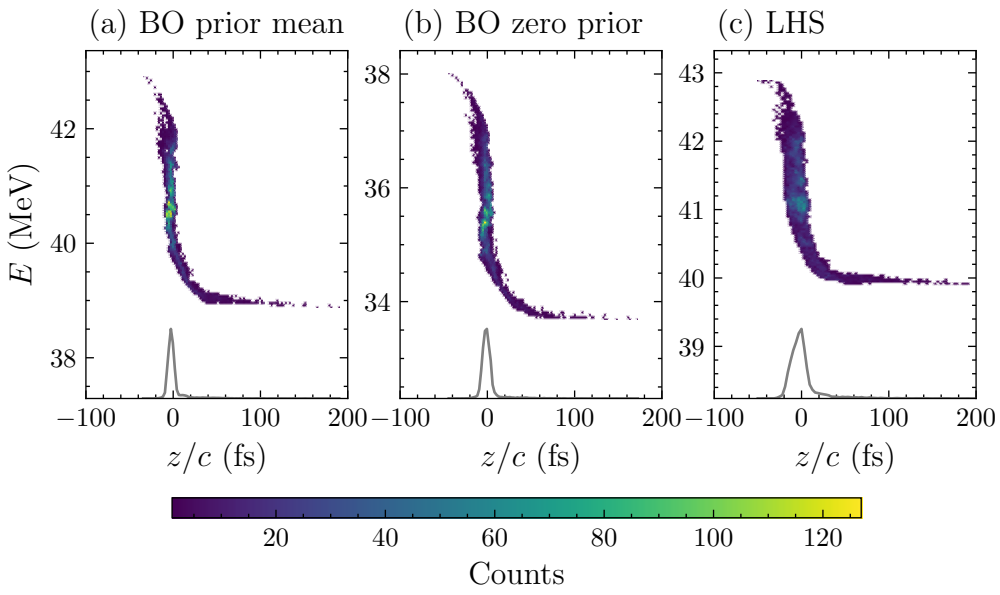


Figure 4.20.: Longitudinal phase spaces of the final beam from different methods: (a) BO using Cheetah as the prior mean function, (b) BO with zero prior mean, and (c) LHS method. The projection onto the z axis is shown at the bottom.

frequency due to a longer bunch length, but it exhibits substructures extending to higher frequencies. Despite having a slightly longer bunch length than the one from BO with zero prior mean, the BO with prior mean result shows a higher radiation power overall, as the bunch has a higher energy. This is consistent with the findings in Section 4.2. Lastly, the electric fields of the generated THz pulses are calculated and shown in Fig. 4.21 (b). It is visible that the shape of the THz pulses follows closely the shape of the charge density distributions of the bunches.

This example shows how Cheetah simulation can be used in combination with BO algorithms and further speed up the parameter optimization. Moreover, the same method can also be applied in the real world to guide the online tuning process of accelerators.

4.5. Summary Machine Learning Assisted Simulated Optimization

Physics simulations are essential components in the life cycle of a particle accelerator. The design and operation of future accelerators with increasing controllable parameters and tighter operation conditions pose rising challenges to physics simulations and optimization methods. In this chapter, the THz pulse generation at FLUTE with the CSR mechanism was studied as an example. By applying ML algorithms, accelerator settings for different bunch charge cases were found to produce ultrashort THz pulses, which are up to 5 times more intensive than the ones from design settings.

The presented usage of ML methods for particle accelerator simulations can be mainly summarized in three aspects. First, data-driven surrogate models like NNs can be trained

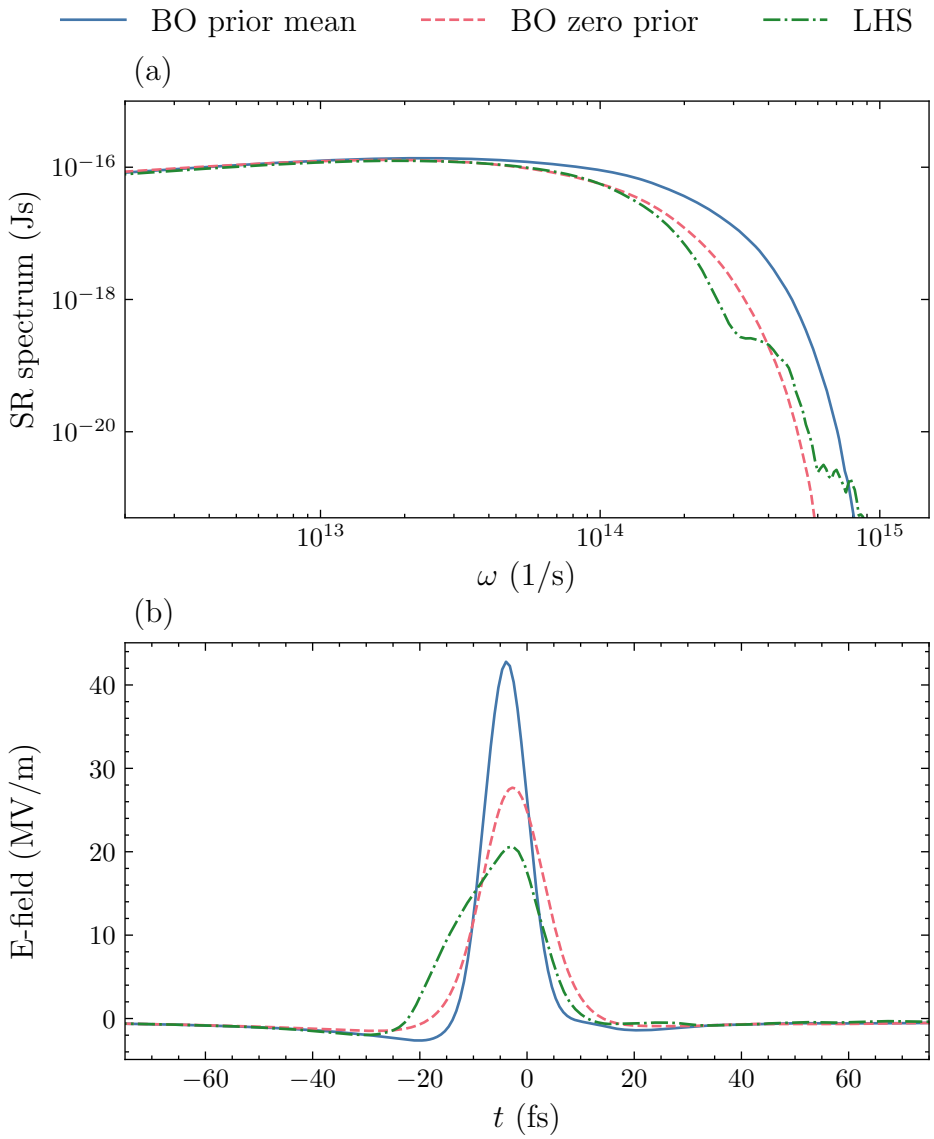


Figure 4.21.: Resulting synchrotron radiation (SR) spectra and the electric fields of the CSR pulses generated by the electron bunches with different optimization methods, including BO with Cheetah as prior mean, BO with zero prior mean, and the LHS.

from existing simulation data as a replacement for the time-consuming simulations. They are cheap to evaluate and can provide fast and high-resolution predictions on the beam parameters, reducing the need to repeatedly run simulations with similar parameter settings. They can also be utilized to provide insight into the dependency of the beam parameters on input accelerator settings. Second, intelligent algorithms like BO can be used to efficiently solve black-box optimization tasks in simulations. Utilizing the parallel execution in computing clusters, the performance of BO exceeds the conventional genetic algorithms. Third, there is an emerging generation of accelerator simulation codes with

support for differentiation and ML methods. The Cheetah beam dynamics simulation code developed in this dissertation is one prominent example. The AD feature allows direct gradient-descent optimization of the accelerator parameters such as RF settings and magnet strengths to produce the desired beam. Compared to the black-box optimization algorithms like BO, the differentiable simulations providing gradient information allow more efficient optimization that can be effortlessly scaled up to thousands of input parameters. At present, they have implemented only a small amount of physical effects like collective effects. This inevitably limits the number of simulation tasks that can benefit from the differentiable simulation. Nevertheless, it can be foreseen that the applicable tasks will grow with the future development of the differentiable simulation codes.

Most importantly, the presented ML applications can also be combined to design more tailored methods for individual accelerator tasks. For example, both the NN surrogate model and the Cheetah simulation can be used as a physics-informed prior mean for the GP modeling in the BO algorithms to improve the sample efficiency. This allows BO to be applied to more complex tasks with a higher number of input parameters. NN models can also be trained to model the computationally intensive effects and integrated as additional elements in the particle tracking routines. This approach increases the fidelity of the Cheetah simulation while maintaining the execution speed.

While all the applications presented in this chapter are based on FLUTE with a few adjustable parameters, the approaches used here can be generalized to any simulation task for particle accelerators. The advantages of ML methods are expected to be more prominent when traditional approaches become infeasible, for example when scaling to hundreds of input parameters or dealing with computationally intensive simulations taking hours to run.

Last but not least, with the increasing capability of computing hardware and ML algorithm development, it is foreseeable that the physics simulation will be more integrated and benefit the operation of particle accelerators. Trained surrogate models can be used as virtual diagnostics to provide high-resolution and non-destructive information on the beam parameters, saving the beam time required for setup and measurement during operation. Fast-executing simulations can also be directly run in situ to guide the operation with the help of on-site computing clusters.

5. Photo-Injector Laser Pulse Shaping

At linear accelerators, the characteristics of the electron beam are largely governed by the initial beam distribution in the injector section. For example, the normalized emittance can at best be preserved or grow slowly along the accelerator. To extend the operation limit and increase the radiation output at free electron laser (FEL) facilities, it is essential to optimize the initial electron bunch distribution. For photo-injector-based accelerators, this can only be achieved via tailored control of the laser pulses. This chapter presents the photo-injector laser shaping experiments with spatial light modulator (SLM) devices at the far-infrared linac and test experiment (FLUTE). The proof-of-principle studies demonstrate that machine learning (ML) techniques can facilitate high-dimensional, adaptive control towards a tailored laser shaping. This highlights a scenario where ML methods can broaden the range of beam parameters and enable new operation modes for existing and future accelerators, for example expanding the limits on bunch emittance and bunch charge. Section 5.1 shortly motivates the importance of laser shaping for photo-injector-based linear accelerators. Section 5.2 discusses how the laser pulse is modulated using SLMs and introduces the photo-injector laser at FLUTE. A proof-of-principle transverse laser shaping setup using a neural network (NN)-assisted algorithm is demonstrated in Section 5.3. Initial tests of transverse modulation for the FLUTE photo-injector laser are presented in Section 5.4 and the outlook for a full 3D laser shaping setup at FLUTE is discussed in Section 5.5. Parts of the results presented in this chapter have previously been published in [129, 130].

5.1. Motivation for Photo-injector Drive Laser Shaping

A three-dimensional (3D) uniform ellipsoidal distribution produces only linear space charge force fields, which can be further compensated by tuning the solenoid magnetic field [131, 132], resulting in a minimal emittance growth in the photo-injector gun. As such, they are the desired initial bunch distribution for applications that require high charge and small emittance electron bunches, for example, high-brightness FELs, ultra-fast electron diffraction experiments, and energy recovery linacs [133, 134, 135]. In linear accelerators with an RF photo-injector gun, the electron bunch characteristics are largely determined via the drive laser pulses, and forming a uniform ellipsoidal distribution after the bunch generation can only be achieved in special operation modes and with limited bunch charge. A promising alternative is to generate such electron distribution directly at the photo-injector by using a uniform ellipsoidal laser pulse. Such a 3D uniform laser profile is obtained by consecutive transverse (spatial) and longitudinal (temporal) shaping setups. Nevertheless, spatial-temporal laser shaping requires specialized setups and can not

be accomplished with standard optical elements, making it less commonly implemented in accelerator facilities.

For the transverse profile, the hard-edged uniform distribution, also known as a *flattop* distribution, is hard to realize experimentally. In practice, the transverse flattop laser is often approximated by a Gaussian laser pulse and using an iris to cut off the edge, resulting in a so-called truncated Gaussian distribution. The normalized radial charge density $\rho(x, y)$ of these three distributions can be expressed as

$$\begin{aligned}
\rho_{\text{uniform}}(x, y) &= \frac{1}{\pi r^2}, && x^2 + y^2 \leq r^2 \\
\rho_{\text{Gaussian}}(x, y) &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right), && \\
\rho_{\text{trunc. Gaussian}}(x, y) &= \frac{1}{2\pi\sigma^2 \cdot \operatorname{erf}\left(\frac{r}{\sqrt{2}\sigma}\right)} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right), && x^2 + y^2 \leq r^2
\end{aligned} \tag{5.1}$$

The transverse space charge (SC) electric field E_x can be calculated as

$$E_x(x, y, z) = \frac{1}{4\pi\epsilon_0} \frac{\partial}{\partial x} \iiint \frac{\rho(x', y', z')}{|\mathbf{r} - \mathbf{r}'|} dx' dy' dz', \tag{5.2}$$

where $\boldsymbol{r} = (x, y, z)$ is the position vector.

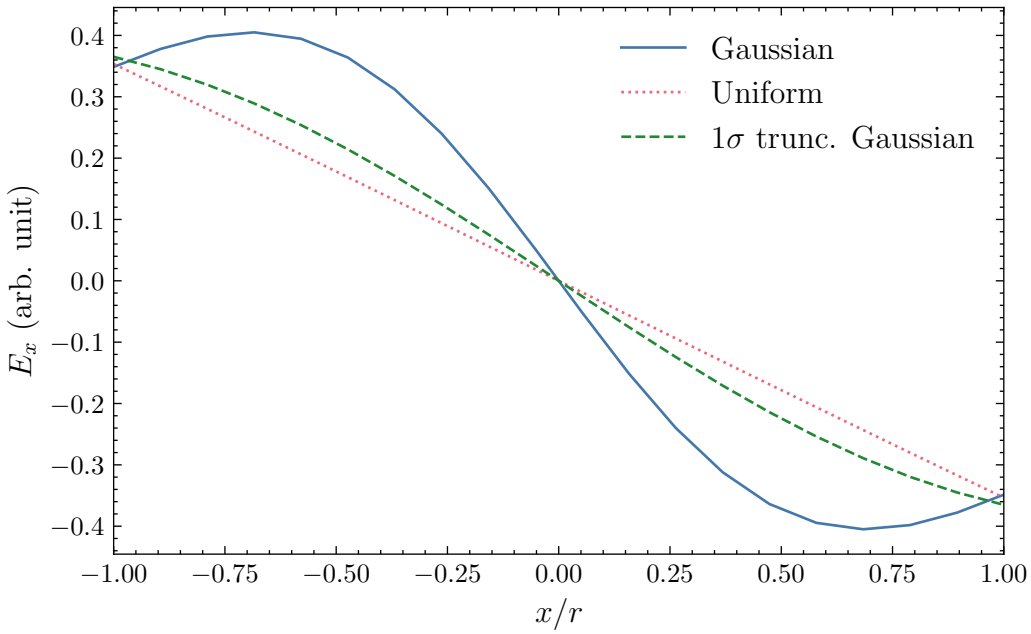


Figure 5.1.: Illustration of transverse space charge fields for different radial beam distributions: Gaussian, uniform, and 1σ truncated Gaussian distribution.

The transverse SC fields for the different radial charge distributions are visualized in Fig. 5.1, showing a Gaussian bunch, a uniform bunch, and a 1σ truncated Gaussian bunch ($r = \sigma$). The calculations are based on [136], assuming a longitudinal-Gaussian cigar-like

bunch, i.e. with a bunch length much larger than the transverse beam size $\sigma_x/\sigma_z \ll 1$. It is visible that the uniform distribution shows a more linear SC field compared to the 1σ truncated Gaussian distribution. In the case of a pancake-like bunch $\sigma_x/\sigma_z \gg 1$, the 1σ truncated Gaussian distribution behaves better than the uniform distribution [137]. Nevertheless, both uniform and truncated Gaussian distributions clearly outperform the Gaussian distribution in terms of linearity in the SC field.

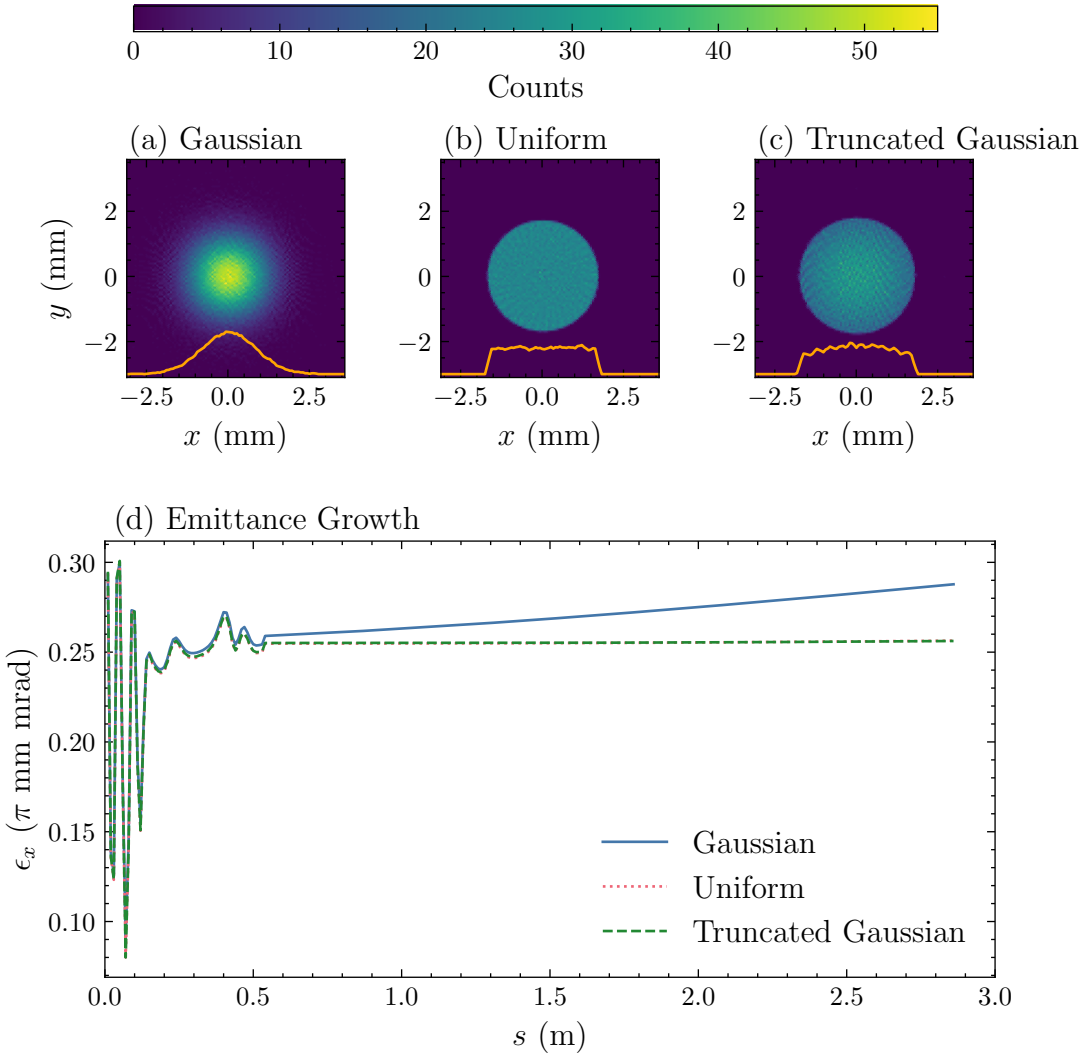


Figure 5.2.: Transverse emittance growth of 1 pC electron beam in the FLUTE low energy section for three different initial radial distributions: Gaussian, 1σ-truncated Gaussian, and uniform distribution.

To further illustrate this effect, the emittance growth of the three distributions is studied using the RF photo-injector gun at FLUTE. The transverse beam images of the simulated particle distributions are shown in Fig. 5.2, each containing 50,000 macroparticles and a bunch charge of 1 pC. The three bunches are tracked in ASTRA using the same accelerator settings through the RF gun and solenoid magnet until 2.86 m, before the entrance of the linac, with an energy of 5 MeV. As can be seen in Fig. 5.2, the emittance increase is

stronger for the Gaussian bunch due to the space charge forces. The uniform and the 1σ truncated Gaussian bunches showed very similar behavior and in total about a 11% emittance reduction compared to the Gaussian bunch. This difference becomes even larger for higher bunch charges. In FELs, for example, low emittances are mandatory to reach enough transverse overlap of the electron bunch and the emitted optical beam and thus achieve strong coherent radiation [138]. Similar considerations exist also in the longitudinal direction. To reach the ideal 3D uniform electron bunch, both transverse and longitudinal modulations are required for the photo-injector laser. This is also referred to as the *spatio-temporal* shaping. Overall, shaping the photo-injector laser is an effective method to provide tailored electron bunch distributions and minimize the emittance growth. This effectively pushes the operation limit and increases the radiation output of the FEL facilities.

5.2. Laser Shaping with Spatial Light Modulators

Several non-adaptive approaches for laser shaping have been successfully demonstrated at particle accelerators and could produce certain pre-determined laser profiles with high precision. These include employing optical gratings and masks, temporal shaping with stacking and superposition of the pulses, and a novel technique to incorporate the shaping process in the up-conversion into ultraviolet (UV) [133, 139, 140, 141]. Nevertheless, they either require manufacturing new optical elements for different modulations or are inherently limited in the degrees of freedom of the generated laser profiles. Having the possibility to shape the laser on demand is advantageous in the operation of future accelerators. It allows the modulation to be adaptively controlled, which means the modulation pattern can be iteratively refined to improve the quality of the electron bunch profile or to compensate for drifts of other components in the laser path. In particular, they could also correct for the varying quantum efficiency (QE) of the photocathode surface due to the degradation over a longer period and ensure the generation of the desired charge distribution with high fidelity [142].

A promising adaptive method for laser shaping is by using an SLM. It is a programmable device that allows arbitrary modulation patterns, also known as computer generated holograms (CGHs). It consists of pixelated cells filled with liquid crystal (LC). One type of SLM is electrically-addressed, which means that the applied modulation can be controlled electronically through voltages. When subjected to a voltage, the LC molecules rotate and lead to spatial modulation in either phase, amplitude, or both. This chapter mainly focuses on the phase-only type of SLM, which employs the liquid crystal on silicon (LCoS) technique and operates in the reflective mode. Figure 5.3 shows schematically the working principle of such a device. It is composed of multiple layers including cover glass, transparent electrodes, LC molecules, reflective pixelated electrodes, and a silicon substrate. The orientation of the LC molecules at each pixel is controlled by the voltages applied between the electrodes. As a result, the reflected light undergoes a phase modulation spatially, changing its wavefront compared to the incoming light. [143]

SLMs have been successfully applied to the spatial-temporal shaping of photo-injector lasers. In [142], arbitrary laser and electron profiles were generated using an SLM in

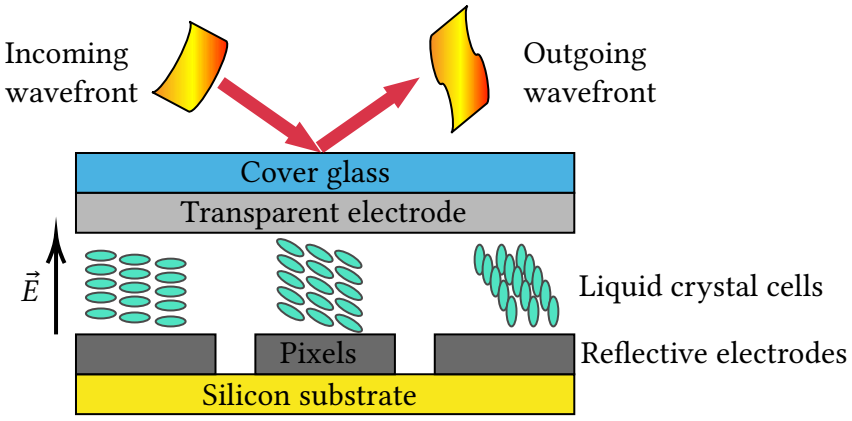


Figure 5.3.: Schematic working principle of an LCoS SLM. The SLM consists of layers of cover glass, transparent electrodes, liquid crystal cells, reflective pixelated electrodes, and a silicon substrate. The electrical fields between the electrodes control the orientation of the LC molecules at each pixel. The reflected light receives the phase modulation and contains a different wavefront than the incoming light.

combination with a cathode with a direct current (DC) voltage. At the Photo Injector Test Facility at DESY in Zeuthen (PITZ), the laser shaping capabilities of SLMs were demonstrated with multiple setups. One of them used two SLMs with an infrared (IR) laser to generate the 3D uniform ellipsoidal laser distributions. A recent work used SLMs with the green laser at $\lambda = 515\,\text{nm}$ and directly applying the modulated laser to the CsK_2Sb cathode for electron generation [144, 145, 146, 147].

It needs to be noted that, in addition to the electrically-addressed SLMs discussed here, other programmable or adaptive devices for laser shaping also exist. One notable example is the acousto-optic modulator (AOM) device [148, 149], which is mostly used in temporal shaping. AOMs demonstrate comparable performance to SLMs in terms of fidelity of the modulated pulse but often have lower transmission efficiencies. The main advantage of using an AOM is that they can operate directly in the UV regime and handle very high laser power, with the gas-based device being able to modulate GW-scale lasers [150]. It has been shown that AOMs could be used for spatial-temporal shaping of the photo-injector laser pulses [151]. A recent work [152] also demonstrated successful temporal shaping of a photo-injector laser, where the control is also achieved with the help of a NN, similar to the work presented in the following sections. Another approach uses deformable mirrors equipped with electrostatic piezo actuators [153]. They also have the advantage of operating with short-wavelength and high-power laser than the LC-based SLMs. However, deformable mirrors are limited in their actuators' degrees of freedom (commonly $d < 100$), and subsequently the fidelity of the modulated laser distribution [142]. A more extensive review on bunch shaping in electron linear accelerators can be found in [154].

5.2.1. Implementing the Laser Modulation Setup at FLUTE

The FLUTE accelerator uses a photo-injector driven by an 800 nm Titanium:Sapphire (Ti:Sa) laser. Figure 5.4 shows the simplified schematics of the laser transportation path. The 800 nm laser pulses are first generated in the clean room, at 1 kHz. The laser is transported with the uncompressed pulse length of 200 ps to reduce the dispersive effects in the long transportation path of ~ 35 m to the photocathode.

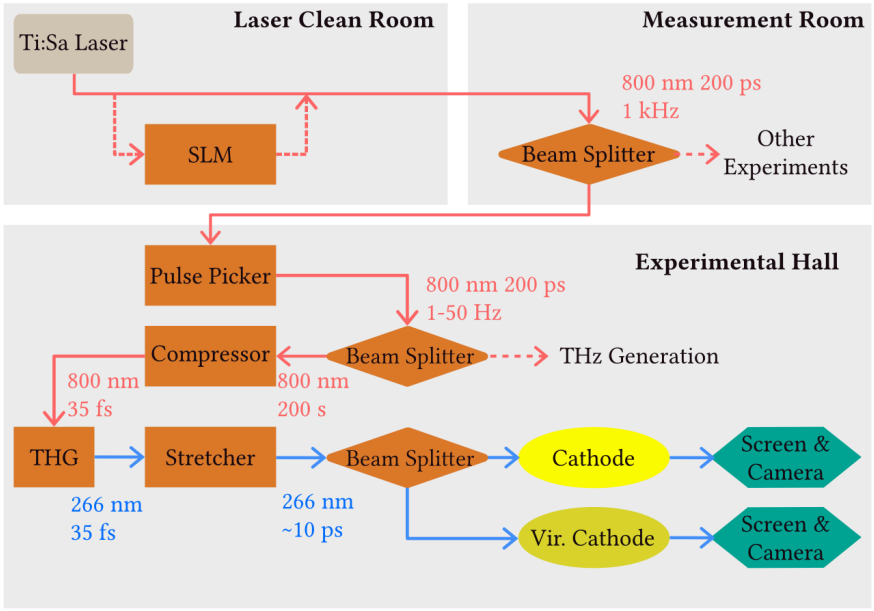


Figure 5.4.: Schematic of the FLUTE laser path layout. The diagram is simplified and only shows components that are relevant to the study in this dissertation.

The laser pulse is further guided through the measurement room into the experimental hall, where the accelerator FLUTE is located. A pulse picker reduces the pulse repetition rate to 1 - 50 Hz, depending on the selected operation mode. Subsequently, the laser is split using a variable attenuator so that a part of the intensity can be used for a laser-based THz generation setup [155]. The laser pulse is then compressed down to 35 fs pulse length and converted to 266 nm through the third harmonic generation (THG) process. The UV pulse is then stretched to ~ 10 ps using quartz rods of various lengths. The pulse is split before entering the vacuum chamber and focused onto the copper cathode for electron generation. The reflected laser from the cathode is imaged on a screen with a camera to capture the transverse profile of the laser pulse. A part of the laser intensity is projected onto a virtual cathode, which is placed at the same distance as the real cathode for additional diagnostics.

As current LC-based SLM cannot work in the UV range, the SLM needs to be positioned before the UV conversion and modulate the IR laser. In addition, due to the space

constraints, the SLM setup could not be placed in the FLUTE experimental hall. This is very challenging for laser shaping, as the modulation quality can degrade significantly due to the long optical transportation and non-linear processes such as the THG. Two SLM setups were built and investigated at FLUTE, including a standalone test setup in the measurement room and an integrated setup in the laser clean room.

5.2.2. Controlling the Spatial Light Modulator

Algorithm 1 Gerchberg-Saxton algorithm for phase-only laser shaping

Require: Target laser amplitude T at the image plane, incoming laser amplitude E at the Fourier plane, $T, E : \mathbb{R}^2 \to \mathbb{R}$
Initialize phase at the Fourier plane $\phi^{(0)}$ randomly
$A_1 \leftarrow E * \exp\{(i\phi^{(0)})\}$ ▷ Represent the wavefront as a complex-valued function
for j=1, …, n **do**
$A_2 \leftarrow \mathcal{F}(A_1)$ ▷ Propagate the wavefront to the image plane
$A_3 \leftarrow T * \exp\{(i \cdot \text{Phase}(A_2))\}$ ▷ Replace the amplitude with the target T
$A_4 \leftarrow \mathcal{F}^{-1}(A_3)$ ▷ Propagate the wavefront back to the Fourier plane
$A_1 \leftarrow E * \exp\{(i \cdot \text{Phase}(A_4))\}$ ▷ Replace the amplitude with the incoming laser amplitude E
end for
Return $\phi^{(n)} = \text{Phase}(A)$

In this chapter, the phase-only SLMs are used for the laser shaping setups. Despite their names, they are versatile devices as they can provide both phase and amplitude control [156] with proper setups. They are more efficient than the amplitude-only devices which block part of the incident light. The setups use a commercially available phase-only LCoS-SLM from Hamamatsu (model X13138-02). It has a specified wavelength of (800 ± 50) nm, a pixel number of 1024×1272, and a frame rate of 60 Hz. The SLM is connected to a laptop as a second display monitor. The voltage and hence the phase modulation patterns, are set using grayscale images in 8-bit encoding, i.e. the phase shift of $[0, 2\pi]$ is discretely mapped to gray levels of $[0, 255]$. In total, it adds up to $256^{(10^6)}$ degrees of freedom.

In phase-only SLMs, only the phases of the light are modulated. The modulated laser further propagates along the optical path and gets measured on an image plane. Due to the interference, the final image will also contain an amplitude modulation. For each desired amplitude pattern, the corresponding modulation pattern on the SLM needs to be calculated. Due to the high degrees of freedom in the modulation, dedicated algorithms are developed to perform the CGH calculation efficiently. One of the most widely used algorithms for calculating the CGH for an arbitrary target image is the Gerchberg-Saxton (GS) algorithm, originally designed as a phase-retrieval method. The CGH task can be described as follows: given the desired laser pulse intensity at the *image plane* $I(x, y) = |T(x, y)|^2$, corresponding to an amplitude of $T(x, y)$, find the modulation pattern $\phi(x, y) \in [0, 2\pi)$ at the *diffraction plane*. When using the phase-only modulation, a lens is commonly used to reconstruct

the image in practice so that the light propagation can be described using the far-field approximation, i.e. the Fraunhofer approximation. In such a case, the wavefronts at these two planes are related by the Fourier transform (FT), and the diffraction plane is also referred to as the *Fourier plane*. The GS algorithm utilizes this idea to determine the phase modulation pattern ϕ by iterative FTs and constraining the amplitudes at these two planes.

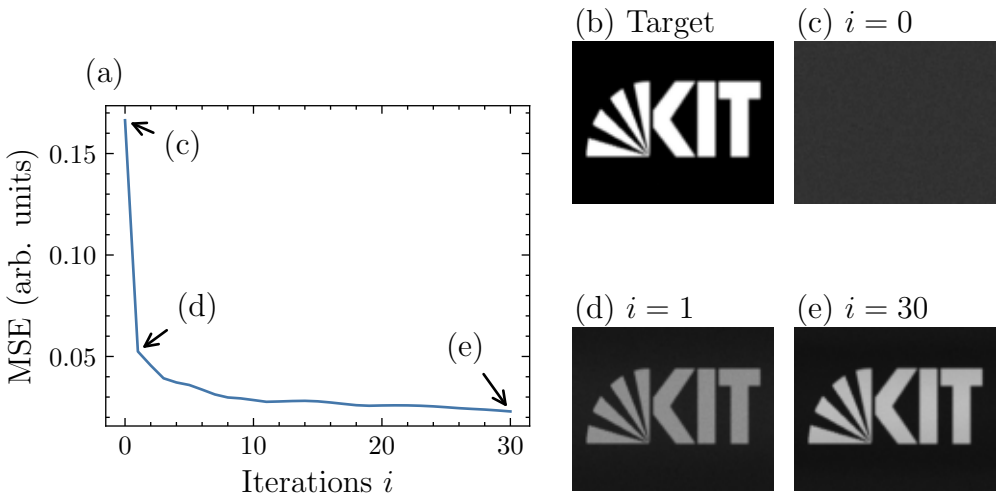


Figure 5.5.: Progress of Gerchberg-Saxton algorithm. (a) The mean square error of the reconstructed image using the GS algorithm compared to the target image. Right: (b) the target image containing a KIT logo, and the reconstructed images corresponding to (c) the randomly initialized phase, (d) at iteration $i = 1$, and (e) at iteration $i = 30$.

The progress of the GS algorithm in this setting is outlined in Algorithm 1 with an illustration in Fig. 5.5. The target image is a 1024 grayscale image of the KIT logo. In the initial step (c), the modulation pattern is randomly generated, and no meaningful image can be reconstructed. Already after one iteration (d), the target intensity pattern starts appearing, although some noise is still visible. The mean squared error (MSE) between the target and the reconstructed image continues to decrease and converges in around 30 steps. The final image (e) could accurately reproduce the target image. The above computation of 30 iterations takes about 1 s on a normal laptop, mostly attributed to the $O(n \log n)$ complexity of fast Fourier transforms (FFTs). For lower-resolution images, this comes down to the millisecond range. Consequently, this method is sufficiently responsive for most linear accelerators operating at low repetition rates and commercially available SLMs with 60 Hz frame rate.

In the context of transverse photo-injector laser modulation with an SLM, which is further investigated below, the intensity $I(x, y)$ at the image plane corresponds to the laser pulse intensity at the cathode surface, which directly influences the initial radial distribution of the generated electron bunch. The SLM is put in the Fourier plane and the retrieved phase $\phi(x, y)$ is the modulation pattern to be set in the SLM. As SLMs are pixelated devices, the modulation pattern $\phi(x, y)$ is a matrix with the dimension of pixels, and the FT is achieved via FFT.

5.3. Demonstration of Transverse Laser Modulation

As mentioned above, one drawback of using an SLM for laser shaping is the operating wavelength, commonly ranging from visible to infrared, as the LC cells are quickly damaged by UV lasers. Therefore, the SLM has to be placed before the UV conversion for a photoinjector laser. This inevitably introduces multiple optical elements between the SLM and the cathode plane, affecting the quality of the modulated laser. It is expected that ML methods, specifically NNs, can be used to compensate for the errors and improve the modulation quality. To study the feasibility of SLM control using NNs, a standalone setup for the transverse laser pulse modulation was first built in the measurement room, as a proof-of-principle experiment.

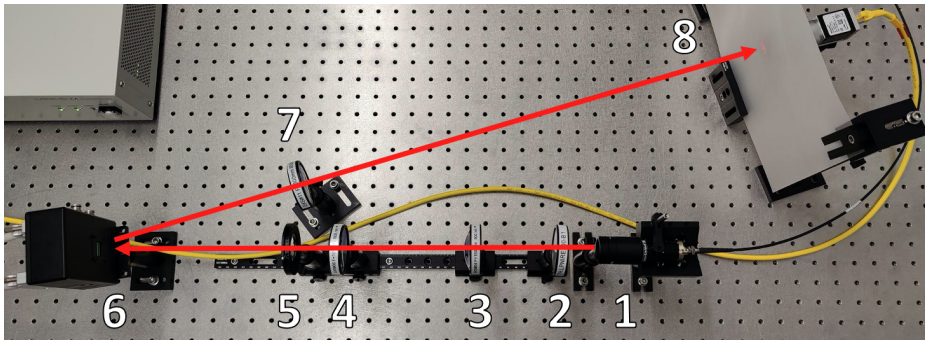


Figure 5.6.: **Test setup for transverse laser manipulation**. (1) The red line depicts the path of the 638 nm laser. (2) Horizontal polarization filter. (3, 4) $f = 150$ mm, $f = 100$ mm lens. (5) Iris. (6) SLM. (7) $f = 500$ mm lens. (8) Screen with the camera behind. [129]

The setup for transverse laser manipulation is depicted in Fig. 5.6. A continuous wave (CW) test laser with a wavelength of $\lambda = 638$ nm in the visible range is used in this proof-of-principle setup, enabling a fast iteration process, as it facilitates easier alignment and result visualization. The test laser is first collimated and passes through a polarization filter, generating horizontally linearly polarized light required by the SLM. Next, a telescope consisting of two lenses expands the laser to utilize the SLM's active area fully. The laser is subsequently phase-modulated and reflected by the SLM before passing through the focusing lens. Lastly, the image is projected onto a screen and captured by a camera [129].

5.3.1. Laser Modulation Correction using Convolutional Neural Network

To generate a laser pulse with an intensity pattern I_{target}, a CGH (phase pattern) $\phi = \text{GS}(I_{\text{target}})$ is calculated using the GS algorithm and displayed on the SLM. The pulse modulated by the SLM further propagates through the optical path $\mathcal{P}$ to the image plane and is captured with an intensity pattern I. Thus, the forward mapping $\mathcal{M}$ of the intensity patterns can be written as a composite function of GS and $\mathcal{P}$

$$I = \mathcal{M}(I_{\text{target}}) = \mathcal{P}(\phi) = \mathcal{P} \circ \text{GS}(I_{\text{target}}). \tag{5.3}$$

The optical propagation $\mathcal{P}$ needs to be accurately modeled and incorporated into the back-and-forth propagation steps in the GS algorithm to reproduce the I_{target} on the image plane with high quality. However, this is not always feasible due to additional elements present in the laser path and the mismatch between optical simulation and the real-world setup. NNs are a promising method to model an unknown propagation $\mathcal{P}$ thanks to their capability as general function approximators. Previous work [157] demonstrated that a convolutional neural network (CNN) can be used to learn the inverse process $\mathcal{P}^{-1}$, i.e. predicting the modulation pattern ϕ to be applied on the SLM from any given target image.

In our setup, however, learning the inverse process $\mathcal{P}^{-1}$ directly is difficult and undesired, as it adds complexity to the training process by requiring the network to learn the FT, because the image plane is placed in the far-field to the SLM. Therefore, an alternative strategy is proposed: train a CNN to directly learn the inverse process $\mathcal{M}^{-1}$ between the target image I_{target} and the produced image I at the image plane $\mathcal{M}^{-1}(I) = I_{\text{target}}$.

The network structure used here is inspired by the U-Net [158]. It can map an input image to an output image with the same dimension and perform modifications on the image while preserving the spatial information in the input. The U-Net structure consists of multiple downsampling blocks, upsampling blocks, and skip-connections between the same level of down- and upsampling blocks. The downsampling block contains convolution layers and pooling layers to extract the spatial features in the image, reducing the image dimension and increasing the channel number. Each channel in the CNN can be viewed as a certain feature in the image. The upsampling blocks utilize the transpose convolution layers to increase the dimension back to the input image size sequentially. The skip-connections preserve the spatial information in the images and help the upsampling blocks to locate the features in the image. This structure suits the transverse setup here as the distortion in the optical path is small so the target image and the modulated image are expected to be strongly correlated.

Figure 5.7 shows the network structure used in this study. The definitions of the network layers and blocks are as follows: BN is a 2D batch normalization layer [159]; rectified linear unit (ReLU) is the rectified linear unit activation function; $\text{Conv}(K, S)$ is the 2D convolution layer with kernel size K and stride S; $\text{TConv}(K, S)$ is the transposed 2D convolution layer. The ConvSame block consists of a $\text{Conv}(3, 1)$ and a BN layer, which changes the number of convolution channels but preserves the pixel count. It is used for processing the input image and predicting the output image. The "Down" block with $\text{Conv}(3, 2)$ reduces the image resolution and increases the number of channels C, whereas the "Up" block does the opposite. The network is chosen to take grayscale input images with a pixel count of $N^2 = 32^2$, mapping to an initial count of $C_i = 16$ convolutional channels. The downsampling blocks and upsampling blocks of the same depth are skip-connected to aid the training and preserve the spatial information. It includes four downsampling blocks, ultimately expanding to 256 channels at the deepest layer of the network.

The dataset for training comprised images from the widely-used MNIST database, which contains handwritten digits [160]. These images, originally at a resolution of 28^2, are zero-padded to 32^2 to fit the network's input size. Figure 5.8 illustrates the process of training data generation. The original image (a) I_{target} is transformed into a CGH, i.e. an image consisting of phase patterns, using the GS algorithm. The phase pattern (b) $\text{GS}(I_{\text{target}})$ is then displayed on the SLM. The incident laser is modulated by the SLM and

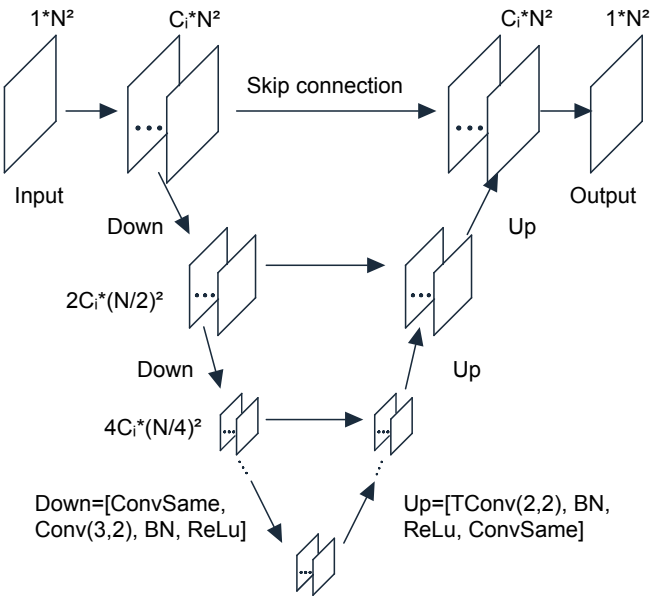


Figure 5.7.: **Structure of the CNN**. The network consists of sequential downsampling and upsampling blocks, which are skip-connected. C: numbers of the convolution channels; N^2: resolution of the image; BN: Batch normalization; ReLu: Rectified linear unit activation. [129]

further propagates onto the screen. The projected screen image (c) is captured using a camera. Subsequently, the camera image is processed by cropping to region of interest (ROI) and re-scaling to match the network's expected resolution of 32^2, resulting in the final image (d) $I = \mathcal{M}(I_{\text{target}})$. It is worth noting the bright spot in the middle of the image, also known as the zeroth order diffraction (ZOD). This is due to the unmodulated light that does not interact with the CGH, resulting primarily from dead areas between pixels on the SLM surface or window reflections. The zeroth order should be less significant when working with the FLUTE driving laser with $\lambda = 800$ nm, which is inside the working range of the SLM. Nevertheless, this presents a challenging scenario for testing the capability of NN-based corrections to improve the laser shaping fidelity. In practice, the ZOD can be further reduced or entirely eliminated with various methods, e.g. by applying a grating pattern and only taking the first diffraction order or by adding a virtual focusing lens and using an iris to filter out the rest of the light [143].

The training dataset for the NN consists of 10,000 pairs of such 32^2 grayscale images taken from the experimental setup, as shown in Fig. 5.8(a, d). The network is then trained to learn the inverse mapping $I \mapsto I_{\text{target}}$ (d $\rightarrow$ a) by minimizing the loss function L. It is defined as the MSE between the NN output $I_w := \text{NN}(I|w)$ and target images I_{target}

$$L(w) = \frac{1}{N_x N_y} \sum_{x=1,\ldots,N_x} \sum_{y=1,\ldots,N_y} \sqrt{(I_w(x,y) - I_{\text{target}}(x,y)^2}, \tag{5.4}$$

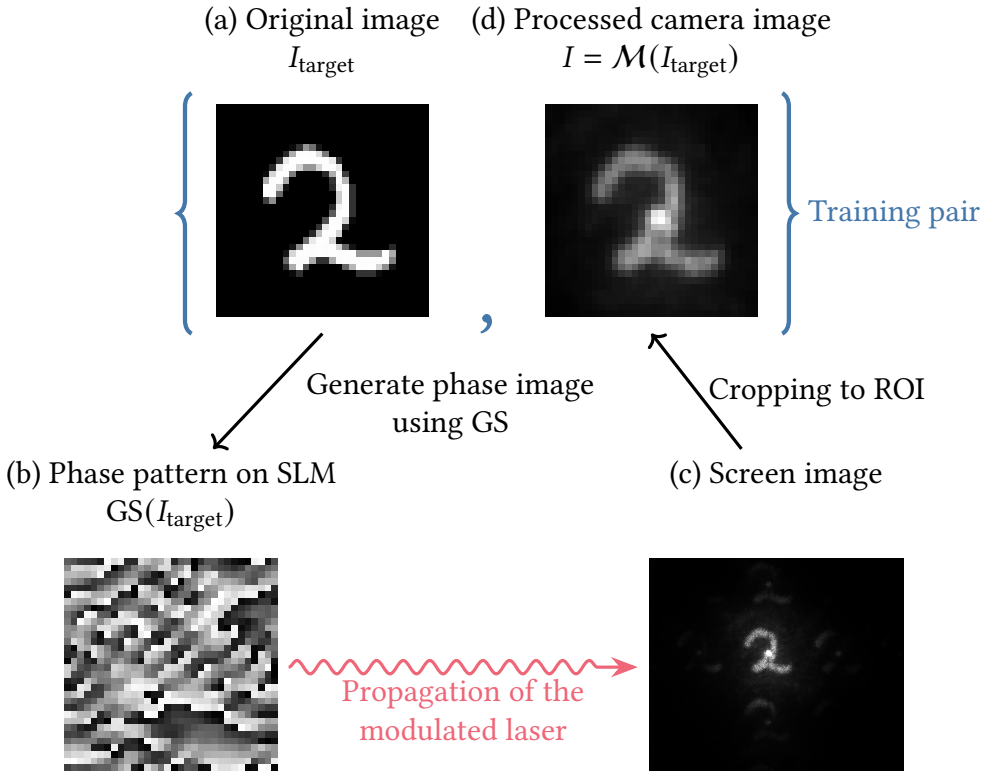


Figure 5.8.: Generation and preprocessing of the CNN training data. (a) The original image I_{target} to be displayed. (b) The phase pattern corresponding to the original image is computed using GS and displayed on the SLM. (c) The laser is modulated by the SLM and propagates onto the screen. A camera captures the screen image. (d) The camera image is processed by cropping to the region of interest (ROI) and re-scaling to the desired dimension. (a) and (d) are used as one data pair for the NN training.

where w are the NN weights. The training process utilizes the Adam optimizer [63] with an initial learning rate of 0.001. The training was conducted in mini-batches of 32, with the process capped at a maximum of 100 epochs to prevent overfitting.

After the training was completed, the CNN was employed to generate laser patterns for three different beam shapes, a digit, a Gaussian beam, and a flattop beam. The result of the Gaussian beam is shown in Fig. 5.9. The target image (a) is input into the trained CNN to predict an image that compensates for the distortions due to the light path setup. The CNN attempted to suppress the zeroth-order diffraction by predicting less intensity to be generated in the center of the image as shown in (d). The corresponding phase pattern (e) is calculated and encoded on the SLM device. The resulting laser pulse at the screen (f) is observed by the camera. As a comparison, the target image is directly transformed to a phase modulation pattern (b) using the GS algorithm, with the camera image of the modulated laser shown in (c). Comparing the two laser profiles shows a notable improvement in the one from the CNN output, with reduced zeroth-order diffraction visibility. This suggests that CNNs can effectively learn and compensate for systematic

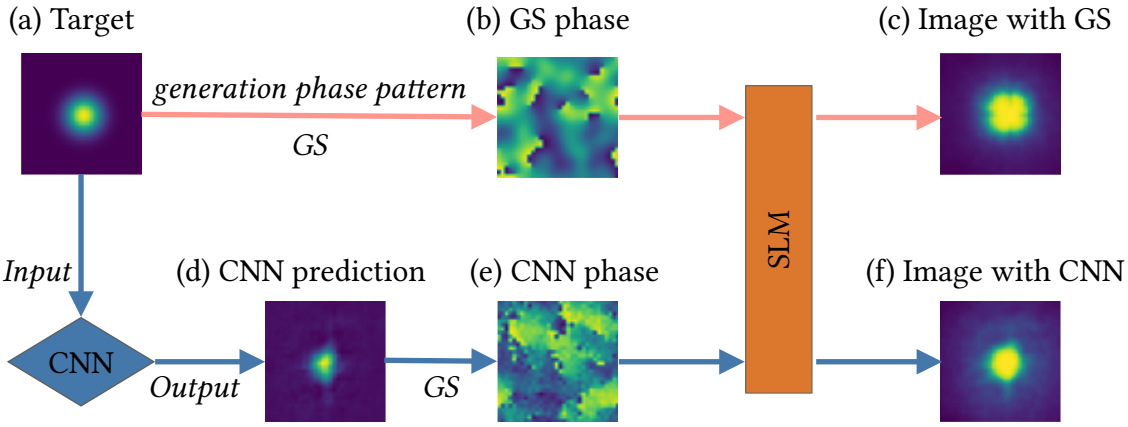


Figure 5.9.: Results of the transverse laser shaping using the trained CNN. (a) The target laser shape is a Gaussian distribution. (b) Direct generation of the phase modulation pattern using the GS algorithm. (c) The camera image corresponding to the GS output. (d) The CNN prediction of the target laser shape. (e) The corresponding phase pattern for the CNN prediction. (f) The camera image of the laser modulation using the CNN prediction. More results are given in Appendix A.3.

aberrations in the laser transportation path, and such an NN-enhanced approach is a promising method for improving the modulation quality of SLMs. More results on the CNN predictions at the transverse SLM setup are shown in Appendix A.3.

5.4. Transverse Spatial Light Modulator Setup for FLUTE Photo-Injector Laser

The SLM was then integrated into the beam path of the photo-injector laser at FLUTE [130]. The experimental setup is shown in Fig. 5.10. Two flip mirrors were installed to deflect the Ti:Sa laser $\lambda = 800$ nm onto the SLM, allowing effortless switching between the operation modes with and without the SLM. A $\lambda/2$-waveplate was installed (not shown in the image) to rotate the polarization of the laser pulse and ensure that the SLM correctly receives the incoming laser with horizontal linear polarization. The SLM was placed in the laser clean room due to the place constraints on the optical table close to the cathode, as shown in Fig. 5.4. This results in a long in-air transportation path of ~ 35 m after the modulation with SLM, including additional elements like the pulse picker, compressor, and the THG.

Before generating complex laser profiles, simple modulations were first performed to study their effects on the photo-injector laser at FLUTE. As an initial test, SLM was used

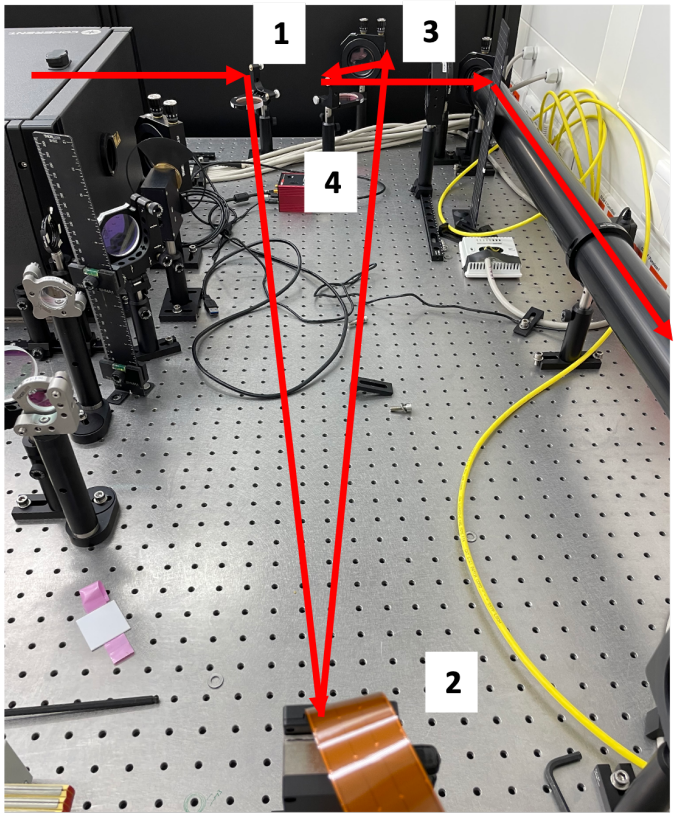


Figure 5.10.: Transverse laser modulation setup using SLM for FLUTE photo-injector laser. Two flip mirrors (1,4) allow easy switching between operation modes with and without the SLM. The setup is built in the laser clean room as shown in Fig. 5.4.

to split the incoming laser and create two concurrent laser pulses for the photo-injector, which was expected to generate two concurrent electron beams. Such beamlets can be further used to characterize the photo-injector at FLUTE, e.g. by performing efficient beam-based alignment of the solenoid. Such a modulation can be achieved by a *blazed grating*, which can efficiently reflect the light in a given diffraction order. In comparison, for other patterns such as the sinusoidal grating, the intensity of the light is distributed in many diffraction orders. For an incoming light normal to the grating, the outgoing light gets diffracted with an angle θ

$$\sin(\theta) = \frac{m\lambda}{d}, \tag{5.5}$$

where $m \in \mathbb{N}$ is the diffraction order, λ is the laser wavelength, and d is the length of the grating period. Such a diffraction grating can be electronically represented at the SLM by phase images consisting of sequential linear ramps. The working principle of the blazed grating and the corresponding phase image are illustrated in Fig. 5.11. In that case, the grating period is an integer multiple of the pixel width $d = n \cdot d_{\text{pixel}} = n \cdot 12.5\,\mu\text{m}$. The strongest diffraction grating that can be achieved by the SLM is 3-pixel wide, with phase values of $\{0, \pi, 2\pi\}$. This corresponds to a maximum deflection angle of $1.22°$ in the first

diffraction order $m = 1$ with the 800 nm laser. In general, a horizontal blazed grating on the SLM can be expressed as

$$g(x) = a \cdot \frac{x \bmod n}{n} \cdot 255, \tag{5.6}$$

where a is the amplitude of the grating pattern, controlling the fraction of light that will be deflected. For $a = 0$ the phase pattern becomes flat and no light is diffracted. This allows a programmatic selection of the amount of laser power in the diffraction order and the deflected angle.

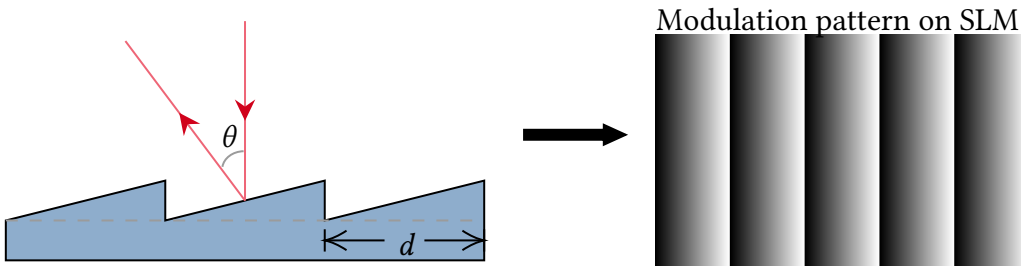


Figure 5.11.: Illustration of the blazed grating and the corresponding modulation pattern on SLM. The deflection angle θ of the laser can be controlled by varying the grating period d.

Here, $a = 0.6$ and $n = 260$ were chosen so that the zeroth and first diffraction order could be simultaneously observed by the camera with similar intensities and a separation of about 1 mm. Figure 5.12 shows the experimental results captured by three cameras, with the inset lines depicting the projected profiles in the horizontal and vertical directions. In the first column are the laser distributions observed at the virtual cathode. The second column shows the laser intensity observed directly at the cathode surface. The visible strips in the laser distributions are expected to be artifacts from the surface of the injection mirror, which reflects the laser onto the photocathode. The third column shows the transverse electron bunch profiles projected on a Yttrium Aluminum Garnet (YAG) screen positioned at 2.86 m downstream of the cathode, with a bunch charge of 5 pC.

First, the reference images are taken without the modulation, as shown in the first row. Dashed lines are added as a visual aid to the center of the laser and electron profiles. The intensities of the images are normalized to the reference images respectively. Next, with the grating pattern turned on, the laser spot on the cathode exhibited a division into two points horizontally, as expected. Since the driving laser power was kept constant, each of the pulses contained less intensity compared to the reference pulse after splitting. The observed intensity ratio between the first diffraction order and the zeroth order appeared to be higher than the specified 0.6/0.4 by the grating settings. This is partially attributed to the non-linear conversion efficiency in the THG, which is larger for higher laser power. With the modulated laser pulse, two co-propagating electron beams could be observed with a vertical displacement. This is due to the off-axis propagation of the bunch in the solenoid magnetic field. By scanning the strength of the solenoid, in addition to the focusing and defocusing of the bunches, the off-center second electron bunch could also be observed

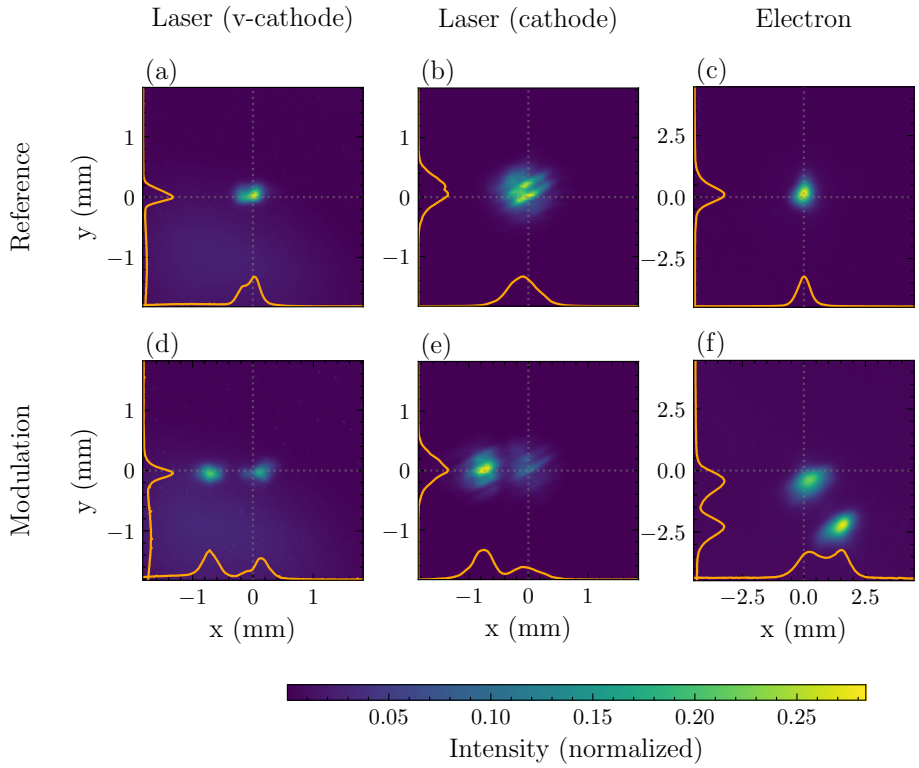


Figure 5.12.: Double electron bunches generated with SLM modulated laser. Camera images of the laser at the virtual cathode (a, d), the cathode (b, e), and the electron bunch on a diagnostic screen (c, f). Inset lines show the projected profiles onto horizontal and vertical axes. The top row shows the images with unmodulated laser and the bottom row (d, e, f) shows the images with SLM modulation. The laser pulse was modulated via a blazed-grating pattern so that part of the intensity was shifted to another position on the cathode, simultaneously generating a second electron bunch. The second electron bunch was rotated around the center bunch due to the solenoid magnetic field. The intensities of the images are normalized column-wise to the reference images respectively. Figure adapted from [130].

rotating around the centered original bunch. Overall, it successfully demonstrated that the SLM is effective for modulating the photo-injector laser at FLUTE.

5.5. Outlook for Full Laser Modulation at FLUTE

The next step is to generate arbitrary transverse shapes as achieved in the proof-of-principle setup in Section 5.3. However, this proves to be a non-trivial task in the current photo-injector laser setup at FLUTE. Due to the non-linear conversions and the long propagation path mentioned above, the intensity distribution of the final UV pulse is very sensitive to the applied modulation pattern at the SLM. Arbitrary phase patterns mostly lead to negligible conversion efficiencies and loss of the laser pulses. As a result,

no meaningful dataset could be constructed for a NN to learn the mapping between the modulation phase and the resulting laser profile.

In the following, the planned strategies are discussed that could address this challenge and establish a full laser shaping setup at FLUTE. First and foremost, the SLM should be positioned as close to the cathode as possible, minimizing the propagation length and the number of optical elements between the SLM and the photo-injector. The long in-air transportation path in the current setup not only limits the feasible modulation range but also degrades the quality of the modulated laser significantly, as additional noise is introduced by the air turbulence. Although the SLM still has to be put before the THG due to wavelength limits, placing the SLM after the pulse picker and compressor is expected to reduce the distortion significantly. However, this requires a new design of the laser path and rearrangement of all the optical elements in the optical table. Due to time constraints, this cannot be carried out within the scope of this dissertation but can be considered in the future.

In addition, more diagnostics on the modulated laser can benefit the learning process. The setups presented above only rely on the camera images, corresponding to the amplitude part $|A|$ of the complex light field $A = |A| \cdot \exp(i\phi)$, providing only incomplete information on the modulated light. A wavefront sensor could be employed to measure simultaneously the actual wavefront of the modulated laser, providing direct information on the phase ϕ.

In this setup with the FLUTE photo-injector laser, it is no longer possible to apply arbitrary phase patterns and observe the resulting laser image at the cathode plane. One limitation comes from the long distance of in-air transportation, which limits the wavefronts that can be transported along the laser path. In addition, laser pulses with arbitrarily modulated shapes have a poor conversion efficiency into UV, due to the non-linear efficiency by the THG. As a result, the laser pulses measured in the cathode plane often have low intensities and high signal-to-noise ratios. The NN cannot learn a meaningful mapping from the data pairs. To mitigate this issue, a pragmatic approach is to constrain the training datasets to include only the ones that can generate physically feasible laser pulses, which can be propagated through the laser path and have enough intensity after the THG conversion. This essentially also reduces the degrees of freedom of the task. These patterns should generate laser pulses that can survive the non-linear transformation and can be measured at the photocathode with sufficient intensities. In addition, decreasing the dimensionality of the actuators would significantly reduce the required number of training samples.

Zernike polynomials are a promising solution to both reduce the degrees of freedom and limit the patterns to physically feasible modulations. They form an orthogonal basis of the functions defined on a 2D unit disk, which is discussed further in Appendix A.4. In other words, any wavefront configuration can be decomposed into combinations of those Zernike polynomials, and the dimension reduction can be effectively achieved by discarding higher orders of the polynomials. They are particularly of interest as each Zernike term corresponds to a specific aberration mode, such as tilt, astigmatism, and spherical aberration. Figure 5.13 shows some early results of the laser pulses and electron bunches resulting from applying Zernike polynomials on the SLM. More images can be found in Fig. A.6.

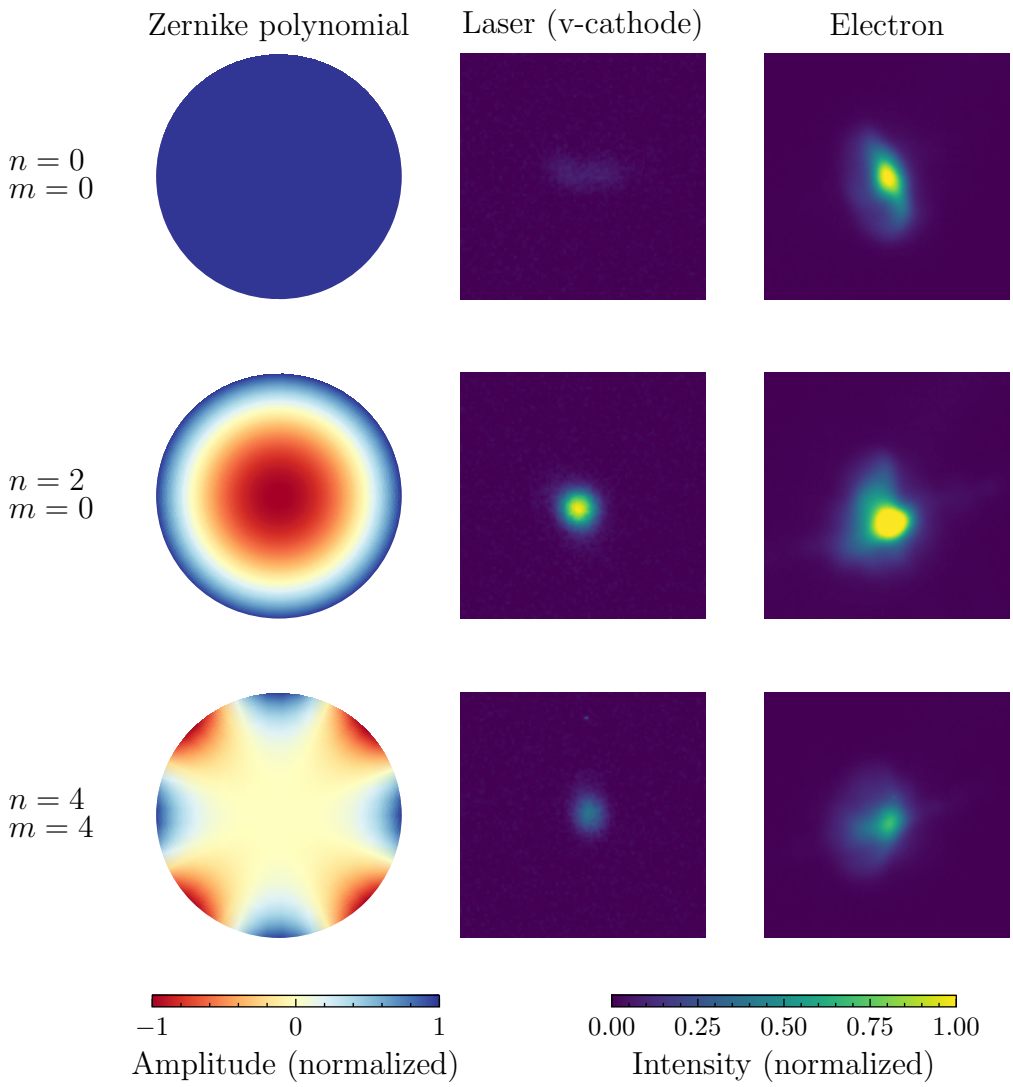


Figure 5.13.: Results of laser shaping using Zernike polynomials. The left column visualizes the Zernike terms applied on the SLM, the middle column shows the laser pulses observed at the virtual cathode, and the right column shows the generated electron bunches. Each row shows the modulation with a different Zernike term with their indices (n, m). The image intensities are normalized column-wise to the maximum pixel intensity.

It is clearly visible that with the $(n, m) = (2, 0)$ term in the second row, the laser pulse was more focused compared to the reference beam $(n, m) = (0, 0)$, resulting in higher intensities for both the laser and the electron bunch. Such results indicate that learning a mapping between the different Zernike polynomials and the resulting laser pulse on the photocathode is possible, especially with the help of additional information obtained by wavefront sensors. Consequently, an adaptive controller can be trained, for example using reinforcement learning, to control the Zernike coefficients to generate a desired laser pulse.

Lastly, although this chapter mostly focuses on transverse laser profile shaping, it is important to note that temporal profile shaping is also possible using SLM. A separate setup was built in the FLUTE laser clean room by Carl Sax and successfully shaped the laser to a spectral quasi-flattop profile [129]. This was achieved by using a grating to spatially separate the spectral components of a chirped laser pulse and image them onto the SLM. The displayed pattern on the SLM modulated the laser in the spectral domain. Afterward, the spectral components of the modulated laser pulse were reassembled using a grating which is mirrored to the first one. The spectral modulation corresponded directly to the temporal modulation when using a chirped laser. To achieve a spatial-temporal laser shaping setup for future experiments, the temporal and the transverse modulation setups could be combined. A full spatial-temporal laser shaping could be achieved with a minimum of two SLMs, where one for the $x - y$ plane and one for the z (λ) plane as described here. Alternatively, they could both be placed in the spectral modulation setup and control $x - z$ and $y - z$ plane respectively as described in [145].

5.6. Summary Machine Learning Enabled Laser Pulse Shaping

Laser shaping with programmable devices like SLMs is a promising strategy to enhance the performance of current and future photo-injector-based electron linear accelerators. It enables full control of the laser shape and thus the electron bunch profile, which was previously not achievable. They effectively extend the operation capabilities, for example by increasing the maximum charge and reducing the beam emittances. The ability to adaptively shape the laser with high-fidelity opens up avenues for novel operation modes such as providing tailored electron bunches on demand during operation without changes in hardware configurations or downtimes. Moreover, the utilization of shaped laser pulses extends beyond optimizing the operation of RF photo-injectors. It also facilitates electron shaping via other techniques, such as laser heater in the XFEL operation [161, 162] or direct electron bunch shaping with ponderomotive forces [163]. Although such programmable devices have very large degrees of freedom, common classical algorithms can only produce limited modulation shapes or can be very time-consuming. It is also non-trivial to deal with the degradation of the modulated laser due to non-linear processes in the transportation path. Machine learning techniques are promising to mitigate these issues and provide efficient, high-quality control for such SLM devices. Initial experiments in the transverse laser shaping, as presented above in the section, have demonstrated that CNNs can enable the modulation process and increase the quality of modulated laser pulses using SLMs. Using FLUTE as a test bench, this chapter showcases how an existing facility can integrate the SLM into the current laser setup. Several planned steps are discussed to achieve a full spatial-temporal shaping of the photo-injector drive laser at FLUTE. In summary, laser shaping represents a variant of how machine learning methods can contribute to enabling diverse operation modes and achieving tailored control of the electron bunches in future accelerators, advancing the frontiers of accelerator technologies.

6. Autonomous Online Accelerator Tuning

In particle accelerator operations, tuning the accelerator is a challenging task, but also crucial for it to realize its full design performance [164]. The measured signals are often noisy, both due to inherent random noise and uncertainties in the diagnostic devices. In addition, the accelerator system is not fully stationary and subject to drifts at different time scales, for example, due to the heating up of components, seasonal changes in temperatures, and changes in the upstream beam parameters [64, 164]. These time-dependent effects require either repeated parameter optimizations or some controller to keep track of the achieved optimal settings. To date, manual tuning remains to be frequently employed in most accelerator facilities. Nevertheless, the manual tuning is usually slow as humans can only operate one actuator at a time. The performance of manual tuning also highly depends on the operator's experience and the state of the accelerator on that day. In the past decade, more automated tuning routines have been developed and applied to assist the manual tuning in particle accelerators and mitigate issues of high variance in performance and scalability to higher dimensional tasks. This includes traditional optimization methods like parameter scans, random search, Nelder-Mead simplex optimization [165, 166], robust conjugate direction search (RCDS) [167, 168], and extremum seeking (ES) [169, 111]. Although they have been successful in individual applications, they are often limited in some aspects, such as sample efficiency, robustness to noisy measurements, and the ability to escape local optima and perform global optimization.

Automating the accelerator tuning process can greatly improve the availability It is also an essential step towards a paradigm change in the accelerator operation scheme, allowing the operators to specify high-level physical quantities required for experiments instead of focusing on how individual parameters can be achieved. Machine learning (ML) algorithms are promising methods to design novel intelligent controllers and optimizers and address these challenges in online particle accelerator tuning. For online particle accelerator tuning, two ML approaches have been especially successful and found wide adoptions, namely Bayesian optimization (BO) and reinforcement learning (RL). BO is a class of algorithms designed to globally optimize functions that are expensive to evaluate [170, 64]. It builds a surrogate model for the objective and uses an acquisition function to guide the search. Apart from its usage in simulated optimization tasks, BO can also be directly applied for online tuning where the system dynamics are not fully known thanks to its sample efficiency. RL is a paradigm closely related to control theory. It aims to find the strategy to achieve optimal results in a data-driven approach, i.e. through trial-and-error interactions with an environment. Although it is easier to justify the usage of RL algorithms for time-sensitive tasks such as the microbunching control at storage rings [37, 38, 99] and power supply regulations [39], RL has been successful also in tuning tasks where real-time feedback is not necessarily required [36, 35, 171, 39]. The latter approach employs the idea

that RL algorithms can be used to train domain-specific optimizers [172, 173] if enough training data or a fast-executing environment is available.

As the field of applying ML methods for particle accelerator tuning is young and emerging, most works in this field focus on exploring the feasibility of these methods and applying them to new tuning tasks. There is a lack of overview or consensus on what are the advantages of different ML methods and the practical challenges in applications. In the scope of this dissertation, I worked on the Autonomous Accelerator project [53], a collaboration between KIT and DESY devoted to developing novel and transferable ML methods for the autonomous operation of linear accelerators. The developed tuning methods are expected to be transferred between the two similar test facilities, the far-infrared linac and test experiment (FLUTE) and the accelerator research experiment at SINBAD (ARES). We investigated BO and RL in detail and performed systematic studies comparing both methods in terms of performance, engineering effort, implementation details, and potential limitations.

This chapter includes the simulated and experimental results of applying BO and RL for online accelerator tuning at different particle accelerators, i.e. the European X-Ray Free-Electron Laser (EuXFEL), the FLUTE, and the ARES. It showcases the workflow of applying BO and RL methods for new online tuning tasks and provides a guideline on method selection and the practicalities of implementations. First, Section 6.1 presents the experiments of applying BO for the radiation intensity tuning at the EuXFEL. It shows that the BO is now mature enough as a ready-to-use optimizer with performance on par with the routine operational tools. Next, the FLUTE tuning task as presented in Section 4.2, is used as a proof-of-principle task to demonstrate how such an accelerator task can be alternatively formulated and solved as an RL problem. In Section 6.2, an RL agent is trained to solve the FLUTE tuning task in simulation, with the goal of applying the agent for online tuning in the future. To compare the performance of BO and RL experimentally, the ARES Experimental Area (EA) beam tuning task is used as a benchmark. We performed the first detailed comparison study of these two promising ML approaches in accelerator tuning and evaluated them both in simulation and experiment. Section 6.3 presents the comparison results and discusses the performance of each method in different realistic scenarios. Based on the findings from the comparison study, Section 6.4 further discusses the potential of these ML-based methods and points to future research directions. In particular, the domain randomization technique is studied to train RL agents that can be applied to similar lattice sections in ARES and FLUTE. This idea is further extended using the meta-RL algorithm. It is expected that reliable and generalizable controllers can be built for future accelerator operations by combining the strengths of RL and BO algorithms, for example in the form of GP-MPC [174]. Parts of the results presented in this chapter have previously been published in [76, 175, 42, 95, 176, 102].

6.1. SASE Tuning at European XFEL

A common scenario for online accelerator tuning is that the dependency of the optimization objective on the input parameters is completely unknown or no accurate simulation model exists. In such cases, one has to resort to black-box optimization methods. The self-

amplified spontaneous emission (SASE) process at linac-based free electron lasers (FELs) is one prominent example of such tuning tasks. In SASE, the spontaneous undulator radiation in the first section serves as the seed, modulating the electron bunches to form microbunching and generating coherent narrowband radiations in the subsequent undulator sections. High beam qualities, i.e. low emittance and low energy spread, as well as a precise overlap between the bunches and the radiation pulses are required to maintain the exponential power growth through the undulators until saturation. Tuning the SASE intensity is a complicated task and needs to be performed on a day-to-day basis during the FELs operations. At the EuXFEL, for example, over 400 devices have been used to perform the SASE optimization [177]. The beam qualities entering the undulator beamlines can be controlled by optimizing the injector sections, accelerating structures, and quadrupole magnets, where detailed simulation models exist to assist the tuning process. However, the electron beam trajectory inside the undulator sections is largely affected by real-world issues such as the residual undulator fields, and beam-based orbit correction is needed. This beam orbit tuning task is performed regularly using the Nelder-Mead simplex algorithm. It is implemented within the OCELOT optimizer framework and has become the standard tuning tool at EuXFEL. As I gained experience in BO both in simulation as shown in Section 4.2 and experimentally for the storage ring injection optimization [31], I was also invited to a beam time at EuXFEL, where I could test the BO method for FEL tuning.

Specifically, this section considers the beam orbit tuning task for the SASE1 beamline at EuXFEL [6], the layout of which is shown in Fig. 2.8. The beamline consists of 35 undulator cells of 5 m length, separated by 1.1 m sections equipped with air coil correctors, permanent magnet phase shifters, quadrupoles, and beam position monitors (BPMs). Two combined horizontal and vertical air coil correctors are installed at the entrance and exit of each undulator to perform fine orbit adjustments and compensate for the field errors. The goal of the tuning task is to maximize the FEL pulse energy measured by the X-ray gas monitor, which is an ionization chamber downstream of the undulators.

One of the challenges in the SASE tuning task is the noisy signal. As the seed radiation comes from the shot noise in the initial charge density distribution of the bunch, the output radiation will have a high shot-to-shot variance both in the temporal and spectral profiles. The pulse energy measurements are averaged over 30 shots to reduce the noise and stabilize the SASE tuning. Figure 6.1 shows the measured standard deviation of the output signal depending on the pulse energy. Apart from the linear dependency for low pulse energies, most signal noises are centered at $\mu_{\text{std}} = 126\,\mu\text{J}$ and $\sigma_{\text{std}} = 100\,\mu\text{J}$ due to the noisy SASE process. For the 10 Hz repetition rate at EuXFEL, the averaging over 30 shots corresponds to in total 3 s per observation.

6.1.1. Implementing Bayesian optimization for SASE tuning

The SASE energy maximization task can be viewed as a black-box optimization task. The BO proves to be a promising technique to solve such kind of task in a sample efficient way [64]. For this work, I implemented a custom version of BO using the BoTorch [178] package. The Matérn-5/2 kernel is used as the Gaussian process (GP) kernel function and upper confidence bound (UCB) is chosen as the acquisition function. The corresponding GP hyperparameters are fitted dynamically during the optimization process. This data-

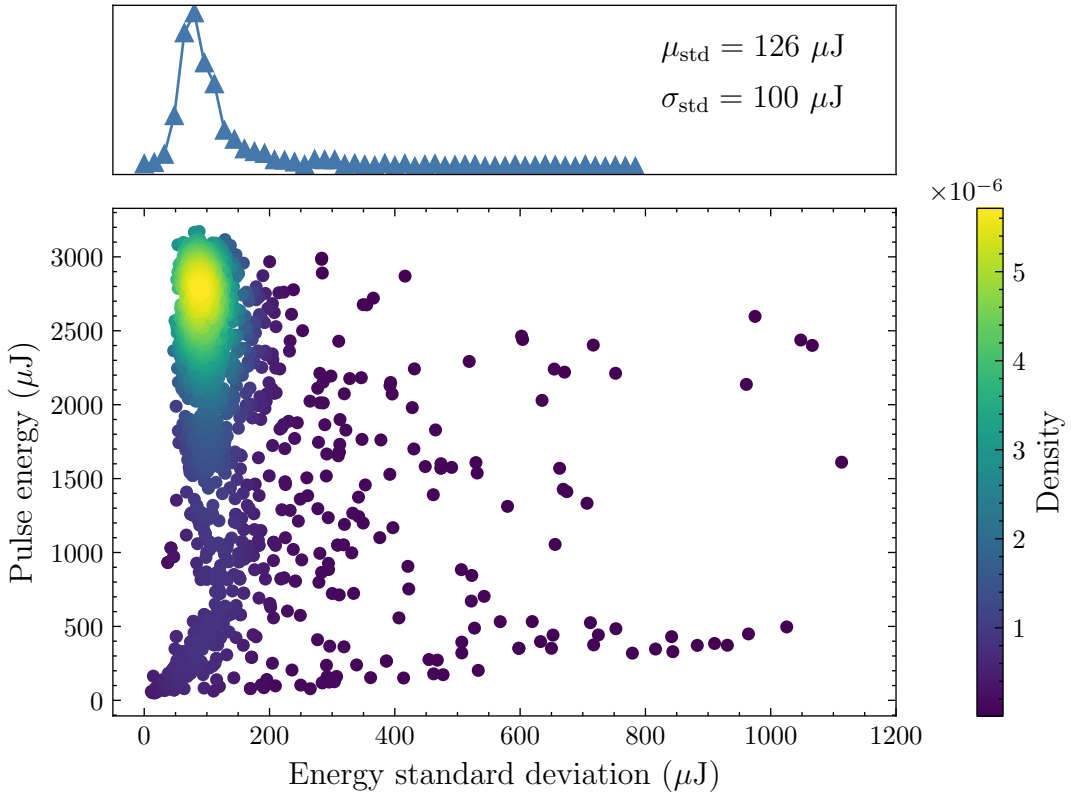


Figure 6.1.: Measurement noise with respect to the photon pulse energy. The lower plot shows measured X-ray pulse energies averaged over 30 shots and the corresponding standard deviations. The measured points are color-coded by their density. The upper plot shows the measurement noise integrated over all energies, giving an average noise of 126 µJ [76].

driven approach is more robust against machine condition changes compared to the earlier applications [31, 28], where the hyperparameters are determined before the optimization by fitting the archived data.

A well-known problem of BO is its over-exploration behavior as a global optimization method. Due to the exploration-exploitation trade-off, it sometimes performs large steps toward unexplored regions of the parameter space. In practice, this can either lead to a sudden performance drop or even cause damage in the case of high-energy accelerators [30, 31]. Safety or step size constraints are often introduced to mitigate this issue, but hard limits can potentially over-constrain the search space and hamper the optimization performance [64]. For the studied SASE tuning task, the *proximal biasing* [75] is employed to mitigate this over-exploration issue. Taking the univariate optimization task as an example, instead of setting hard limits on the optimization step sizes, the next sample point is chosen by maximizing the product of the acquisition function and a normal distribution $\mathcal{N}$ centered at the current settings $x^{(t)}$

$$x^{(t+1)} = \arg\max_x \alpha(x) \cdot \mathcal{N}(x^{(t)}, l_{\text{b}}^2). \tag{6.1}$$

This acts effectively as a soft limit, as the possibility to sample parameter settings that are far away from the current setting is reduced. The amplitude of this penalization is controlled by the biasing lengthscale l_b. It can be easily extended to the multivariate formulation, where the acquisition function is multiplied by the multivariate normal distribution and the lengthscale becomes a vector $\boldsymbol{l}_b$, containing the bias terms for each input parameter.

6.1.2. European XFEL SASE tuning results

The implemented BO method was benchmarked against the Nelder-Mead simplex algorithm implemented in the OCELOT optimizer framework. The hyperparameters of the simplex method, such as the initialization of the simplex, have been tuned by the operators during years of operation. The detail on the Nelder-Mead simplex algorithm is further described in Appendix A.5.1. To evaluate the performance of the optimization algorithms in a reproducible manner, the corrector strengths were manually detuned to the same initial settings, reducing the X-ray pulse energy to about one order of magnitude lower than what is obtained in normal operation.

First, the BO and simplex methods are compared using one pair of horizontal and vertical air coils starting from the same initial setting. The progress of the obtained X-ray pulse energy is shown in Fig. 6.2 (a). Both methods could successfully maximize the pulse energy to about 2800 µJ within 50 steps. The evolution of the two corrector values during the optimization is visualized in Fig. 6.2 (b, c), where a clear distinction is visible. In the case of simplex optimization, the correctors were varied back and forth in the allowed parameter space before converging to the optimal settings. In comparison, the proximal biased BO demonstrated a much smoother convergence towards the optimum, as the smaller steps are preferred and the large oscillations are penalized. As the air coil correctors investigated here do not suffer from magnetic hysteresis, both methods demonstrated similar performance. In general, however, such a smooth parameter change could be more beneficial when hysteresis is present or stress on the magnetic power supply needs to be minimized.

The optimization was then repeated with an increasing number of randomly selected air coil correctors up to 10 degrees of freedom. The final results are listed in Table 6.1. In most cases, BO and simplex reached comparable final pulse energies, with the only exception being the four-dimensional case, where BO seemed to converge to a local optimum. The steps to convergence for each algorithm are also given. Here, the convergence is defined as the step at which the variation of the objective values afterwards are smaller than 226 µJ, i.e. the 1σ upper bound of the measurement noise as calculated earlier. BO reached a faster convergence than the simplex method in 4 out of 5 trials. As expected, the steps to convergence for both methods increase with the number of tuning parameters, as the parameter space of the task grows exponentially. For the 10-dimensional case, both methods optimized the SASE intensity within about 100 steps, corresponding to about 5 min of beamtime. Although the entire section offered over 100 actuators, in practice only a subset of them need to be tuned to reach maximal FEL output power. In other words, local optima with high enough objective values are sufficient for the SASE optimization

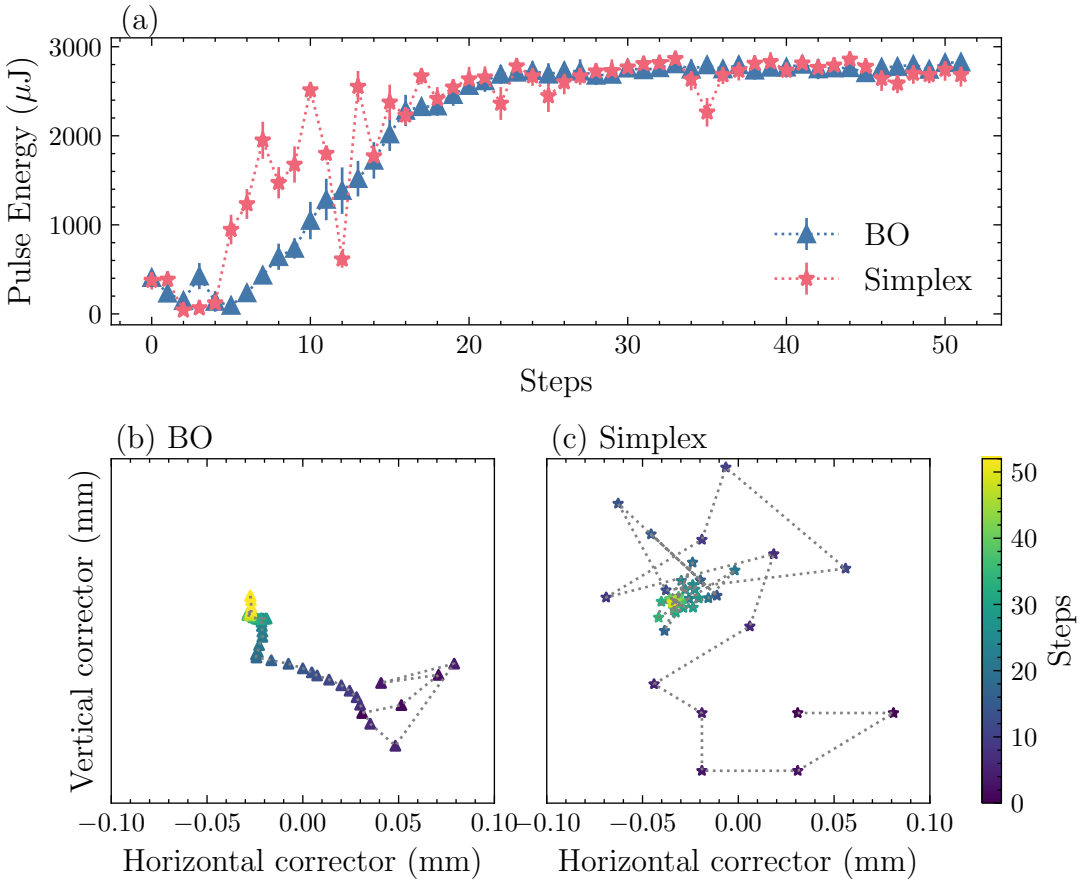


Figure 6.2.: Progress of BO and simplex for the two-dimensional optimization task using one pair of horizontal and vertical correctors in an undulator. The measured X-ray pulse energies are shown in (a). The evolution of the corrector values during the optimization steps for BO and simplex are visualized in (b) and (c) respectively. [76]

task. Therefore, running several optimizations consecutively with a smaller amount of actuators is often preferred, to reach a good enough setting in minimal beamtime.

In this beam orbit tuning task, the time for the objective evaluation is mainly attributed to the averaging of the radiation signals, since the air coils have a fast settling time below 1 s. One possibility to further improve the tuning speed is by reducing the number of averaging samples when measuring the output SASE pulse signals. The BO algorithm was tested again on the 6-dimensional task with different numbers of averaging pulses, corresponding to an averaging time of 3, 1, and 0.5 s. The progress is shown in Fig. 6.3. Despite the more noisy signal, reducing the averaging samples to $n_{\text{avg}} = 10$ does not seem to have an impact on the BO performance. With $n_{\text{avg}} = 5$, the optimization became slower and reached a lower final pulse energy. However, at this configuration, the magnet settling time and the computation time of BO started to become the major contributor to the latency, taking up to about 1 s. Reducing the averaging number further will not effectively speed up the tuning process. As a result, reducing the number of averaging samples to 10

Table 6.1.: Optimization results of BO and simplex with up to ten tuning parameters. The number of steps to convergence is defined as the step after which the variation of the objective values is smaller than the 226 µJ corresponding to the measured signal noise.

# of inputs	Final pulse energy (µJ)		Steps to convergence	
	BO	Simplex	BO	Simplex
2	2880	2864	22	36
4	2500	2900	27	28
6	2845	2852	34	45
8	3120	2944	102	105
10	3011	3049	84	78

has the potential to maintain the FEL performance and reduce the overall tuning time by a factor of two.

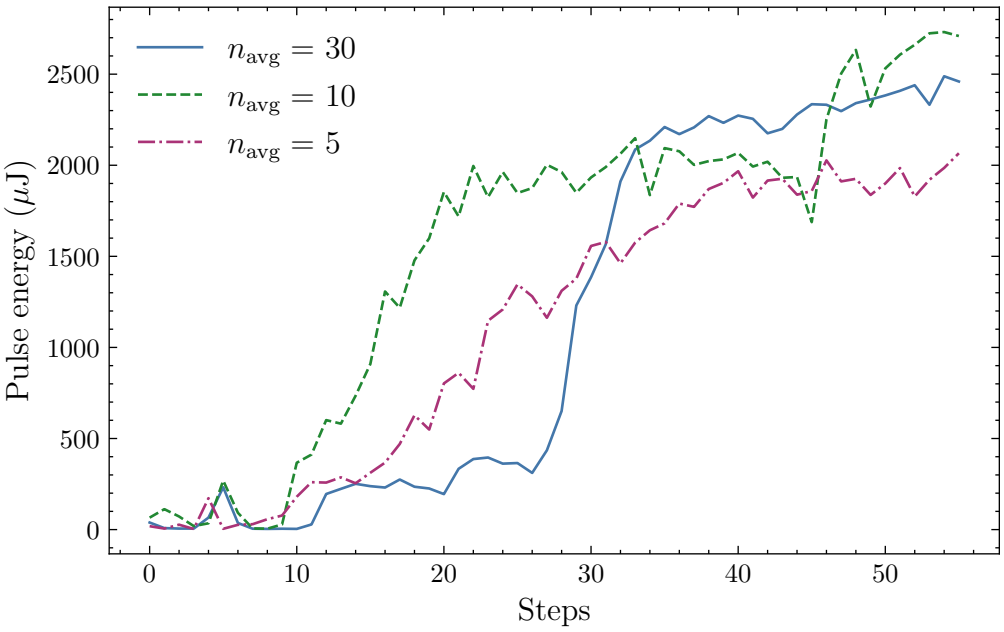


Figure 6.3.: EuXFEL tuning with BO for different numbers of averaging X-ray pulses. For the 10 Hz repetition rate, the numbers $n_{\text{avg}} = 30, 10, 5$ correspond to 3, 1, and 0.5 s respectively. In all the runs, six tuning parameters were used with the same initial settings.

An important feature of BO is that it can shed light on the optimization task by using its GP model trained from the data points. The GP model can be viewed as a surrogate model, as discussed in Section 4.1. For example, the SASE tuning task using four air coil correctors is visualized in Fig. 6.4. In each subplot, the GP-predicted posterior mean function of one pair of input parameters is shown, while the other two parameters take the average values. The one-dimensional histograms at the top show the dependencies of the objective value

on the individual corrector strength. It can be seen that the pulse energy is very sensitive with respect to both horizontal and vertical correctors in cell 3, showing narrow peaks in the posterior landscape. The condition is more relaxed for the horizontal corrector in cell 7, shown by a longer vertical strip in the posterior parameter space plot. This could either assist the operator in monitoring the optimization process during operation or help gain a better understanding of the system's dynamics afterwards.

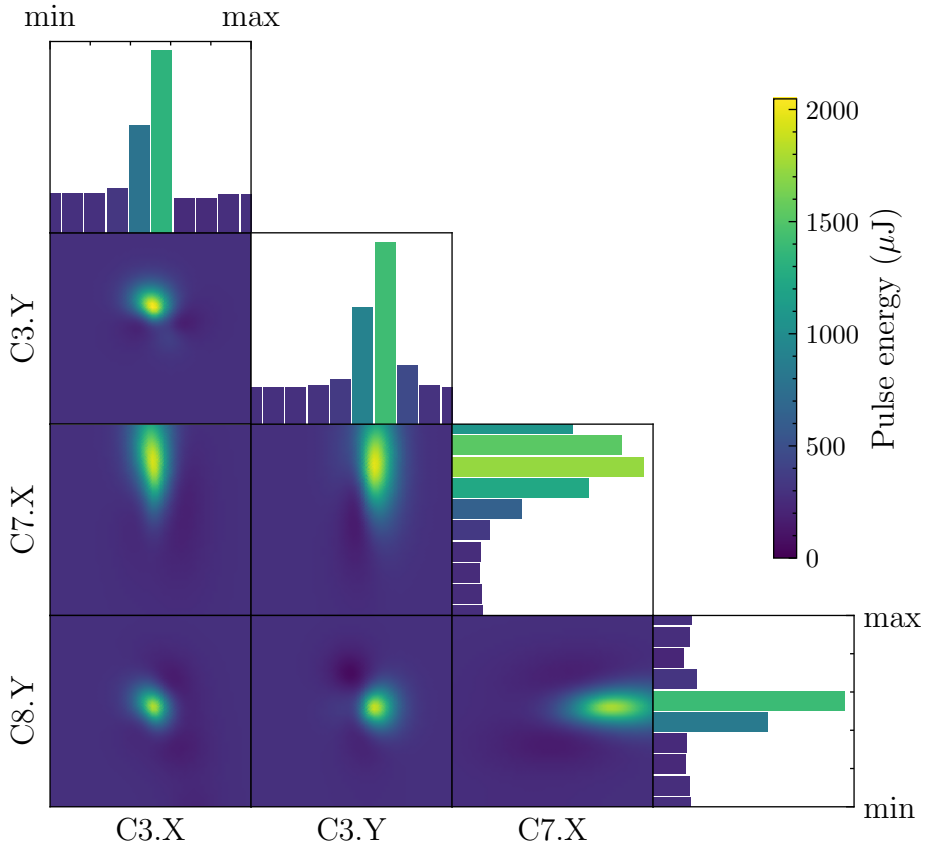


Figure 6.4.: GP Model visualization of the tuning task with four correctors located in undulator cells 3, 7, and 8, where X and Y denote horizontal and vertical respectively. The two-dimensional subspace plots show the posterior mean predicted by the GP model with different combinations of correctors, while the other parameters take their average values. The one-dimensional histograms show the dependencies of the pulse energy on the individual corrector strength.

It can be concluded that BO proves to be a promising method for solving tasks like FEL SASE pulse energy tuning. Even without fine-tuning to the specific task, BO achieved comparable final pulse energies and faster convergence compared to the routinely used simplex tuning method. In this custom implementation of BO, the overhead of applying BO for new tuning tasks is largely reduced, so that little to no expert knowledge is required during the application, making it an operator-friendly tool in the accelerator control room.

6.2. Reinforcement Learning Control for FLUTE Tuning

The task of efficiently tuning the performance of an accelerator during operation can also be viewed as a control task, which can be solved using the framework of RL algorithms [172, 173]. In such a formulation, the goal is to train a *domain-specific optimizer* by interacting with the environment, which can subsequently solve the task efficiently during operation [95]. In particle accelerators, the beamtime is often scarce and safety constraints need to be respected during operation. This makes online training of RL agents on real-world accelerators infeasible, unless the training can be performed at a very high rate, such as using edge-computing techniques at storage rings [38, 99]. As a result, the agents are often trained in a simulated environment and transferred to real-world tasks afterwards. In this section, the FLUTE THz generation tasks as described in Section 4.2 are investigated using an RL approach. The goal is to demonstrate the process of implementing and training RL agents to solve a particle accelerator tuning task.

6.2.1. Formulation of the FLUTE Tuning as a Reinforcement Learning Task

Here a simplified version of the FLUTE tuning task is considered. The electron travels from the exit of the linac into the bunch compressor and is observed at the end of the FLUTE lattice. The final electron bunch length σ_z is to be minimized by tuning the strength of the bunch compressor dipole magnets. The accelerator task first must be formulated as a Markov decision process (MDP), i.e. the state, action, and reward tuple. In this task, a fixed beam at the exit of the linac is considered, using the optimized tracking results obtained in Section 4.2. Therefore, the final beam parameters depend only on the bunch compressor setting. In a simulated environment, the state is fully observable. The observation is defined as

$$\boldsymbol{o} = (\theta_{\mathrm{BC}}, \sigma_z), \tag{6.2}$$

where θ_{BC} is the bunch compressor strength expressed in bending angle, σ_z is the compressed bunch length at the end of FLUTE. The action that the RL agent applies is the change to the current magnet setting

$$\boldsymbol{a}_t = \Delta\theta_{\mathrm{BC}}. \tag{6.3}$$

This formulation of applying actions in limited increments prevents the RL from memorizing the global optimal setting in the training environment and losing the ability to generalize to other environments. It proves to be an important design choice when training the agents in simulated environments and transferring them to real-world accelerators [36, 96]. The agent observes the current state and makes predictions based on the trend and direction of changes, rather than directly predicting absolute values. To some extent, this resembles the behavior of human operators. Here, the maximum allowed action is set to be 10% of the complete action space range, allowing the task to be solved within 10 steps for any initial conditions.

The reward function is chosen to be

$$R(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) = w_{\mathrm{beam}} R_{\mathrm{beam}} + w_{\mathrm{action}} R_{\mathrm{action}}, \tag{6.4}$$

which is a weighted sum of a reward based on beam parameters and a reward based on action changes. The beam parameter reward is designed to encourage the RL agent to achieve the target beam, i.e. the shortest bunch length. It is defined as

$$R_{\text{beam}}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) = \text{ELU}(\sigma_{z,\text{threshold}} - \sigma_{z,t+1}) + w_{\text{improvement}}\Theta(\sigma_{z,t} - \sigma_{z,t+1}), \tag{6.5}$$

where Θ is the Heaviside step function giving only a positive reward if the bunch length is improved in the current step, and $\sigma_{z,t+1}$ is the new bunch length after applying the action $\boldsymbol{a}_t$. The first term contains an exponential linear unit

$$\text{ELU}(x) = \begin{cases} x, & x > 0 \\ \exp(x) - 1, & x \leq 0, \end{cases} \tag{6.6}$$

which limits the reward between $[-1, \sigma_{z,\text{threshold}}]$. Based on the simulated results obtained in Section 4.2, the $\sigma_{z,\text{threshold}}$ is chosen to be 5 fs. Note that for simplicity, the bunch length is expressed here in the unit of time. The RL agent receives a positive reward for bunch lengths smaller than this threshold value. In addition, the agent will receive an improvement reward if it reaches a better beam parameter than the current step.

The action change reward is defined as

$$R_{\text{action}}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) = 1 - \left|\boldsymbol{a}_t / (\boldsymbol{a}_{\max} - \boldsymbol{a}_{\min})\right|, \tag{6.7}$$

which is normalized in $[0, 1]$ and penalizes large actions. This acts analogously as a regularization term [179, 180] to prevent the trained agent from performing oscillating actions around the optimal value. Such actions would lead to similar beam parameters but are undesired when applying in real-world systems. In the training, the weighting parameters are heuristically chosen to be $w_{\text{beam}} = 1$,$w_{\text{improvement}} = 0.5$, and $w_{\text{action}} = 0.2$ to balance the behavior of the RL agent.

6.2.2. Implementation of FLUTE Tuning in a Reinforcement Learning Framework

The RL task formulated above is implemented in the Gymnasium [181] environments, a standard interface for representing general RL tasks in Python. The Gymnasium library is the successor of the OpenAI Gym [182] library which was developed during the period of this dissertation. For simplicity, the following sections refer to both libraries as Gym environments.

To facilitate a seamless transfer of the trained RL agents between different simulation models and potential future real-world deployments, the Gym environment is implemented in a backend-agnostic manner, as shown in Fig. 6.5. The environment only implements the basic logic of the RL task, i.e. the observations, the actions, and the reward function. How the environment reacts to an action is implemented in the *backends*. In each step, the Gym environment receives the new action from the RL agent, calculates the new parameters, and sends the new parameters to the backends. For example in the simulation environment, the backend runs the simulation with the new parameter settings and returns the output beam parameters to the Gym environment. The interaction with real-world accelerator

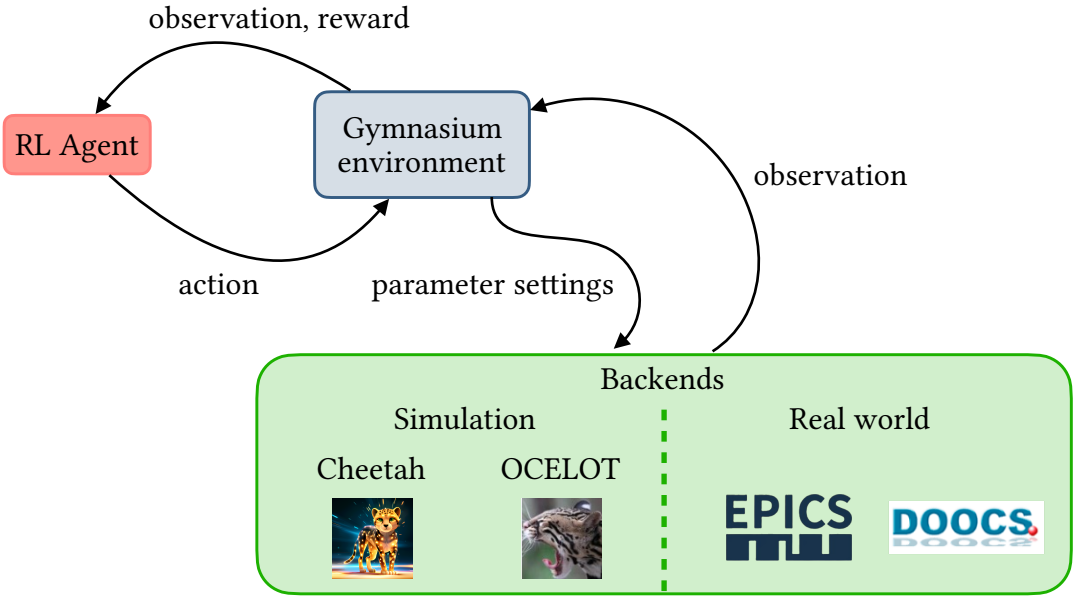


Figure 6.5.: Outline of the implemented RL framework. The accelerator tuning task is encapsulated in the Gymnasium environment. The RL agent interacts with the environment by taking actions and receiving observation and reward signals. The Gymnasium environment includes different backends and interfaces with either simulation or accelerator control systems.

control systems like EPICS [183] and DOOCS [184] can be implemented similarly. The RL agent only interacts with the abstracted Gym environment, allowing the agent to be directly transferred between different backends. For the FLUTE tuning task, the real-world transfer is not tested as the required accelerator section is still under commissioning at the time of completing this dissertation. Nevertheless, the same principle has been demonstrated in the ARES tuning task, as described below in Section 6.3.

The Cheetah [42] simulation model is used as the training environment for its execution speed. The RL agent is trained for 50 000 steps using the proximal policy optimization (PPO) [89] algorithm as implemented in the Stable Baselines3 [185] package. The complete list of parameters used for training is provided in Appendix A.6. In this training configuration, most hyperparameters of the PPO take their default values without extensive tuning. It needs to be mentioned that the large amount of training steps is mainly attributed to the default values of PPO, which are optimized to solve more complex tasks. The required number of steps can be significantly reduced if further adjusting the algorithm to the complexity of the task, for example by increasing the frequency of the policy updates.

To demonstrate that the RL agent trained in simulation can also be deployed in a realistic condition, we subsequently evaluated it in the OCELOT simulation including the space charge and CSR effects. The result is shown in Fig. 6.6, with the evolution of bunch lengths and the bending angles shown in (a) and (b) respectively. The trained RL could tune the bunch compressor smoothly and converge towards a short bunch length of $\sigma_z = 3.4\,\mathrm{fs}$ within about 5 steps. It remains stable and does not oscillate any further afterwards, due

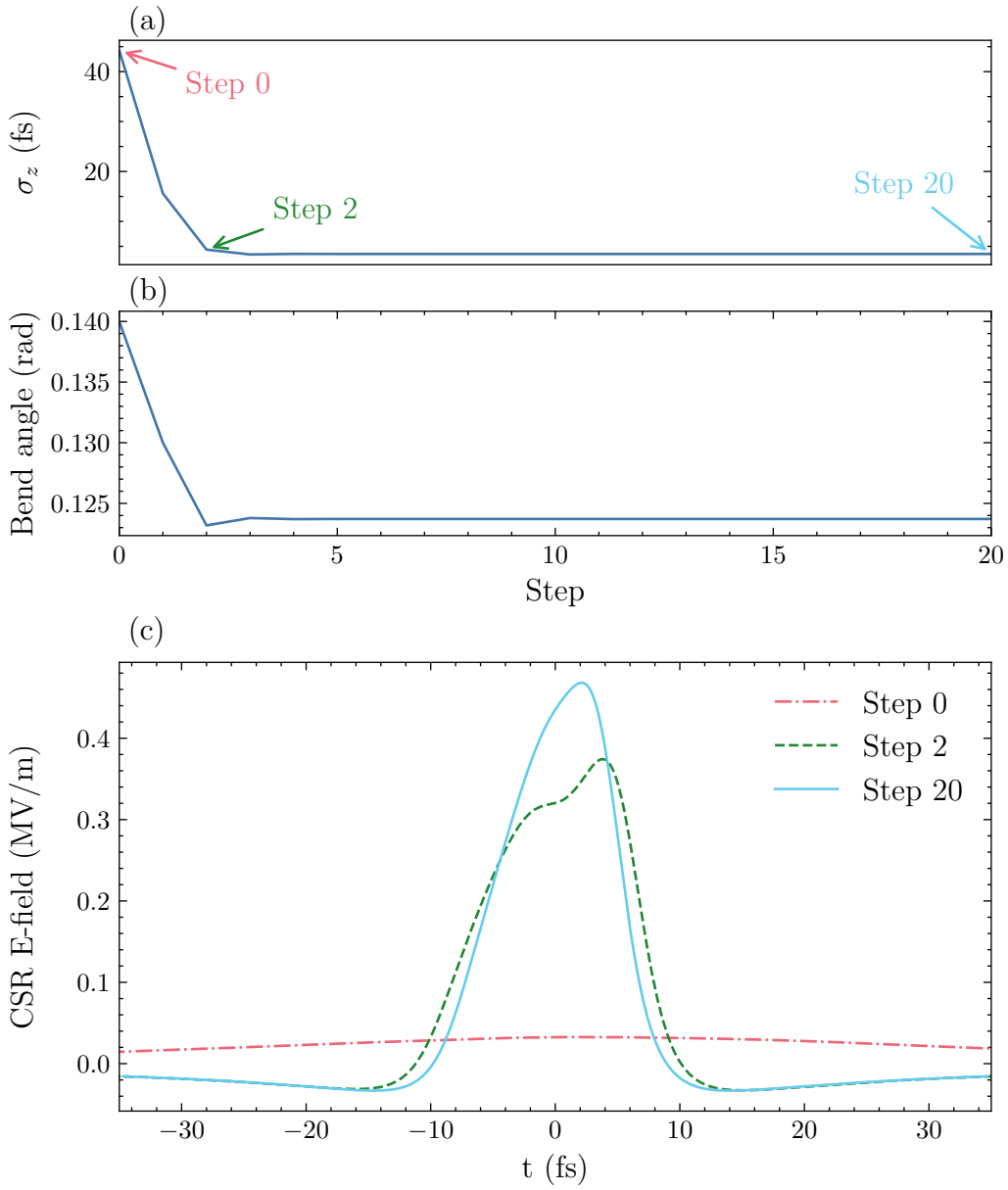


Figure 6.6.: Evaluation of the trained RL agent for the FLUTE tuning task. The agent is evaluated in the OCELOT simulation environment including collective effects. (a) Evolution of the bunch lengths for the first 20 steps. (b) Evolution of the bunch compressor bending angles. (c) The obtained coherent synchrotron radiation (CSR) electric field at different stages of the tuning process.

to the action reward term. The corresponding electric fields of the emitted CSR pulses are shown in (c). After the efficient tuning with RL, the CSR emission can be enhanced to a peak field of 0.46 MV/m. Although the agent is trained in the reduced-physics Cheetah simulation, it can be effortlessly transferred in the more complex OCELOT simulation. The evaluation at the real-world accelerator is not performed within the time frame of the dissertation, as the bunch compressor section at FLUTE is still under commissioning. In conclusion, such a beam tuning task can be easily formulated as an RL task. Once trained using enough, the RL policy can be used as an efficient, domain-specific optimizer.

6.3. Comparing Reinforcement Learning with Bayesian Optimization for Online Tuning

Although RL and BO originated from different research fields, they have both proven effective for online tuning tasks at particle accelerators, achieving superhuman performance. Along with the traditional numerical optimizers like the simplex method and control algorithms like ES, the toolbox of accelerator tuning now includes a variety of algorithms, each with its unique advantages and limitations. When faced with a new tuning task, it is natural to wonder which algorithm to choose, and what limitations and practical challenges might arise. To address these questions, this section systematically compares RL and BO for accelerator tuning in terms of the achieved results, convergence speed, and the practical challenges associated with implementing and applying each algorithm. Specifically, these algorithms are evaluated using their most common setups, where the policy model of RL is completely trained in simulation due to the large number of required training samples, while BO is evaluated from scratch without prior information, learning the Gaussian process GP model entirely online. It needs to be noted that although these setups are the most representative ways to apply RL and BO for online tuning tasks, additional modifications could be applied which will change the behavior of the algorithms. The possibilities for task-specific modifications are also discussed below and in Section 6.4.

6.3.1. Transverse Beam Tuning at ARES Experimental Area

The benchmark task considered here is focusing and steering the electron beam on a diagnostic screen by adjusting a set of magnets, which is a recurring task at many linear particle accelerators. The study is conducted at the EA section at the accelerator research experiment at SINBAD (ARES) [186, 52], which is introduced in Section 2.4.2. The EA section consists of three quadrupole magnets and two steering magnets, also called steerers. Downstream of the magnets, a scintillating diagnostic screen equipped with a camera is used to observe the electron bunch image. At the end of the EA section is an experimental chamber, where user experiments with the electron beam take place. The section is frequently used to fine-tune the pointing of the electron beam and provide specific beam parameters for experiments, such as the dielectric accelerator experiment requiring micrometer-level precision [187, 188].

The layout of the EA section is sketched in Fig. 6.7. The magnets are positioned in the order of $\{Q_1, Q_2, C_\mathrm{v}, Q_3, C_\mathrm{h}\}$, where Q stands for quadrupole magnets, C stands for

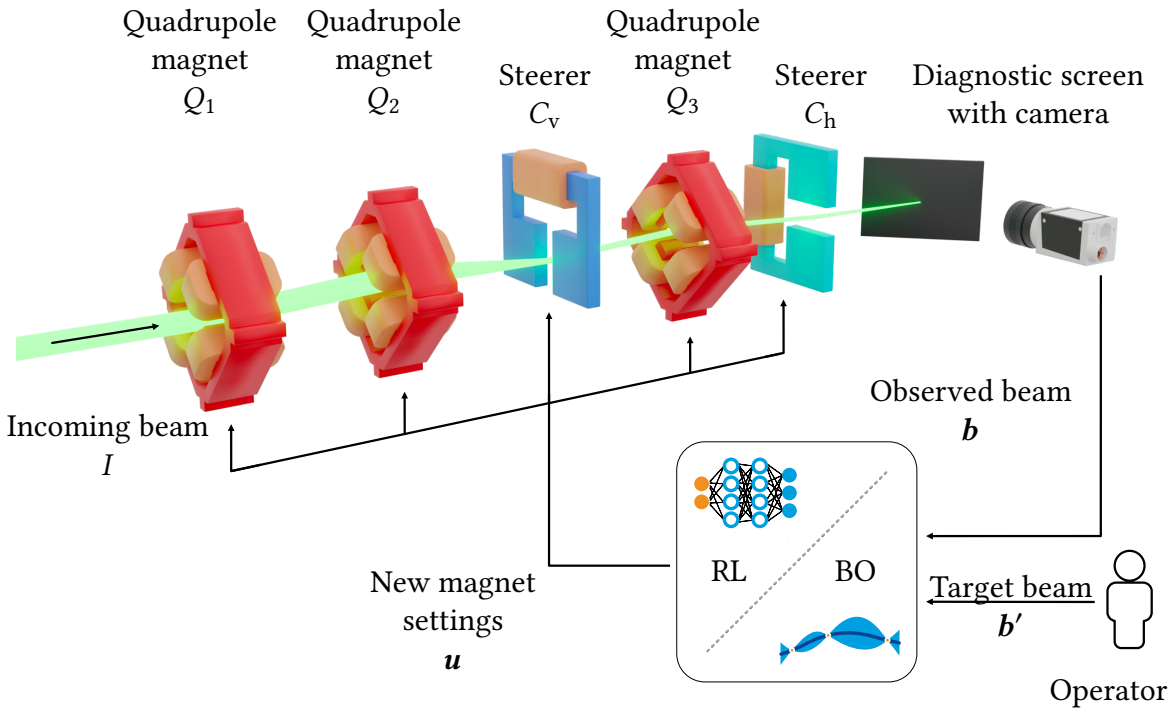


Figure 6.7.: Simplified 3D illustration of the considered transverse beam tuning task at the EA section of ARES particle accelerator. The section consists of three quadrupole magnets and two steering magnets $\{Q_1, Q_2, C_v, Q_3, C_h\}$, followed by a diagnostic screen. The measured beam $\boldsymbol{b}$ and the desired beam parameters $\boldsymbol{b}'$ are provided to the algorithm performing the tuning. In the case of BO, they are used to compute the objective. In the case of RL, they are provided along with the magnet settings as input to the policy and are used to calculate the reward. Both algorithms output either the next settings to the magnets $\boldsymbol{u}$ or a change $\Delta\boldsymbol{u}$ to the current magnet strengths. Reproduced from [95].

corrector/steering magnets, and the subscripts stand for vertical and horizontal steerers respectively. The quadrupole magnets can focus and defocus the beam and the steerers can transversely deflect the angle of the beam. The quadrupole magnets can be operated with normalized field strengths up to $k = 72\,\mathrm{m}^{-2}$. The steering magnets can deflect the beam at maximal $\theta = 6.2\,\mathrm{mrad}$. The camera observing the electron beam has a resolution of 2448×2040 pixels and a size of 8 mm by 5 mm. The effective resolution of the screen camera setup is about 20 µm.

For simplicity, the screen image of the electron bunch is reduced to four scalar values representing the beam parameters $\boldsymbol{b} = (\mu_h, \sigma_h, \mu_v, \sigma_v)$, which are the positions and the sizes of the beam in the horizontal and vertical positions respectively. In the tuning process, certain target beam parameters $\boldsymbol{b}'$ are specified by the operator. The magnet settings are given in the normalized field strengths $\boldsymbol{u} = (k_{Q_1}, k_{Q_2}, \theta_v, k_{Q_1}, \theta_h)$, including the quadrupole

strengths and the deflecting angles of the steerers. The objective O of the tuning task can be generally defined as minimizing the difference between the current beam $\boldsymbol{b}$ and an arbitrary target beam $\boldsymbol{b}'$

$$\min_{\boldsymbol{u}} O(\boldsymbol{u}|M, I, \boldsymbol{b}') = \min D(\boldsymbol{b}(\boldsymbol{u}|M, I), \boldsymbol{b}'), \tag{6.8}$$

with the component misalignments M and an upstream incoming beam I. Here, D is a distance metric quantifying the discrepancy between the current and target beam. In this study, the mean aboslute error (MAE) is used as the difference metric for evaluation

$$D_{\text{MAE}} = \frac{1}{4} \sum_{i=1}^{4} \left| \boldsymbol{b}^{(i)} - \boldsymbol{b}'^{(i)} \right|, \tag{6.9}$$

where $\boldsymbol{b}^{(i)}$ denotes the i-th component of the beam parameters. The output beam parameter depends on the incoming beam I, the current magnet settings $\boldsymbol{u}$, and the misalignments of the components. In the simulations, the incoming beam can be described as a 13-dimensional vector

$$I = \boldsymbol{\mu_x} \oplus \boldsymbol{\sigma_x} \oplus (E_0) \tag{6.10}$$

where E_0 is the reference energy, $\oplus$ denotes the vector concatenation, and $\boldsymbol{x}$ denotes the 6-dimensional canonical phase space coordinates as defined in Eq. (2.14)

$$\boldsymbol{x} = (x, p_x, y, p_y, z, \delta). \tag{6.11}$$

In the simulation, we consider that the particles in the incoming beam are Gaussian distributed in each phase space dimension, characterized by its mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ respectively. For the component misalignments, only the translation offsets of the quadrupole magnets and the screen are considered. They can be described as an 8-dimensional vector

$$M = \left(m_{x,Q_1}, m_{y,Q_1}, m_{x,Q_2}, m_{y,Q_2}, m_{x,Q_3}, m_{y,Q_3}, m_{x,\text{Screen}}, m_{y,\text{Screen}}\right), \tag{6.12}$$

where m_x and m_y denotes the horizontal and vertical misalignments respectively. The misaligned quadrupoles introduce additional dipole kicks and the screen offsets shift the measured beam position. The steering magnets are not affected by the translation offsets. In simulation, it is assumed that the components are not tilted or rotated.

6.3.2. ARES Experimental Area Tuning with Reinforcement Learning

The task is implemented using the Gym environments, similar to what is discussed in Section 6.2. The EA lattice segment is implemented in the Cheetah simulation model to provide fast interactions in the training phase.

6.3.2.1. Formulation as a Reinforcement Learning Task

The first step of applying RL algorithms for the tuning task is to formulate it as an RL task, i.e. define the states, the observations, the actions, and the reward function. The task is

defined as episodic, which means that the environment has initial and terminal states. An episode ends when agent-environment interactions exceed some limits or the difference between the current and target beam is smaller than a threshold value. As discussed above, the output beam parameter depends on the incoming beam I, the misalignments M, and the current magnet settings $\boldsymbol{u}$. The state is therefore defined as

$$\boldsymbol{s}_t = \boldsymbol{u}_t \oplus \boldsymbol{b}_t \oplus \boldsymbol{b}' \oplus (M, I), \tag{6.13}$$

where the subscript t denotes the time step within one RL episode. The target beam $\boldsymbol{b}'$ is given at the beginning of one episode. Other values (M, I) are also assumed to be constant during the episode. The target beam $\boldsymbol{b}'$ is included in the state definition so that the trained RL agent can tune the magnets to achieve all the combinations of possible beam parameters required during operation. In addition, letting the RL agent train with different target beam parameters also allows it to be more easily transferred to the real-world accelerator later.

Although in the simulation all the states could be specified or read, they are only *partially observable* in the real world, as the incoming beam I and the misalignments M cannot be fully measured. These hidden states also affect the output beam parameters and can only be inferred during the episode. In other words, the ARES EA tuning task a partially observable Markov decision process (POMDP). This is also one of the reasons for the mismatch between simulation and real-world measurements and leads to the sim2real transfer challenge. To match with what can be observed during operation, the observation is defined as

$$\boldsymbol{o}_t = \boldsymbol{u}_t \oplus \boldsymbol{b}_t \oplus \boldsymbol{b}', \tag{6.14}$$

i.e. the current beam, the target beam, and the current magnet settings. To increase the robustness of the RL agent when applied in the real world, a *delta-action* scheme is employed, analogous to what is described in Section 6.2. The actions are defined as

$$\boldsymbol{a}_t = \Delta\boldsymbol{u}_t = (\Delta k_{Q_1}, \Delta k_{Q_2}, \Delta\theta_\mathrm{v}, \Delta k_{Q_1}, \Delta\theta_\mathrm{h})_t, \tag{6.15}$$

which are the changes to be added to the current magnet settings. An RL agent trained using direct actions, i.e. predicting the absolute magnet settings, will fail to transfer as there is uncertainty in output beam parameters due to the hidden states. An additional important mismatch between simulation models and real-world accelerators is the calibration error in magnet settings. Overall, this delta-action scheme proves to be crucial for the success of sim2real transfer [36, 96]. The reward function used for training is defined as

$$\begin{aligned} R(\boldsymbol{s}_t, \boldsymbol{a}_t) &= \left(1 + \Theta(-\Delta O_{\mathrm{RL},\,t})\right) \Delta O_{\mathrm{RL},\,t} \\ \Delta O_{\mathrm{RL},\,t} &= O_{\mathrm{RL},\,t} - O_{\mathrm{RL},\,t+1} \\ O_{\mathrm{RL},\,t} &= \ln \sum_i^4 w_i \left| \boldsymbol{b}_t^{(i)} - \boldsymbol{b}_t'^{(i)} \right|, \end{aligned} \tag{6.16}$$

where Θ is the Heaviside step function. The difference between the objectives $\Delta O_{\mathrm{RL},\,t}$ represents the improvement the RL agent achieves at step t. The Heaviside step function is used to enhance the penalization (negative reward) the RL receives when the action results

in a negative improvement. Note that the objective used here O_{RL} takes a different form than the evaluation objective O as defined in Eq. (6.8). The reason is to accelerate the RL training. The weighting parameter $\boldsymbol{w} = (1, 2, 1, 2)$ assigns more weights for the beam sizes than for the beam positions, as it is harder to achieve beam focusing than beam steering. As the tuning task contains beam parameters across orders of magnitude, from millimeter to micrometer range, the changes on the smaller scale will be negligible compared to the initial changes. However, these final steps with precision at the tens of µm range are especially desired and hard to reach in practice. The logarithm in the objective is used to mitigate this effect and encourage the RL agent to perform such fine adjustments.

6.3.2.2. Training the Reinforcement Learning Agent

Here the training configuration of the RL agent is briefly described, more details can be found in [36]. The allowed actions are 10 % of magnet strengths ranges with $\pm 30\,\mathrm{m}^{-2}$ for quadrupole magnets, ±3 mrad for the vertical steerer, and ±6 mrad for the horizontal steerer. The allowed changes for the vertical steerer are half of what is allowed for the horizontal one. This is because the distance from the vertical steerer to the diagnostic screen is two times the distance of the horizontal one to the screen. In such a way, both steering magnets will have similar deflecting effects. In addition, the actions are normalized to $[-1, 1]$ to improve the performance of the RL policy, a 3-layer fully-connected neural network (NN). The rewards and observation are also normalized using a running average over the training progress.

The RL policy is trained using the twin delayed deep deterministic policy gradient (TD3) [90] algorithm as implemented in the Stable Baselines3[185] package. The TD3 is chosen for its relative sample efficiency compared to other algorithms. Nevertheless, other popular RL algorithms like PPO can also be used to train agents with similar performances. The training takes 6 000 000 steps in the fast-executing Cheetah simulation, corresponding to a total of 6 hours of computation time.

As the RL agent is trained in Cheetah and needs to be applied on the ARES accelerator, it is important to account for the discrepancies between the simulation model and the real-world accelerator. During the training, *domain randomization (DR)* [189, 190] is applied to train a robust RL policy. Specifically, at the beginning of each episode, the misalignments M, the incoming beam I, and the target beam $\boldsymbol{b}'$ are randomly sampled from uniform distributions. The misalignments M and incoming beam I can be viewed as unknown in the real world and contribute to the mismatch between the training environment and the real-world environment. This ensures that the trained RL agent can solve the tuning task in a variety of task configurations, with the hope that the real-world accelerator also falls in the sampled distributions.

6.3.3. ARES Experimental Area Tuning with Bayesian Optimization

At the same time, the EA tuning task can be also treated as a function optimization task straightforwardly. It has a clear objective which is to bring the measured electron beam as

close as possible to the target beam. The goal can be described as maximizing the following objective

$$\max_{\boldsymbol{u}} \left[-O_{\mathrm{BO}}(\boldsymbol{u}|M, I, \boldsymbol{b}')\right] = \max_{\boldsymbol{u}} \left[-\ln \mathrm{MAE}(\boldsymbol{b}, \boldsymbol{b}') + w_{\text{on-screen}}(\boldsymbol{b})\right], \tag{6.17}$$

which is the negative logarithm of the MAE between the current and target beam. An additional on-screen reward $w_{\text{on-screen}} = 10$ is added to the objective when the beam can be observed on the screen, and subtracted from the objective when the beam is off the diagnostic screen. This is used to reduce the possibility that BO explores settings where the beam has a large transversal offset. It needs to be noted that this effect can also be achieved by explicitly modeling the beam positions as safety constraints and applying safe BO [77, 78]. The BO objective function contains also the logarithm to encourage further improvements when the beam parameters have already reached the sub-millimeter scales, analogous to the case in RL.

In this study, a custom BO version is implemented using the BoTorch [178] library. The magnet settings $\boldsymbol{u}$ are min-max normalized to $[-1, 1]$, and the objective values are standardized to improve the numerical stability of GP model fitting. The covariance function of the model is the sum of a Matérn-5/2 kernel and a white noise function. The GP hyperparameters, such as the lengthscales and the signal noise, are dynamically fitted to the data using maximum log-likelihood methods in each optimization step. For the evaluation results shown below, the expected improvement (EI) is used as the acquisition function, automatically balancing the exploration and exploitation behavior. Other acquisition functions like UCB have been tested as well and reached a comparable performance.

A custom wrapper was developed for this study. During the period of this study, the Xopt [15] package has been developed as a general-purpose optimization library for accelerator tasks, which also includes implementations of state-of-the-art BO algorithms. These algorithms in the Xopt package are also tested and a custom interface of Xopt with the Gym environment was developed. The results are further detailed in Appendix A.7. Overall, the variant of BO implemented in this dissertation proves to match the performance of those implemented in the Xopt package.

6.3.4. Benchmarking Reinforcement Learning and Bayesian Optimization results

The BO and the trained RL agent were subsequently evaluated in simulation models and at the ARES accelerator. In the simulation, a set of 300 randomly selected trials were used as the evaluation configurations. Each trial is defined by

$$(\boldsymbol{b}', M, I), \tag{6.18}$$

a tuple consisting of a target beam, components misalignments, and the incoming beam parameters. The target beam was sampled in a range of ±2 mm for beam positions and 0 - 2 mm for beam sizes. The ranges are chosen to cover a wide range of measurable beam parameters at the diagnostic screen. The incoming beam parameters are randomly generated within the normal operating range of ARES. The misalignment range is chosen

to be ±0.4 mm, which is larger than the estimated range in the real accelerator. For evaluation, each episode is started from a fixed focusing-defocusing-focusing setting of the quadrupole magnets, with the strengths being $(k_{Q1}, k_{Q2}, k_{Q3}) = (10, -10, 10)\,\mathrm{m}^{-2}$. The steering magnets strengths are set to 0 mrad.

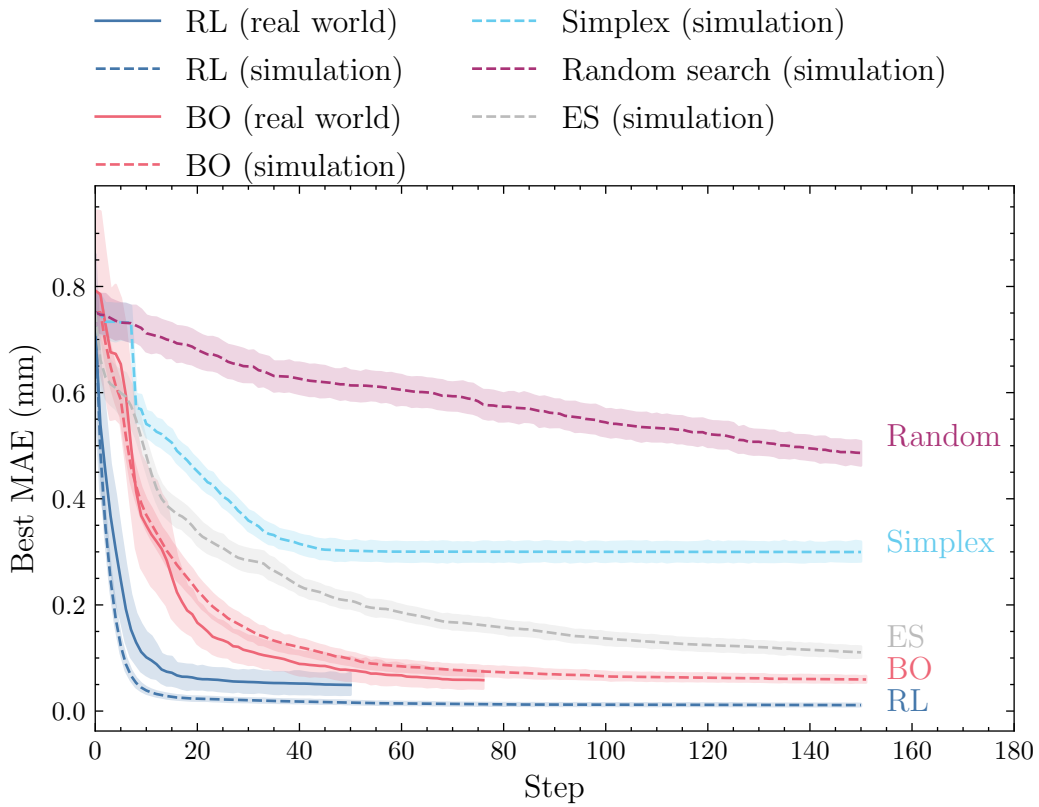


Figure 6.8.: Beam difference convergence plots for different optimization algorithms. The evaluation results of different algorithms are shown for the real-world experiments (solid lines) and the simulated experiments (dashed lines), averaged over all evaluated trials (300 in simulations and 22 in the real world). The simplex, ES, and random search have only been evaluated in simulation. The shaded envelopes show the 95 % confidence intervals of the beam differences averaged over all trials. In the vertical axis, the best beam differences encountered up to each step are shown, i.e. the beam differences that one would return to if the optimization was terminated in the respective step. This metric is called the *best MAE*. [95]

In addition to the learning-based algorithms RL and BO, Nelder-Mead simplex, ES, and random search are also evaluated as baseline algorithms. Details on the Nelder-Mead simplex method can be found in Appendix A.5.1 and ES is discussed in Appendix A.5.2. Both of them are tuned to find the hyperparameter settings that produce the best MAE results in the evaluated trials. The results of the simulation evaluations are shown in Fig. 6.8, with the beam parameters and convergence steps listed in Table 6.2. The best MAE shown in the plot is defined as the best beam differences the algorithm achieved up

Table 6.2.: Performance of different optimization algorithms on the ARES task in simulation and real world. The simulation study consists of 300 trials with different task configurations. The real-world results consist of 22 trials. In the real-world studies, RL was evaluated for 50 steps, and BO was evaluated for 75 steps due to the limited beamtime. In simulation studies, all algorithms performed 150 steps. The steps to target are defined as the number of steps an algorithm took, until MAE is below the threshold of $\epsilon = 40\,\mu m$. An optimization succeeds when MAE below ϵ is reached. The steps to convergence are defined as the number of steps, after which the improvement of best MAE is smaller than the threshold ϵ.

Optimizer	Final MAE (µm)		Steps to target			Steps to convergence	
	Median	Mean	Median	Mean	Success rate	Median	Mean
Simulation							
RL	4	11	7	12	92%	6	7
BO	45	60	40	48	43%	19	29
ES	81	111	121	107	17%	43	47
Simplex	270	300	56	55	1%	28	25
Random	460	490	/	/	0%	28	48
Real world							
RL	24	29	12	13	55%	7	8
BO	44	58	45	48	45%	13	17

to each step. This is used as BO and random search can make large steps leading to a noisy MAE progress. The system can always return to the previously obtained good settings, assuming that the tuning task is not time-dependent. It can be seen that the learning-based algorithms RL and BO outperform the baseline methods both in final beam parameters and convergence speed. The reached MAEs are half of what is achieved by ES and almost an order of magnitude smaller than the ones achieved by the simplex and random search methods. The trained RL policy could even reach a median final beam difference of 4 µm, which is an order of magnitude smaller than the BO result, and also smaller than what could be realistically reached at the ARES accelerator with an effective screen resolution of about 20 µm.

The convergence speed of the algorithms is reported in the Table 6.2 with two metrics. The *steps to target* are defined as the number of steps an algorithm took until the MAE to the target beam is below the threshold ϵ. In this study, the threshold value ϵ is defined as two times the effective resolution of the real measurement setup $\epsilon = 40\,\mu m$. The *steps to convergence* are defined as the number of steps, after which the improvement of the MAE does not exceed the threshold ϵ. They roughly represent, how many steps an algorithm needs to interact with the environment before reaching its best result.

It needs to be pointed out that the convergence speed difference is mostly attributed to the experience RLO gained during the training phase. The number of steps required by BO to convergence can be greatly reduced with an informed GP model, for example by

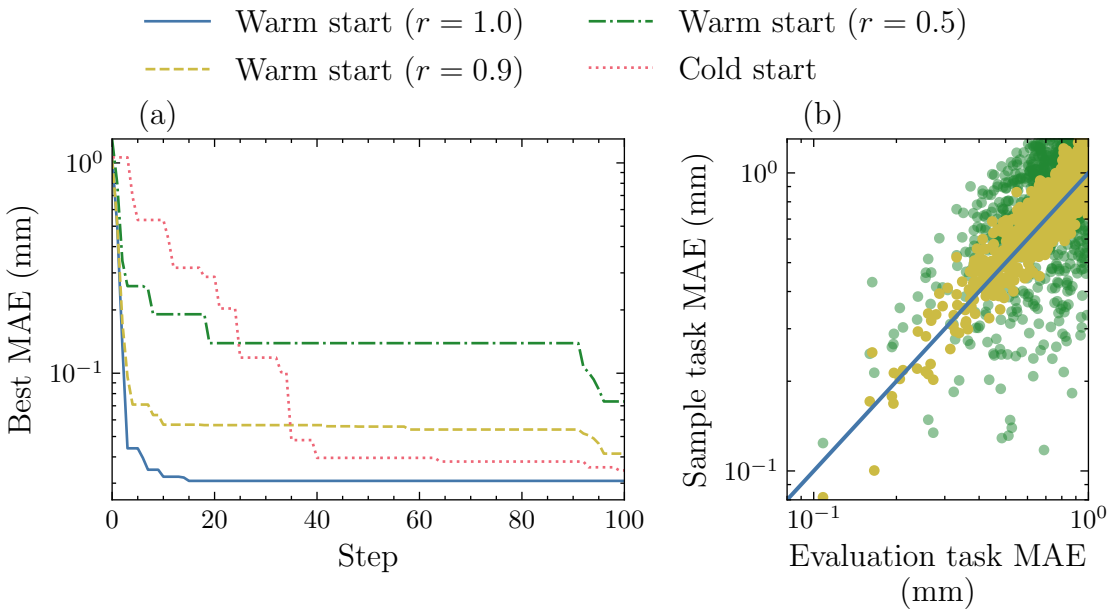


Figure 6.9.: BO warm start using data from previous runs. (a) The progress of BO using historical data with various qualities, quantified by their correlations with the current task, compared with running BO from scratch. (b) Correlation plot of the historical data with respect to the current task. [95]

including historical samples and data from simulations. This idea is tested in the simulation with the results visualized in Fig. 6.9. In the *warm-start* runs, 100 samples from previous simulation results are used to build the initial GP model. The samples are taken from another task, where the incoming beam was varied from the task where the evaluation is run. As expected, the performance of BO improves when historical samples are taken from a task configuration that is similar to the new task. The similarity between tasks can be characterized by the correlation coefficient r, calculated from the MAEs for the sample task and the evaluation task with the same magnet settings. At the extreme, where $r = 1$, i.e. previous samples stem from the same task, convergence can even be achieved in one step. The required amount of data points for warm-starting BO before optimization is four orders of magnitude lower than the samples required by RLO training, due to the sample efficiency of GP models. This makes BO an especially appealing candidate in the absence of fast-executing simulations. As expected, the benefit of using previous samples is decreasing with the correlation r between tasks and can even hamper the performance when the tasks are less correlated, e.g. $r = 0.5$. One way to mitigate this issue is to introduce a weighting factor for the historical samples and gradually decrease their contribution to the GP model building, analogous to what is used in prior-mean assisted BO [112] or adaptive BO [73].

The RL and BO were also tested in the real world. During the evaluation, the settings of the ARES accelerator were usual working points and not set to a particular state. The purpose is to test algorithms' robustness against the different usual conditions. The upstream beam parameters are left at the same state that ARES was in from previous

beamtimes, and therefore as unknown parameters for the evaluated algorithms. The experiments were conducted on 9 different days, running with different bunch charges at 2.6 - 29.9 pC, and reference energies around 154 MeV. The evaluations of RL and BO were run consecutively for each trial, to make sure that the accelerator conditions are comparable. Before each episode, a rough beam-based alignment was performed so that the upstream beam passes closely, but not exactly, through the magnetic centers of the quadrupole magnets and the beam could be observed on the diagnostic screen. This can also be assumed to be a standard operation condition, as the beam is not expected to deviate too far from the design trajectory. The implication of this is further discussed in Section 6.3.5.2. Due to the limited availability of beamtime, 22 distinct trials out of the set of 300 trials were used for evaluation at the ARES accelerator. As opposed to the simulated study, the misalignments and the incoming beam could not be measured or changed during the experiment. Only the target beam parameters were set to the ones as defined in the trials.

The convergence behavior of both algorithms at the ARES accelerator is also shown in Fig. 6.8. It can be observed that although RL still reached a lower final MAEs than the BO results, the differences between them are much smaller compared to the ones in the simulation study. BO demonstrated almost the same performance in all metrics in the simulation and the real-world studies. In comparison, the RL agent reached a median MAE of 24 µm, which is worse than the result in the simulation. The possible reason for the degradation is the mismatch between the training environment and the real-world accelerator, despite dedicated measures to improve the robustness of the RL policy. Among the 22 trials, RL still outperforms BO in the final MAE in 13 trials.

One example trial on the accelerator is shown in Fig. 6.10. The optimization progress of RL is shown in the left panels (a,b,e,g) and the progress of BO is shown in the right ones (b,d,f,h). At the real-world accelerator, the trained RL policy used much smoother actions (a,c) compared to the ones using BO (b,d). This is also reflected in the smooth convergence of the beam parameters (e,g). Due to the exploration behavior, strong oscillations in the beam parameters (f,h) can be observed for the BO episode. The actual beam images observed before and after the optimization are shown in (i,j), where the target beam parameters are marked using dashed lines. Despite having completely different actions during the optimization, both methods produced final beams visually close to the desired beam.

6.3.5. Practical Challenges for Online Accelerator Tuning

The deployment of ML-based algorithms for online accelerator tuning is faced with various practical challenges, including the sim2real transfer mentioned above. This section discusses the ability of the algorithms to deal with such realistic issues, for example, limited diagnostic information and time-dependent systems.

6.3.5.1. Sim2real Transfer

One major challenge of designing an optimizer or controller and deploying it to an accelerator tuning task is the *sim2real* challenge. This issue generally arises when the algorithm

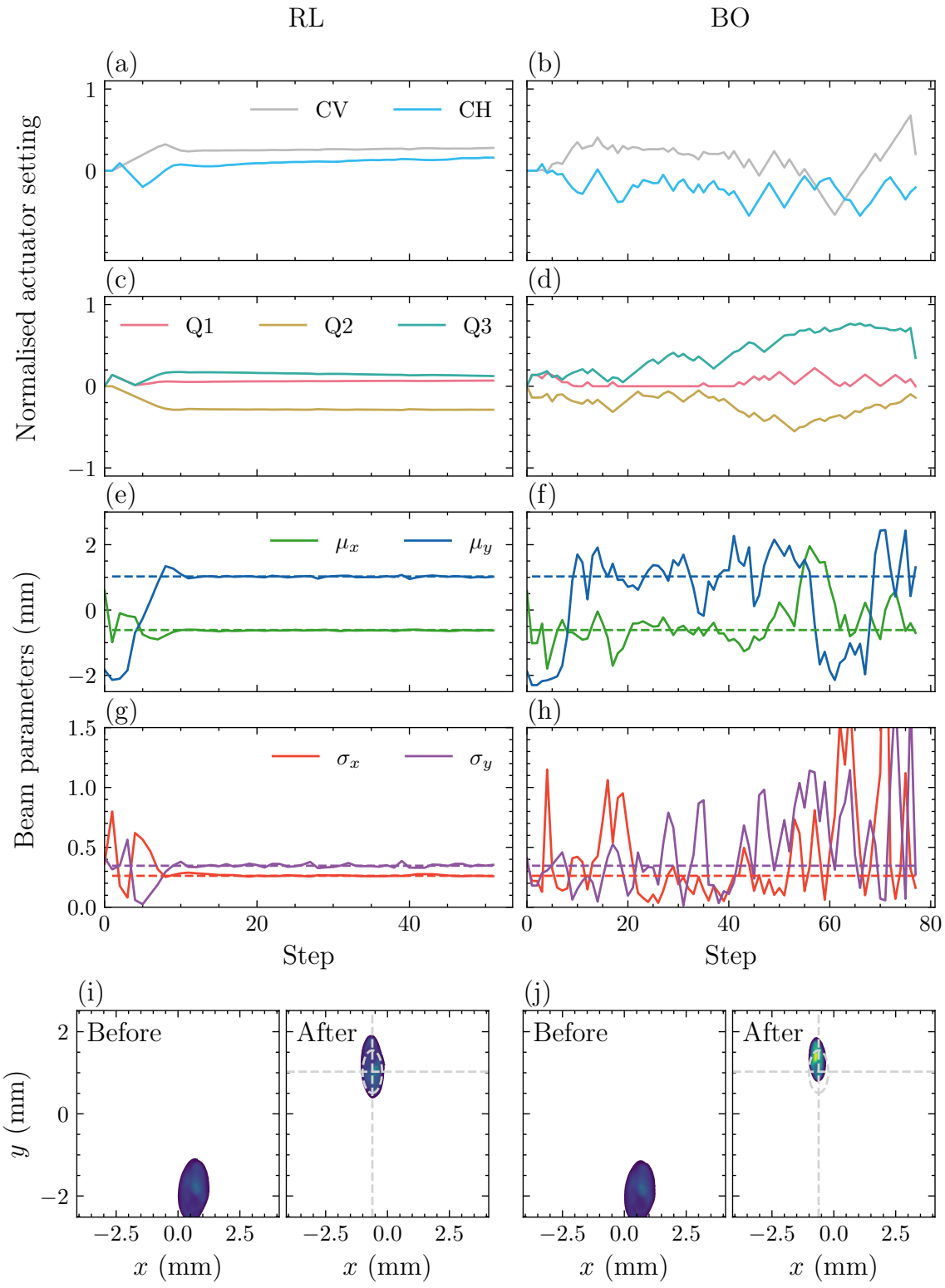


Figure 6.10.: Example optimizations for one trial on the ARES accelerator. (a,c,e,g) show results using RL. (b,d,f,h) show results using BO. The steering magnets strengths are shown in (a,b), and quadrupole strengths are shown in (c,d). (e,f) show the beam positions and (g,h) the beam sizes. (i,j) show the beam images before and after the optimization using RL and BO respectively. The target beam size and position are indicated with dashed lines. [95]

is developed using a cheap-to-evaluate model, where a large amount of data can be collected, and there exist mismatches between the simulation and the real-world system. To overcome the sim2real gap, one needs to be extra cautious in the task formulation. As discussed in Section 6.3.2.1, many design decisions for the RL are made to ensure a seamless transfer between the simulation and the real-world task. One example is the delta-action scheme, which increases the robustness of the RL policy when the effect of an action, changes of the magnet strengths, in simulation and the world are mismatched. The DR technique during the training is also essential. By randomizing the task distribution, in this case, the magnet misalignments and incoming beam, it produces a more robust policy at the cost of increasing the complexity of the task itself.

The BO method, in comparison, usually does not have the sim2real problem. As the way it is implemented in this study and applied to the ARES task, the entire dataset to build the GP model is obtained online during the optimization and the GP hyperparameters such as the lengthscales are fitted at every step. As a result, the trained model is expected to learn the behavior of the real accelerator without any biases. The high-level hyperparameters, such as the choice of kernel functions and acquisition functions, are usually applicable to a wide range of objective functions and thus robust to the potential sim2real mismatches. Nevertheless, if BO uses the simulation data to warm-start the GP model, similar sim2real challenges as for RL will arise.

6.3.5.2. Robustness in the Presence of Sensor Blind Spots

Another challenge in online accelerator tuning is the noisy or erroneous measurements due to the limitation of diagnostic devices. In the EA tuning task, erroneous beam measurements can occur when the electron beam is off-screen. In such a case, the automatic calculation of beam parameters will result in wrong values and falsify the predictions of the algorithms.

During the evaluations, RL can usually recover the electron beam in just a few steps when the beam is outside the diagnostic screen. This is because the pre-trained policy can roughly predict the beam's position based on the magnet settings, even though the non-present beam is not part of its training experience. In comparison, BO struggles to recover the electron beam especially when it is off-screen during the initial steps. The reason is that the GP model is built entirely at the application time and faulty observations at the first steps lead to a useless initial model. As BO lacks the information about the objective function's topology, it can take arbitrarily many steps before the beam is again detected on the screen. On the other hand, steering the beam in the middle of the optimization is not so critical, as this is punished by using the on-screen term in the objective definition Eq. (6.17). Alternatively, this can be mitigated by introducing constraints [75, 30] in the BO and marking the region with erroneous signals as invalid, so that it is only allowed to take steps within the predicted safety regions. The sensor blindness issue can also be addressed by using an informed prior-mean function for the GP model, either built from previous data or directly from the Cheetah simulation [42] as discussed in Section 4.4.

To better understand the impact of sensor blindness, RL and BO are again evaluated in simulation with finite sizes of the diagnostic screen. Both algorithms were tested again on the same set of 300 trials. Unlike the previous evaluation, now the simulation will return

Table 6.3.: Performance of RL and BO in simulation with finite diagnostic screen. The metrics reported here are as defined in Table 6.2, evaluated on the 300 trials in simulation. The diagnostic screen is assumed to have finite dimension and beam measurement returns an erroneous signal if the electron beam is not on the screen. Both algorithms were tested once with incoming beams as defined in the trials' configuration (no beam-based alignment), and once with the beams aligned to the quadrupole magnets (with beam-based alignment). [95]

Optimizer	Final MAE (μm)		Steps to target			Steps to convergence	
	Median	Mean	Median	Mean	Success rate	Median	Mean
No beam-based alignment							
RL	4	13	7	12	93 %	6	7
BO	38	61	44	53	53 %	20	30
With beam-based alignment							
RL	4	10	6	12	95 %	6	6
BO	23	28	33	37	85 %	19	20

an erroneous reading if the beam is outside the physical dimensions of the diagnostic screen. Specifically, it will produce a centered large beam, analogous to the readout in the real-world system. Two different operation conditions are considered, i.e. starting directly from the randomized incoming beam as defined in the trials, and performing a beam-based alignment beforehand to align the incoming beam to the middle of the quadrupole magnets. The latter configuration ensures that the beam can at least be correctly observed on the screen at the first step. The results are listed in Table 6.3. As expected, as RL policy contains an internal model of the system, it is fairly robust against the sensor blind spots. RL demonstrated almost identical performance in the two conditions, reaching a median final MAE of 4 μm and over 90 % success rate of converging to the target beam. In contrast, BO benefits from an initially aligned incoming beam. The final MAE is improved by 40 %, the success rate is increased to 85 %, and convergence speed is also increased by about 25 percent. Overall, it can be concluded that the initial data points are very important for BO if the GP model is learned from scratch.

6.3.5.3. Robustness Against Non-static Systems

A common scenario in the real-world tuning task is the time-dependent system. This happens in particle accelerators at multiple time scales, from the random noise in the low-level control objects like RF appearing below the seconds level to long-term drifts such as the environmental temperature change over months. The high-frequency noises in the radio frequency (RF) or other components are often accounted for using low-level feedback controllers. The long-term drifts occur beyond the period of one optimization run and their impact on the objective function can often be neglected. For the optimization algorithms running over minutes to an hour, the impact usually comes from the middle-term effects such as the heating up of the magnets. The ability to perform optimization

in a drifting system is thus an important feature of an optimization algorithm. Once the RL policy is trained as in this study, it remains fixed during application time. The next action is predicted merely based on the differences between the current settings and beam parameters and the target beam. This makes the RL policy directly applicable as a continuous feedback controller. On the contrary, BO is not designed to solve such continuous tuning tasks. In this study, the BO implementation assumes that the task can be formulated in a static objective function. Being time-dependent means that the objective function can have different values for the same magnet settings. As a result, the GP model built on the observed data points becomes no longer valid for the current system state, rendering the BO prediction less informative or completely wrong. If the system is known to have a time dependency that affects the objective function, the concept of contextual BO (C-BO) [71], also known as adaptive BO [70], can be used to mitigate it. Instead of the basic formulation, C-BO aims to maximize the noisy measurement signal

$$y_t = f(x_t, c_t) + \epsilon, \tag{6.19}$$

where $f : X \times C \rightarrow \mathbb{R}$ is an unknown function mapping from the actuator space X and context space C to a real-numbered objective. The *context* is a set of additional parameters that can be measured but not directly controlled. In particle accelerators, contextual parameters could be the ambient temperature, electron beam current, or simply the time. One application example is the injection at storage rings. The optimal parameter settings change as the stored current increases due to the collective effects. It has been demonstrated that including the beam current as a context variable leads to a more efficient optimization [31].

In C-BO, the GP model approximates the joint mapping of the input parameters and the contextual parameters to objective values. Even with the same parameter settings, the previous data becomes less relevant if the context parameters are distant from the current ones. The next sample to evaluate is chosen by

$$x_{t+1} = \arg\max_x \alpha(x|c_{t+1}), \tag{6.20}$$

maximizing the acquisition function conditioned on certain context values. They can either be directly measured or inferred in general contextual optimizations, or set to the time that the action is expected to take place in the case of using time as a contextual variable [64].

The optimizers are again tested in simulation to evaluate their performance in the case of a non-static system. In the evaluation, the simulated system remains constant for 10 episodes, and the incoming beam starts to drift afterwards. Concretely, the mean position of the incoming beam demonstrates a sinusoidal pattern with a period of 100 steps. In reality, this could be caused by the instabilities of the photo-injector laser system. The progress of different optimizers is shown in Fig. 6.11, with the respective metrics listed in Table 6.4. For each method, the best obtained MAE in the first 75 steps are reported, and the mean and median MAEs are calculated over the steps afterwards. First, the changing beam MAEs without any action are shown with the label "do nothing" as a baseline. The magnitude of the changing incoming beam corresponds to about 237 μm. The RL policy manages to act reliably as a feedback controller and maintain the MAE at a low level

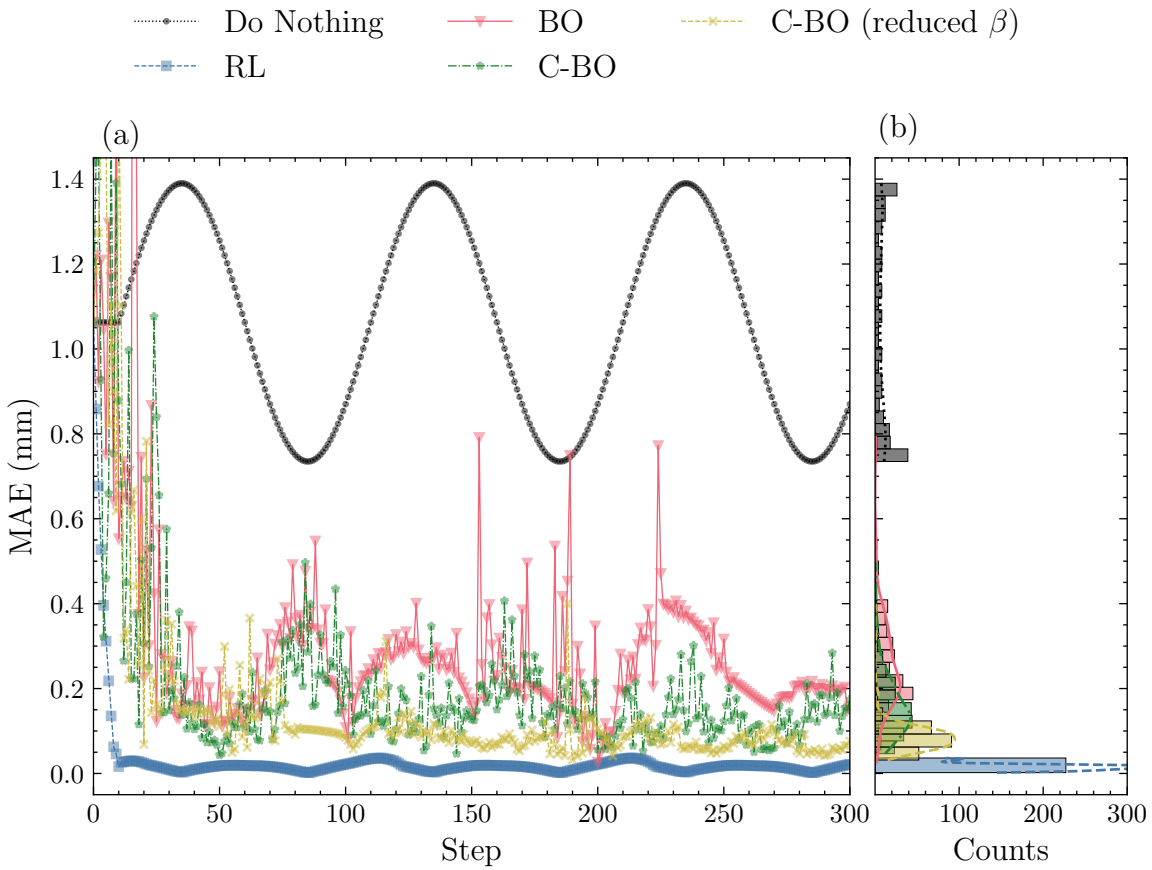


Figure 6.11.: Continuous control of the ARES-EA task with an alternating incoming beam. (a) MAE results of the algorithms: do nothing as a baseline, RL, BO, contextual BO (C-BO), and contextual BO with a reduced exploration factor β_{UCB} after 75 steps. (b) Histogram data for MAE values after 75 steps.

Table 6.4.: Result of applying optimizers for a continuously changing system. "Do nothing" is used as a baseline showing the impact of the changing incoming beam only. The median and mean MAEs are calculated after step 75.

Optimizer	Beam MAE (µm)		
	Best before 75 step	Median	Mean with std
Do nothing	810	1001	1028 ± 237
RL	3	15	16 ± 9
BO	109	231	256 ± 110
Contextual BO	43	141	160 ± 75
Contextual BO (reduced β)	50	82	87 ± 38

with 16 ± 9 µm. The original BO implementation with UCB acquisition, on the other side, starts to strongly oscillate as the incoming beam changes, resulting in a much worse MAE compared to the static case. When the time is explicitly modeled in C-BO, it manages

to reduce the amplitude of the changing beam parameters but still demonstrates large steps due to the exploration behavior. By reducing the exploration factor β_{UCB} after 75 steps, C-BO could follow the near-optimal setting more closely. It reached a mean MAE of $87 \pm 38\,\mu\mathrm{m}$ for the subsequent steps, reducing the changes from the incoming beam by a factor of 6. In this test case, the covariance function for the context variable is also Matérn-5/2 kernel. The performance of C-BO is expected to be further improved if a periodic kernel is used when the structure of the time dependency is known.

It can be concluded that, RL is naturally more suited for controlling continuous systems. BO can also handle optimization tasks with moderate time dependencies if formulated correctly.

6.3.6. Inference Time

Lastly, the inference time is also an important factor to consider when choosing the proper algorithm for an accelerator tuning task. For the benchmark task considered here, the inference time happens to be negligible. In the ARES EA section, the settling time of the magnets and the beam measurement takes tens of seconds in total, which is orders of magnitude higher compared to the inference time of the algorithms. The inference time for RL and BO can vary significantly depending on their choices of models and hyperparameters. For the BO implementation investigated here, the GP as surrogate model is used. One inference step would take about 0.7 s on a modern laptop. This is because BO needs to perform an acquisition maximization at each step, which in turn involves matrix inversions within the GP model with complexity $O(n^3)$, scaling with the number of data points n. A more extensive study on the BO inference time can also be found in [64]. The RL policy, in comparison, is a fully connected NN. At application time, only one forward pass of the network is required for RL to predict the next action. The corresponding time complexity is $O(1)$ with respect to the optimization steps. One inference step of RL takes about 0.2 ms on a laptop, orders of magnitude faster than what is needed for BO. The inference time of RL can be further reduced to the microsecond range with dedicated hardware [99, 38], rendering it a promising method for real-time control of ultra-fast dynamic systems, such as the microbunching control at electron storage rings [37].

6.3.7. Discussion on the Benchmark Study

Based on the evaluation studies for the ARES beam tuning task, RL with a pre-trained policy outperforms BO by achieving better final beam results and faster convergence. Both learning-based algorithms clearly outperform other commonly used tuning methods including the Nelder-Mead simplex and the ES. The implemented RL and BO could be reliably applied at a real-world accelerator, and they always obtained a better beam after optimization than the starting beam. As the RL policy is pre-trained with millions of interactions in the simulation environment, the resulting policy becomes more robust against realistic issues that might arise during the accelerator operation, such as sensor blindness and drifts of the components. The BO, on the other hand, can be deployed without any prior training. This makes it relatively easy to apply BO algorithms for a new accelerator tuning task, as the GP model is very sample-efficient and can be learned entirely

online. When the accelerator conditions change in such a way that the environment that the RL agent trained on becomes outdated or not reliable, the RL agent needs to be retrained. BO is more robust in such scenarios as it is trained fully online. Tailoring the BO further for specific cases can also improve the performance compared to its basic form. It can be concluded that RL should be the preferred option for online tuning, if the upfront engineering effort can be afforded and enough training samples can be gathered, for example via a fast-executing simulation like Cheetah. In comparison, BO is best suited for scenarios without an informative simulation model or any previous experience, such as the commissioning of a new experiment setup.

6.4. Towards Generalizable Machine Learning-based Controller

As the number of accelerator components increases, the amount of distinct tuning tasks increases accordingly. Therefore, it is essential to consider the generalization ability when designing controllers and optimizers for future accelerator operations. Although RL demonstrated promising results in the ARES tuning task, it required a huge amount of upfront engineering effort and training time. The trained RL policy is also limited in terms of transferability, being specialized in the task on which it is trained. It often requires complete retraining when it is applied to another accelerator task. It is uneconomical or even infeasible to train dedicated RL controllers for every single tuning task in a particle accelerator. One future research avenue would be to design novel algorithms to produce RL agents that are robust against changing systems for one single task or can be easily transferred to various similar tuning tasks. This section discusses several approaches to combine the strengths of different ML methods to build intelligent and generalizable controllers to assist the operation of future particle accelerators.

6.4.1. Generalizable Reinforcement Learning Agent with Domain Randomization

The ARES beam tuning task demonstrated that DR is a promising technique to train robust RL agents. During the training phase, the RL agent is exposed to a distribution of MDPs, forcing the agent to take actions under various conditions. This technique successfully trains an RL agent in simulation that is robust enough to be applied in the real-world task zero-shot, i.e. without any adaptation steps. It is natural to extend this idea by increasing the magnitude of the randomness in the training phase, aiming to produce even more robust RL agents.

Particle accelerators are designed following the same principles of physics, resulting in similar lattice sections across different facilities. One example in the linear accelerators is the EA section at ARES, consisting of three quadrupole magnets and a pair of horizontal and vertical steering magnets. It is a recurring lattice section for beam transportation that can be used to focus the beam and correct for orbit deviations. Such a section can be found in many accelerators, also at the FLUTE accelerator [50].

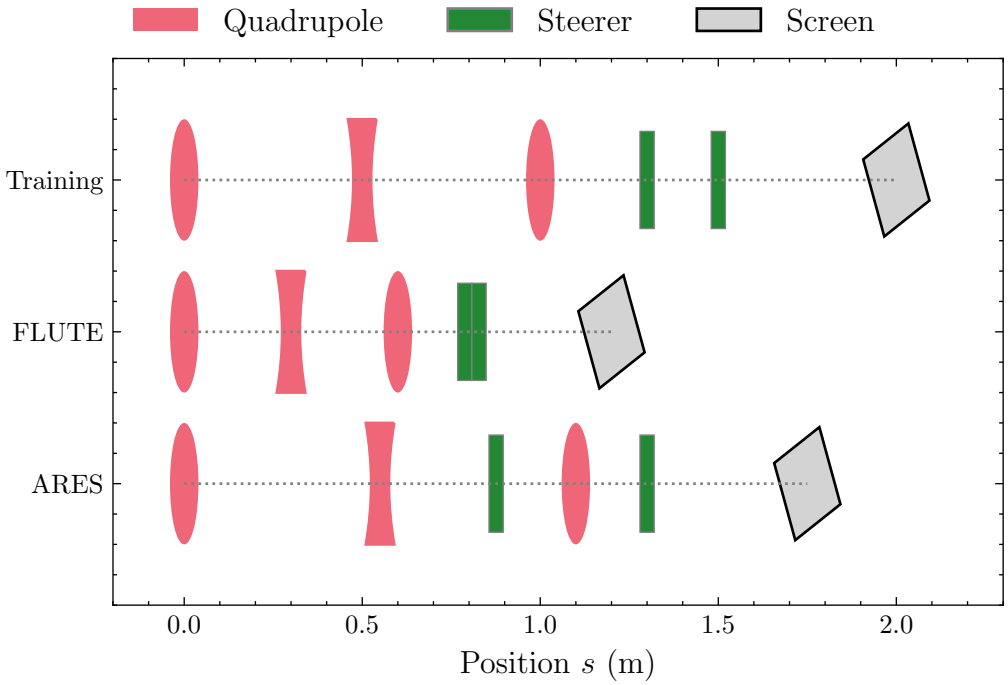


Figure 6.12.: Lattice sections with 3 quadrupoles and 2 steering magnets studied for the transverse tuning task, including a dummy training lattice, the diagnostic section of FLUTE, and the Experimental Area (EA) at ARES. [175]

As a proof of principle experiment, the DR technique is employed to train a robust RL agent that can solve the beam tuning tasks in different lattices with similar magnet configurations. The different accelerator lattices considered for training and evaluation are shown in Fig. 6.12. The training lattice denotes an artificial section that is used for training the RL agent. Two sections of the real accelerators are used for evaluating the agent's performance, namely the diagnostic section at FLUTE and the EA section at ARES.

The environment formulation for the RL task is similar to the ARES one as introduced in Section 6.3.2.1. In each step, the agent applies an action $\boldsymbol{a} = \Delta\boldsymbol{u}$, i.e. the changes of the magnet strengths, which is limited to 10 % of the entire range of possible magnet strengths. The observation is again a 13-dimensional vector, consisting of the magnet settings $\boldsymbol{u}$, the measured beam $\boldsymbol{b}$, and the target beam parameters $\boldsymbol{b}'$. The reward is defined as the normalized negative beam difference

$$R_t = -D(\boldsymbol{b}, \boldsymbol{b}')/D_{\text{max}}, \tag{6.21}$$

where the normalization factor D_{max} is the largest observable beam difference limited by the screen size. Again, the MAE is used as the metric for measuring differences D between beam parameters. The agent will receive a higher reward when the beam parameters are improved. In addition, as the reward is always negative, the agent is encouraged to perform the tuning with fewer steps if possible. The accelerator lattices are implemented using the Cheetah simulation code.

During the training phase, DR was performed on the positions of the components. In each episode, the lengths of the drift sections between the elements are randomly sampled while keeping the order of magnets $(Q_1, Q_2, Q_3, C_v, C_h)$ as in the training lattice. This approach exposes the agent to a wide range of system dynamics, allowing it to learn a more robust policy that can be transferred between different lattices. The RL policy is trained for 500 000 steps in the Cheetah simulation using the soft actor-critic (SAC) algorithm as implemented in the Stable Baselines3 package [185]. The SAC algorithm is used here for its relative sample efficiency and being more robust against hyperparameters than the TD3 algorithm. The hyperparameters of the SAC algorithm are tuned using BO with the experiment tracking package Weights and Biases [191] and listed in Appendix A.8.

One example tuning episode using the trained RL agent is shown in Fig. 6.13. The goal is to produce a focused beam centered at the diagnostic screen $\boldsymbol{b}' = (0, 0, 0, 0)$ with random initial magnet strengths in a focusing-defocusing-focusing setting. The evolution of the beam parameters is shown in (b), with the quadrupole and steering magnets' strengths shown in (c) and (d) respectively. The RL agent changed the magnet settings smoothly and successfully converged to the target beam within only 10 steps. This is consistent with what has been achieved in the ARES tuning task with the same degrees of freedom and significantly faster than manual tuning or numerical optimizers.

The trained policy using DR is evaluated on different lattices, with the results shown in Table 6.5. The RL agent trained using SAC and the DR technique is benchmarked against two other baseline agents, i.e. an untrained agent taking only random actions and an RL agent trained using SAC but without DR. For each lattice configuration, the trained RL agent is tested 100 times with different initial magnet settings and is allowed to interact with the environment for up to 50 steps. The mean value of the best-obtained beam MAEs within 50 steps are listed in the Table 6.5. It can be observed that all the trained RL agents outperform the random actions by an order of magnitude. For the agent trained without DR, its performance drops significantly when applied to the FLUTE lattice, reaching a final MAE 3.6 times worse than the one on the training lattice. When trained with DR, the agent performs much better on the FLUTE lattice, despite reaching a worse MAE on the training lattice than the specialized agent. This decrease in performance is expected when both agents are trained with the same number of steps, as the tuning task becomes inherently more complicated when the magnet positions are changing.

In addition to the linear beam dynamics simulation in Cheetah, a more realistic FLUTE model is implemented using the OCELOT simulation, including the space charge effect. As expected, the RL agent trained on a reduced-physics simulation model could capture the system dynamics and directly be applied to a higher-fidelity simulation model without any degradation in performance. This is consistent with the findings in the ARES tuning task, that the agent trained in simulation can also be applied to the real machine without retraining. The trained agents are also evaluated using the EA section at the ARES linac, with the results shown in the 4th column in Table 6.5. It needs to be noted, that the vertical corrector in the EA section is positioned between the two quadrupoles $Q2, Q3$. This places the EA section out-of-distribution concerning the DR training with a fixed order of magnets $(Q_1, Q_2, Q_3, C_v, C_h)$. Nevertheless, the agent trained with DR is still able to tune the beam and reach a similar performance as other lattice configurations. This indicates that the agent is generalizing to unknown scenarios.

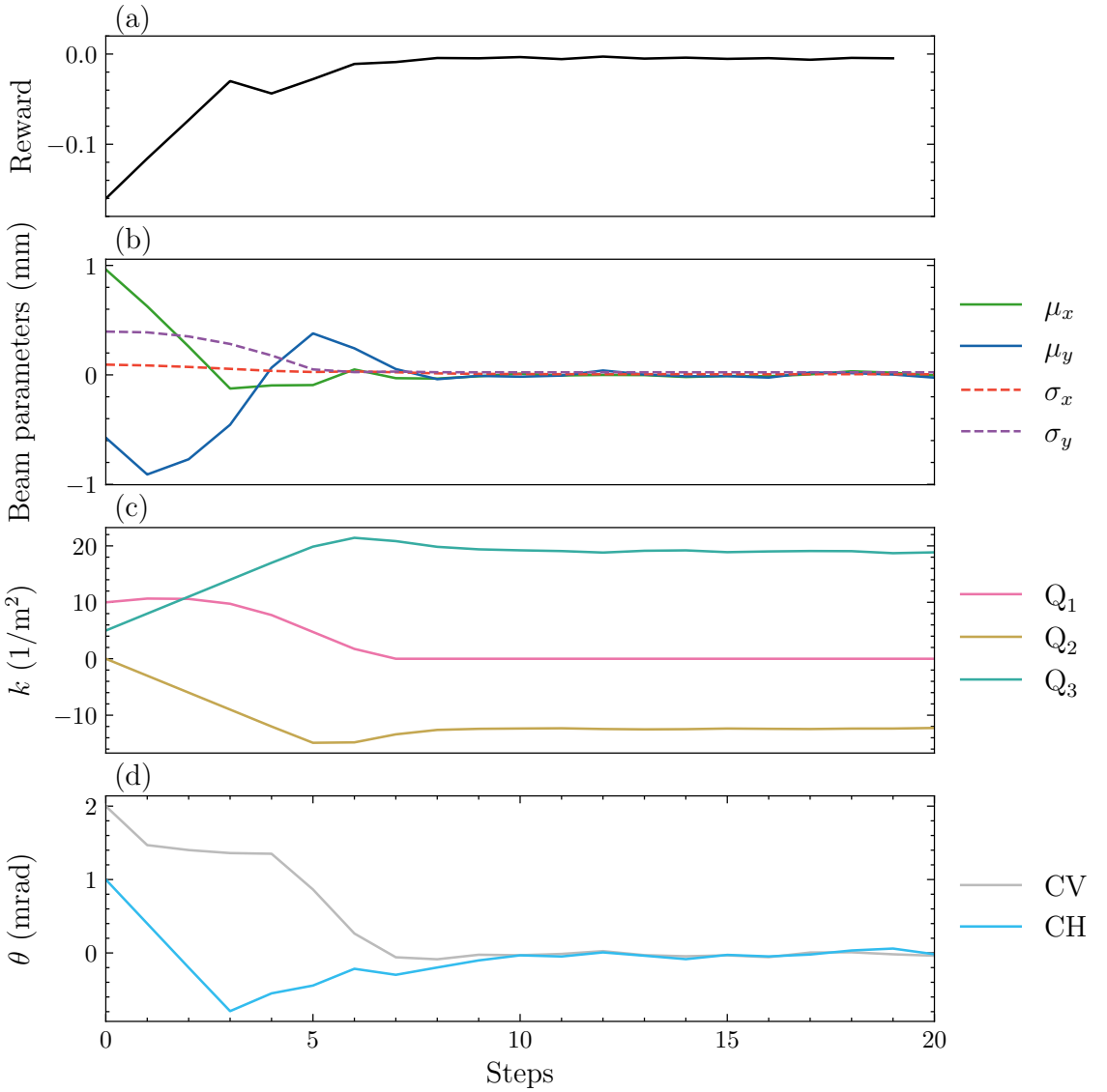


Figure 6.13.: One example tuning episode on the training lattice with the trained lattice-agnostic RL agent. The aim is to focus and center the beam in the diagnostic screen. (a) Progress of the rewards. (b) The evolution of the beam parameters ($\mu_x, \mu_y, \sigma_x, \sigma_y$). (c) Progress of the quadrupole strengths and (d) steering magnets' strengths.

The performance of an RL agent on a single lattice setting can be improved using *fine-tuning*. This means that the pre-trained RL policy is further updated using interactions with the new environments. To evaluate the feasibility of fine-tuning, the domain-randomized agent is retrained for another 10 000 steps, corresponding to only 2 % of the original number of interactions, on the lattice it is tested on respectively. The fine-tuned agent is again evaluated on the lattices, and the results are shown in the 4th row in Table 6.5. After the fine-tuning process, the agent performed better on all the lattices and reached a comparable or better performance than the original agent trained only on one lattice.

Table 6.5.: Performance of the trained RL agents on the simulated training, FLUTE, and ARES lattices. The lattices are implemented in either the linear beam dynamics simulation Cheetah or OCELOT including the space-charge effect. The beam MAEs are averaged over 100 tasks with random initial magnet settings.

	Mean best beam MAE (µm)			
Lattice	Training	FLUTE	FLUTE	ARES
Simulation Code	Cheetah	Cheetah	OCELOT	Cheetah
Random	958	621	548	930
SAC	43	154	150	92
SAC + DR	86	63	60	67
SAC + DR + fine-tuning	52	31	31	36

This indicates that an RL agent trained using DR can be used as a starting point. It can be subsequently retrained on the real accelerators within a reasonable amount of steps. This significantly reduces the overhead of deploying an RL agent for a new tuning task, making the method more accessible across different accelerator facilities.

This simulated study presents as a first step towards training *lattice-agnostic* RL controllers. It shows that the agent trained using DR can be successfully transferred between different particle accelerator sections with similar characteristics, solving the common tuning tasks. To a certain extent, it can also be generalized for out-of-distribution lattices that are not considered in the training phase. At the same time, the DR technique also has its limitations. First, the sample requirement for the training process increases drastically, when the RL task becomes more complicated. At the same time, the required size of the NN policy to capture the dynamics of the task also increases, despite the universal approximation capability of deep NNs [192]. When applying DR with uniformly sampled tasks, the RL policy is trained to optimize its reward on the mean of the task distribution. While this produces a robust policy in the beam tuning task investigated here, the formulation will break if the task distribution is too wide and some tasks require contradictory actions. To conclude, the randomness in the task distribution should be kept to a minimum, containing only what is necessary for solving the real-world task.

6.4.2. Fast Reinforcement Learning Deployment with Meta-Learning

As shown in the simulation results above, while DR produces an RL agent that can solve a distribution of tasks, the trained agent might have inferior performance on the individual tasks compared to specialized agents. This issue can be partially mitigated by fine-tuning the DR-trained agent. As the agent is not specifically trained for this process, fine-tuning still requires a large amount of data, that is sometimes not affordable in the real accelerator. It can even cause an abrupt decrease in performance, referred to as catastrophic forgetting [193]. Meta-learning is one ML paradigm that explicitly addresses this issue. It trains ML models that can efficiently adapt to unseen tasks by leveraging its prior experience. In the meta-RL case, a meta-policy is trained that can adapt to a

specialized task-policy for one RL task among the task distribution, with only a limited amount of training samples. A more detailed treatment on meta-RL can be found in [194].

The meta-RL consists of two levels: a *inner-loop* similar to the normal RL algorithms where the policy is updated by interacting with an environment, and an *outer-loop* which updates the meta parameters based on the results in the inner-loop. The inner-loop is also called the *adaptation* and the outer-loop is called the *meta-training* Specifically, meta-RL aims to maximize the following objective function

$$\begin{aligned} \mathcal{J}(\boldsymbol{\theta}) &= \mathbb{E}_{\mathcal{M}_i \sim p(\mathcal{M})} \hat{\mathcal{J}}_{\mathcal{M}_i}(\pi_{\boldsymbol{\theta}}) \\ &= \mathbb{E}_{\mathcal{M}_i \sim p(\mathcal{M})} \left[\mathbb{E}_{\mathcal{D}} \left[\sum_{\tau \in \mathcal{D}_{K:H}} G(\tau) \middle| \pi_{\boldsymbol{\theta}}, \mathcal{M}_i \right] \right], \end{aligned} \tag{6.22}$$

where $\hat{\mathcal{J}}_{\mathcal{M}_i}(\pi_{\boldsymbol{\theta}})$ is the estimated return when applying the policy on a single task $\mathcal{M}_i$. The goal is to train an RL policy $\pi_{\boldsymbol{\theta}}$ that can efficiently adapt to a distribution of tasks $p(\mathcal{M})$. During adaptation, the policy is trained to learn a new task $\mathcal{M}_i$ drawn from the distribution $p(\mathcal{M})$, using only K episodes with the task $\mathcal{M}_i$. K denotes the number of exploration steps the policy takes, also called *shot*. The policy interacts with the new task $\mathcal{M}_i$ until a maximum number of steps H in each episode is reached, also called the *horizon*. Here, $G(\tau) = \sum_{t=0} \gamma^t R_t$ is the discounted return the policy receives along one trajectory τ in the task $\mathcal{M}_i$. The estimated return $\hat{\mathcal{J}}_{\mathcal{M}_i}$ is calculated with the expectation of the discounted return over sampled trajectories when applying the task policy on a single task $\mathcal{M}_i$. This meta-RL objective $\mathcal{J}_{\boldsymbol{\theta}}$ is defined using the expectation over the sample tasks drawn from the distribution $p(\mathcal{M})$. Formulated in such a way, the DR can also be viewed as a special case of meta-RL, where the meta-policy is trained to maximize the *zero-shot* performance $K = 0$ on individual tasks.

Algorithm 2 MAML algorithm for reinforcement learning [195]

Require: $p(\mathcal{M})$ distribution of tasks, (α, β) learning rates
1: Initialize the meta-policy parameters $\boldsymbol{\theta}$
2: **for** $t = 1, 2, \ldots$ **do**
3: Sample a batch of tasks $\mathcal{M}_i \sim p(\mathcal{M})$.
4: **for** each task $\mathcal{M}_i$ **do**
5: Sample K trajectories D using the policy $\pi_{\boldsymbol{\theta}}$
6: Compute adapted task policies $\boldsymbol{\theta}_i' \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\pi_{\boldsymbol{\theta}})$
7: Sample new trajectories $\mathcal{D}_i'$ with adapted policy $\pi_{\boldsymbol{\theta}_i'}$ in task $\mathcal{M}_i$
8: **end for**
9: Update meta-policy $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{M}_i \sim p(\mathcal{M})} \mathcal{J}_{\mathcal{M}_i}(\pi_{\boldsymbol{\theta}_i'})$ with each meta-trajectories $\mathcal{D}_i'$ and objective $\mathcal{J}_{\mathcal{M}_i}$
10: **end for**

One of the most successful meta-learning algorithms is model agnostic meta-learning (MAML) [195]. The algorithm is outlined in Algorithm 2. It trains a meta-policy $\pi_{\boldsymbol{\theta}}$ and performs the updates on the inner loop using a policy gradient algorithm. When adapting to the new task $\mathcal{M}_i$, the meta-policy is used as the initial policy to interact

with the environment, obtaining trajectories D. The policy parameters can be adapted by estimating the gradient $\nabla_\theta \mathcal{J}(\pi_\theta)$ using the returns

$$\theta_i' \leftarrow \theta + \alpha \nabla_\theta \mathcal{J}(\pi_\theta), \tag{6.23}$$

where α is the inner-loop learning rate. After the updates, the adapted task policy $\pi_{\theta_i'}$ is used to sample new trajectories D_i'. This is repeated for all the sampled tasks. The meta-policy is then updated using the averaged returns over all the tasks

$$\theta \leftarrow \theta + \beta \nabla_\theta \sum_{\mathcal{M}_i \sim p(\mathcal{M})} \mathcal{J}_{\mathcal{M}_i}(\pi_{\theta_i'}), \tag{6.24}$$

with β being the outer-loop learning rate. The outer-loop update is again performed using a gradient-based algorithm.

In particle accelerator tuning, meta-RL can be applied to train adaptable RL agents with task variations at different levels. First, the task distribution $p(\mathcal{M})$ could be the simulation and the real-world tuning tasks. It can be used to model unknown parameters in real-world particle accelerators, allowing the agent to overcome the sim2real gap with a few adaptation steps. The task distribution can also contain the beam tuning tasks in different lattices, such as the case in the lattice-agnostic RL study. Furthermore, the distribution of tasks can represent the different working points in the same accelerator section. One example is the beam trajectory tuning task at the AWAKE accelerator at CERN [35]. Below the AWAKE tuning task using meta-RL is shortly discussed, with more details given in [176, 102].

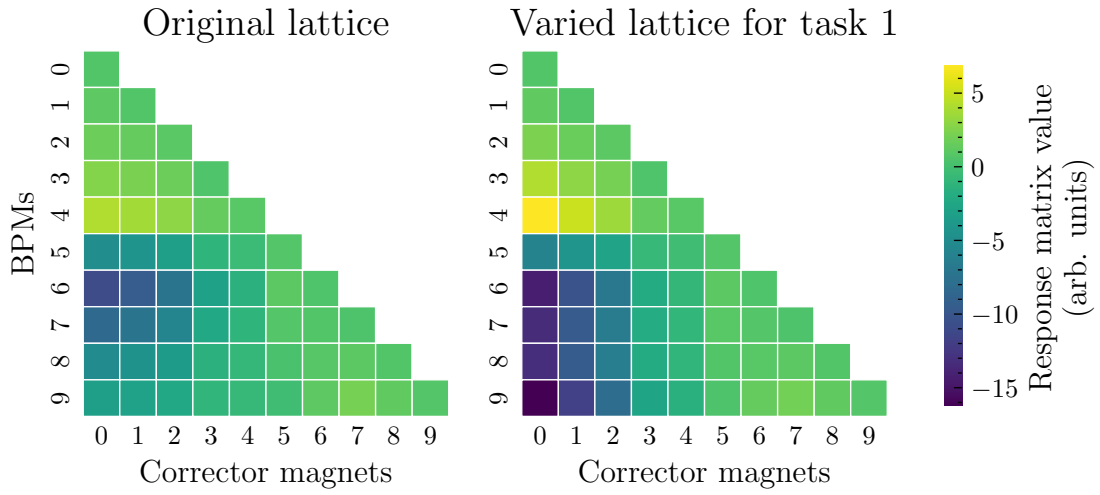


Figure 6.14.: Example tasks visualization for the AWAKE beam tuning task. The tasks are defined by the response matrices of the horizontal corrector magnets (steerers) with respect to the BPM readings. Varying the quadrupole settings along the beamline results in different response matrices, and different RL tasks.

The AWAKE tuning task uses in total of 10 steering magnets to minimize the differences between the beam trajectory and the target trajectory, using measurements from 10 BPMs.

In this case, only the horizontal plane is considered. The task can be easily formulated as an RL task, where the observations $\boldsymbol{o} = \boldsymbol{s} = (s_1, \cdots, s_{10})$ are the readings from the 10 BPMs, and actions $\boldsymbol{a} = (a_1, ..., a_{10})$ are the strengths of the corrector magnets. The reward is defined to be proportional to the negative root mean square (RMS) of the beam positions $r \propto ||\boldsymbol{s}||_2$. The dynamics of the system are defined by the horizontal orbit response matrix

$$\Delta s_i = \sum_j \mathcal{R}_{ij} \Delta a_j, \tag{6.25}$$

where Δa_j denotes the change in the j-th corrector magnet strength, and Δs_i denotes the change in the beam position measured by the i-th BPM. As the AWAKE is a linear accelerator and the BPMs and corrector magnets are placed along the beamline, the response matrix $\mathcal{R}$ is a lower-triangle matrix. The values of the response matrix elements, however, depend on the quadrupole magnets' strengths, which are in turn defined by various working points at AWAKE. Two of such response matrices are visualized in Fig. 6.14. By varying the quadrupole strengths, the impact of the corrector magnet changes, especially for the ones at the beginning of the beamline. Meta-RL can be applied in this case to train a meta-policy, which can quickly adapt to a new working point and perform the orbit correction.

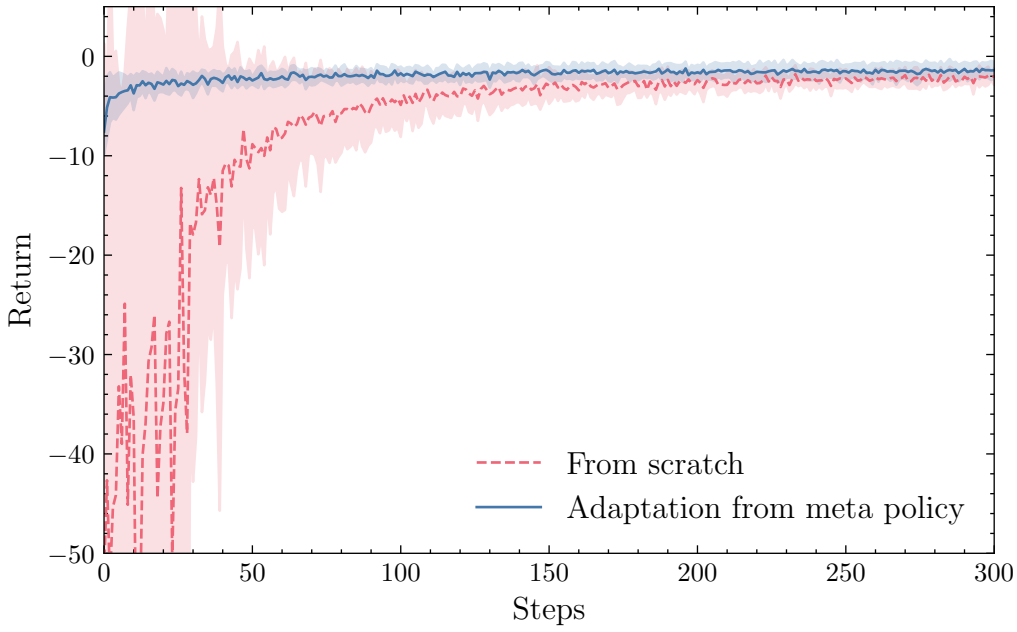


Figure 6.15.: Adapting the RL policy to new tasks using meta-trained policy and random policy on the AWAKE beam tuning task. The lines show returns averaged over 5 evaluation tasks, with the shaded region being the standard deviation.

MAML is applied for the meta-policy training. The inner-loop policy updates are performed using the REINFORCE [57] algorithm and the outer-loop meat-policy updates using the trust region policy optimization (TRPO) [87]. The quadrupole strengths are varied by ±25 % during the training phase, covering the realistic range of AWAKE operation.

The meta-training was performed for 100 steps, each time with inner-loop updates over 8 sampled tasks. For each task in the inner-loop, 16 episodes are sampled with a maximum of 50 steps in each episode. The training process can be performed on a common laptop with around 20 min computation time. Once the meta-policy is trained, it is evaluated on a fixed set of 5 tasks. The obtained return by the adapted policy is shown in Fig. 6.15. The returns are averaged over the 5 tasks, with the standard deviation shown as the shaded region. The meta-trained policy could achieve a good performance already zero-shot, and it quickly adapts to the task within a few steps, reaching an averaged return of -1.3 at the end. For comparison, a random initial policy is also tested. The adaptation process corresponds to a normal RL training. It initially has a much worse return compared to the meta-policy. The obtained return improves over training in about 200 steps. It converged to a final return of -1.9, which is still lower than the one from the meta-policy. To some extent, the meta-trained policy is more robust and performs better than the specialized agent training only on one task, even with the same total steps.

The comparison results show that meta-RL is a promising learning technique to train robust and adaptive RL agents, that can be applied efficiently on real-world particle accelerators and account for different working points, potential environmental changes, and drifts.

Lastly, it needs to be noted that the meta-RL and DR are not mutually exclusive ideas. In practice, one could also combine both techniques as they aim to tackle different issues. The meta-RL produces in the end an adapted agent, that is specialized on one instance $\mathcal{M}_i$ of the task distributions $p(\mathcal{M})$. By doing so, the adapted agent's ability to solve the other tasks in $p(\mathcal{M})$ decreases. Taking the ARES EA beam tuning task as an example, the misalignments of the components should be fixed or change only slowly over a longer period, whereas the incoming beam can change rather frequently on a day-to-day basis. When applying meta-RL to the ARES EA task, the adaptation should not specialize to one specific incoming beam. Otherwise, the RL agent will need to be trained repetitively each time the working condition is changed. Ideally, it should adapt to the real-world hardware configuration, such as misalignments, but remain robust against the uncertainties that can appear during operation. To achieve that, the training could apply both meta-RL and DR to different parts of the RL states. At the time of completing this dissertation, there is an ongoing collaborative effort to apply meta-RL to the ARES task and evaluate its performance against simple DR. When the lattice configurations are also considered in the meta-training, it is expected that the trained RL agent can be applied to solve the same beam-tuning task in different lattice sections at ARES, FLUTE, and other accelerators, requiring only a little amount of beamtime for adaptations.

6.4.3. Towards More Sample Efficient and Explainable Machine Learning-based Controller

The RL algorithms studied in this chapter are all *model-free* methods, which directly learn a policy through trial-and-error interactions with the environment. As most of the model-free methods are very sample inefficient, their successful applications rely on a large number of training samples, often in the range of millions. For some of the

accelerator tuning tasks shown above, the samples could be provided using the fast-executing Cheetah simulation model. However, such a fast simulation environment is not always available, for example in the SASE tuning task. If enough historical data is available, data-driven surrogate models such as NN could be trained, which can provide the required training samples in a computationally cheap way. Alternatively, one could use model-based methods for better sample efficiency. The simulation optimization and online tuning results with BO demonstrate that GPs are promising methods for modeling general function mappings with a small amount of data in the presence of measurement noise. It is also possible to combine the modeling strength of GP and the RL framework. This approach is called Gaussian process model predictive control (GP-MPC) [174]. A collection of GP models are used to approximate the system dynamics, i.e. the transition function

$$(\boldsymbol{s}_t, \boldsymbol{a}_t) \mapsto \boldsymbol{s}_{t+1}, \tag{6.26}$$

mapping the tuple of current states $\boldsymbol{s}_t$ and actions $\boldsymbol{a}_t$ to the next states $\boldsymbol{s}_{t+1}$. The GP models can be trained using the data obtained through interactions with the environment. Then, based on the predictions, model predictive control (MPC) can be applied to plan an optimal sequence of actions, maximizing the expected returns.

The GP-MPC was successfully applied to the AWAKE beam trajectory tuning task [101, 196, 102], solving the 10 degrees of freedom task within 20 steps. Its sample efficiency is more than three orders of magnitude higher than the one of model-free RL algorithms. Compared to the usual BO algorithms, GP-MPC is explicitly designed for online control and can more easily adapt to the changing conditions in real-world systems.

Furthermore, the trained GP models can also provide more insight into the system dynamics, analogous to what is shown in Section 4.1 and Section 6.1. It allows reasoning about the decision-making process during the operation, opening doors to building more explainable ML controllers. In conclusion, it proves to be a highly promising technique for training extremely data-efficient, robust, and adaptive controllers for accelerator operations.

6.5. Summary Machine Learning-based Online Accelerator Tuning

Online tuning is an essential component of particle accelerator operations. The accelerator tuning task is very challenging due to the high number of actuators, non-linear system dynamics, noisy or erroneous measurements, and time dependencies. Automatic tuning methods are required to efficiently perform optimizations to minimize downtime and maintain peak performance during operation. Classical numerical optimization algorithms and manual tuning often suffer from slow convergence and struggle with uncertainties, local optima, and drifting systems.

Two methods among the emerging ML-based algorithms are particularly of interest for designing novel optimizers and controllers, namely BO and RL. They are both investigated in this chapter using real-world accelerator tasks.

On the one hand, BO is a powerful technique for efficient parameter optimization problems with noisy observations. It was applied to the SASE tuning task at EuXFEL. Even without any previous knowledge, it reached a comparable final result to the one from the routinely used simplex optimizer. BO also demonstrated a smoother convergence behavior with fewer steps and could provide an informative model using the obtained data.

On the other hand, RL trains an intelligent control policy through interactions with an environment. We showed that the common accelerator optimization task, such as the FLUTE THz tuning task, can also be formulated within the RL framework. The fast-executing Cheetah simulation code provides enough samples for training the RL agent, which can subsequently solve the accelerator tuning task reliably and efficiently.

The two ML methods were benchmarked on the transverse tuning task at the ARES EA section. This marks the first comparison study of this kind in particle accelerators. In the simulation studies, both approaches significantly outperformed other baseline tuning methods like simplex and ES. When applied at the real-world accelerator, they obtained final beam MAEs of 24 - 44 µm, around the effective resolution limit of the diagnostic setup. The RL agent with a pre-trained policy reached overall the best performance in terms of the final beam parameters and number of steps required for convergence. It is also robust to erroneous measurements and time-dependent systems. In cases where the task needs to be routinely performed and enough training data is available, RL controllers should be used. BO has the advantage that it can be applied to new tasks with very little engineering effort. It is more prone to erroneous measurements if the GP model is completely learned with online data. With task-specific modifications such as the contextual BO, it can also deal with time-dependent systems and act as an adaptive control.

By extending the idea of domain randomization (DR), robust RL agents were trained that can be transferred to lattice sections at different accelerators, solving the same transverse beam tuning task. Meta-RL improves further on the DR technique and explicitly treats the various lattice settings as different tasks. At the AWAKE accelerator, the meta-trained RL agent can adapt to a new working point with only a few episodes and solve the beam trajectory tuning task. Future ML controllers are expected to leverage the strengths of RL and BO. GP-MPC is a promising avenue to build such a controller. The extreme sample efficiency allows it to be trained directly online at real accelerators. It also provides robust and adaptive control in time-dependent systems.

7. Summary and Outlook

Particle accelerators are among the most complex physical systems in the world, comprising thousands of components and subsystems. The design and operation of a particle accelerator are faced with various challenging tasks. There is an ever-increasing demand for higher availability and better beam qualities such as high bunch charge, intensive radiation output, and beam stability for light sources with electron linear accelerators. In addition, as they are single-pass accelerators, tailored beam control and quick switching between working points are essential to meet the requirement of multiple beamline experiments. In the design stage, optimizations of the parameter settings and the layout need to be performed using physics simulations that are expensive to evaluate. This is typically a non-linear black-box optimization problem involving a large number of input parameters. Overall the design optimization is very time-consuming and computationally intensive, especially when the optimization needs to be repeated throughout the design process. In online operations, the simulated optimal settings are usually not directly achieved, due to mismatches between the simulation and real-world systems such as magnetic field errors and component misalignments. Therefore, optimization algorithms are often required for setting up the accelerator and maximizing its performance. In addition, the online tuning of particle accelerators often contains noisy measurements and time-dependent systems. Such tuning tasks also need to be performed regularly and with as little beamtime as possible to achieve maximum availability.

Many of these mentioned challenges can be mitigated using machine learning (ML) approaches. In the last decade, there has been an increasing interest in the ML methods in the accelerator community. Several proof-of-principle studies on the ML applications have been demonstrated by other people, such as speeding up the simulated optimization [108, 33, 16], providing additional measurements as virtual diagnostics [20, 19, 21], and efficient online tuning [64, 111]. It is believed that a paradigm shift will happen with the development and integration of ML methods, changing the way how the future accelerator will be operated. Most importantly, different ML methods will work in orchestration and increase the overall level of autonomy in the operation, such as utilizing the neural network (NN) surrogate model and fast-executing simulation as prior knowledge for online optimizers. Intending to provide a holistic study of the ML methods integrated into the life cycle of linear accelerators, I showed various applications in this dissertation, both in simulation and experiments. Some key contributions are summarized below.

- **Application of ML for simulated optimization and virtual diagnostics** [104, 105]. Using the low-energy section at the far-infrared linac and test experiment (FLUTE) accelerator as an example, I demonstrated that a trained NN surrogate model can provide fast, high-quality predictions in replacement of computationally expensive physics simulations. Such a surrogate model can also serve as a virtual

diagnostics to provide real-time non-destructive measurements during operations. In addition, I showed that the parallel version of Bayesian optimization (BO) can be employed for fast simulated optimization during the design stage, efficiently obtaining the optimal parameter settings using fewer computational resources.

- **Development of a fast-executing, backward-differentiable beam dynamics simulation code *Cheetah*** [42, 197]. Various use cases of such a differentiable simulation code are shown throughout the thesis. First, simulated optimization can be performed using the more sample-efficient gradient-based optimizes, which can be easily scaled up to a high number of input parameters. Second, the Cheetah simulation model can be combined with the BO algorithm as a prior-mean function for the Gaussian process (GP) model. This allows even more efficient parameter optimization in the presence of collective effects, which is still under development in Cheetah. Lastly, its execution speed also makes it the ideal candidate for building the environment and providing samples for the reinforcement learning (RL) agent training.

- **Feasibility study of convolutional neural network (CNN)-based spatial light modulator (SLM) control for laser pulse shaping** [129, 130]. Photo-injector laser shaping is an important step to increase the operational limit of bunch current at linear accelerators and provide tailored electron bunch for user experiments. The SLM is shown to be a promising solution to achieve fine-grained control of the laser pulse shape. At the FLUTE accelerator, two SLM setups were built for the transverse laser profile shaping. For the first time, it is demonstrated that the CNN can be combined with classical computer generated hologram (CGH) algorithms to improve the quality of the shaped laser pulse in the accelerators community. An alternative control method through Zernike polynomials is proposed, consisting of fewer degrees of freedom to be controlled and is more robust against errors during optical propagation. These approaches may lead to full spatial-temporal shaping setups using ML control in the future.

- **First detailed comparison of the Bayesian optimization (BO) and reinforcement learning (RL) for online accelerator tuning** [76, 95]. At the European XFEL, I demonstrated that the BO is mature enough to be used as a turn-key tuning solution, delivering comparable performance as other routinely used optimizers like the simplex method. This dissertation showed in detail how common accelerator tuning tasks can be formulated as BO and RL tasks respectively. This includes a workflow for algorithm implementation and deployment, which applies generally to many accelerator tasks. Using the accelerator research experiment at SINBAD (ARES) beam tuning task as a benchmark, the performance of BO and RL in simulation and experiments are systematically evaluated. The result shows that, when enough upfront engineering effort can be justified, RL should be the preferred algorithm for particle tuning tasks. It outperforms BO and other commonly used optimization algorithms in terms of final tuning results and robustness against time-dependent systems. In comparison, BO is ideal for tasks where no fast simulation environment or not enough data is available. Its versatility and efficiency make it an extremely

promising algorithm in the toolbox of accelerator tuning methods. In addition, I demonstrated how task-specific adaptions can further improve BO's performance, such as utilizing the Cheetah simulation as a physics-informed prior and adding contextual parameters to handle time-dependent systems.

- **Proof-of-principle studies of universal and sample-efficient RL-based controllers** [175, 176, 102, 196]. Based on the insights obtained from the BO and RL comparison study, several extensions of the RL algorithm are further explored to build transferable and more sample-efficient RL agent. The study showed that the domain randomization technique is not only useful for transferring the simulation-trained RL agent to real-world systems but also helpful in training agents that can be applied to different accelerators. This idea is extended and generalized by the meta-RL, explicitly taking into account the different lattices. The agent trained using meta-RL is expected to adapt to specific accelerator tuning tasks, requiring only a small amount of beamtime during operation. This allows the RL controllers to be deployed at different facilities at low engineering costs. The GP-MPC is proposed as a promising candidate to combine the strengths of BO and RL, which can be learned entirely using online interactions with the particle accelerators for the beam trajectory tuning tasks.

It is worth mentioning that although the applications shown in this dissertation are all at linear electron accelerators, most of the developed methods and frameworks can be directly transferred to similar tasks at any other kind of accelerator.

The works in this dissertation have been presented at numerous workshops and conferences, which helped raise interest and awareness for ML-based methods in the particle accelerator community. Notably, fruitful collaborations were formed during this dissertation. For instance, the first review article [64] on BO algorithms applied to particle accelerators was written during this work. It provided both an extensive overview of the methodology and details on the implementation details of BO for accelerator tuning tasks. Second, we established the Reinforcement Learning for Autonomous Accelerator (RL4AA) collaboration [198] and held two international workshops dedicated to the applications of RL at accelerators. At these workshops, we transformed the developed RL codes, like the ones used for the ARES tuning and the meta-RL, into hands-on tutorials [199, 196]. Our work on differentiable simulation Cheetah also gained interest among the community. Many researchers have started using the code and contributing to new feature development, including higher-order tracking maps, space charge effects, and phase-space reconstruction [197, 23].

During the period of this dissertation (end 2020 until 2024), the ML applications for particle accelerators are undergoing an important transition phase, moving from exploratory proof-of-principle works towards operational, routine tools that are made available in the accelerator control room. One notable example is the now widely adopted BO algorithms. New software packages were developed to provide the BO algorithms as read-to-use optimizers, such as Xopt [15] and Optimas [16], and provide an easy interface to the ML optimizers in the accelerator control room, such as Badger [17]. At the Karlsruhe research accelerator (KARA), Badger has been integrated into the control room, which is

now routinely used for commissioning and operation [200]. Looking forward, there is still much work to be done to make ML widely applicable and accessible for daily accelerator operations, apart from the development of general-purpose packages. For the BO, further improvements will come from better incorporating the physics-informed prior knowledge into the GP modeling. Whereas for RL, it is important to explore more approaches like meta-RL for generalization, lowering the upfront cost of training an RL agent and deploying at real-world accelerators. For both BO and RL, the safety aspects need to be better considered, for example using explicit uncertainty estimations [117].

Differentiable programming will undoubtedly be more investigated and developed in the future. While the field is still young and under-developed in the particle accelerator community, more impactful use cases will arise in addition to the already proposed ones [42, 197] for accelerators. As more physics effects are being integrated into the new generation of differentiable simulation codes, they can be expected to speed up the simulated optimization process and provide valuable online information guiding the accelerator operations.

In addition to the advancement and development of new ML methods, it is also important for the community to provide common frameworks, benchmark tasks, and baseline algorithms. As compared to other research fields like robotics, particle accelerators lack those classical benchmark tasks, mostly because each accelerator is built differently. However, this potentially hinders the collaborative effort and slows down the method improvement, as the merits and advantages of the methods might be obscure to people working on other tasks or applying other approaches. As a result, some methods with the potential to be transferred to similar tasks at other facilities, have only been applied to one specific task. This is also one motivation for the comparison study of BO and RL in this dissertation. The ARES beam tuning task studied in this dissertation was implemented in a modular way, with the hope of providing it as one benchmark task for the accelerator community. In addition, common accelerator tasks, such as the free electron laser (FEL) intensity tuning and photo-injector section tuning, can also be included in the benchmark tasks. It is expected that these efforts can spark future collaborations and speed up the development of ML methods tailored to accelerator tasks. Such a collection of benchmark accelerator tasks and environments provides a realistic test suite for the development of new ML methods. It also allows the new method to be tested on more tasks, demonstrating easily whether it would be applicable and beneficial for other tasks. A set of baseline algorithms with clean implementation, on the other hand, enables unbiased and quantified evaluation of the merits of a new ML approach.

In conclusion, the work presented in this dissertation has made significant contributions to the field of applying ML methods for the simulation and operation of particle accelerators. Further studies in these proposed directions will lead to a seamless integration of ML-algorithms into the operation scheme of future accelerators.

List of Figures

1.1. Timeline of X-ray free electron lasers in the world. 2
1.2. Overview of applications of ML methods in the operation scheme of electron linear accelerators. 6

2.1. The curvelinear coordinate system 10
2.2. Synchrotron radiation emitted by an electron passing through a bending magnet 15
2.3. Incoherent synchrotron radiation energy spectrum 15
2.4. Coherent synchrotron radiation spectrum 17
2.5. Working principle of a bunch compressor 18
2.6. Schematics overview of the FLUTE components 19
2.7. Schematic layout of the ARES accelerator 20
2.8. Simplified layout of the European XFEL facility. 20

3.1. Structure of perceptron 22
3.2. Activation functions 22
3.3. Visualization of Bayesian optimization steps. 25
3.4. GP prior and posterior visualizations. 27
3.5. Effects of the lengthscale hyperparameter on the GP model 28
3.6. Maximum log-likelihood fitting of the GP hyperparameters 29
3.7. Acquisition functions used in Bayesian optimization 30
3.8. Generalized policy iteration 37

4.1. Structure of the NN surrogate model for FLUTE low-energy section . . . 45
4.2. Surrogate model prediction error compared to ASTRA simulation results 47
4.3. Beam size prediction by the surrogate model and the ASTRA simulation results 48
4.4. Feature importance study of the surrogate model for the transverse beam size prediction 48
4.5. Correlation coefficients of the FLUTE low energy section 49
4.6. Surrogate predictions compared to real measurements of RF power scan at FLUTE 51
4.7. Surrogate predictions compared to real measurements of RF phase scan . 52
4.8. Penalized acquisition function for parallel Bayesian optimization 56
4.9. Longitudinal phase spaces of the optimized 1 pC bunches 58
4.10. THz optimization results of the 1 pC bunch 59
4.11. Longitudinal phase spaces of the optimized 100 pC bunches 60
4.12. THz optimization results of the 100 pC bunch 61
4.13. Differentiable simulation of a FODO lattice in Cheetah 63

4.14. Kernel density estimation for the longitudinal beam profile 65
4.15. Gradient-based optimization of the THz radiation using differentiable simulation Cheetah 66
4.16. Longitudinal phase space of the electron bunches 67
4.17. Tracking results with different physical effects activated 68
4.18. Effect of using a non-zero prior mean for the GP modeling. 69
4.19. Progress of BO using the physics-informed prior mean 70
4.20. Longitudinal phase spaces of the final beam from different methods . . . 71
4.21. Resulting CSR spectra and electric fields using different optimization methods 72

5.1. Transverse space charge fields for different radial beam distributions . . 76
5.2. Emittance growth of the beam in the RF photo-injector for different initial distributions 77
5.3. Schematic working principle of a spatial light modulator 79
5.4. Schematic of the FLUTE laser path layout 80
5.5. Progress of Gerchberg-Saxton algorithm 82
5.6. Transverse SLM Test Setup 83
5.7. Structure of the CNN 85
5.8. Generation and preprocessing of the CNN training data 86
5.9. Results of laser shaping with a convolutional neural network 87
5.10. SLM setup for FLUTE photo-injector Laser 88
5.11. Blazed grating pattern 89
5.12. Double electron bunches generated with SLM modulated laser 90
5.13. Modulated laser and electron bunches using Zernike polynomials 92

6.1. Measurement noise of the photon pulse energy at EuXFEL. 98
6.2. EuXFEL 2D tuning progress with BO and simplex 100
6.3. EuXFEL BO tuning with lower numbers of averaging shots 101
6.4. GP posterior visualization of the tuning task at EuXFEL. 102
6.5. Outline of the implemented reinforcement learning framework for accelerator tuning 105
6.6. Evaluation of the trained RL agent on OCELOT environment for FLUTE tuning 106
6.7. Simplified 3D illustration of the considered transverse beam tuning task at the ARES particle accelerator 108
6.8. Beam difference convergence plots for different optimization algorithms. 113
6.9. BO warm start results with historical data 115
6.10. One example optimization trial using Reinforcement Learning and Bayesian Optimization on the ARES accelerator 117
6.11. Using BO and RL for continuous control 121
6.12. Lattice sections studied for the lattice-agnostic reinforcement learning agent 124
6.13. Tuning result with the lattice agnostic RL agent 126
6.14. Example tasks visualization for the AWAKE beam tuning task 129
6.15. Adapting to new tasks using meta-policy and random policy on the AWAKE task 130

A.1. Feature importance and dependence of the surrogate model 170
A.2. Working principle of kernel density estimation 172
A.3. Kernel density estimation 172
A.4. Additional laser modulation results of the CNN-assisted laser shaping . . 173
A.5. Visualization of the Zernike polynomial 174
A.6. Modulated laser images using Zernike polynomials 175
A.7. Transformations of the Nelder-Mead simplex method 178

List of Acronyms

ABO adaptive BO

AD automatic differentiation

AI artificial intelligence

AOM acousto-optic modulator

ARES accelerator research experiment at SINBAD

BO Bayesian optimization

BPM beam position monitor

C-BO contextual BO

CDF cumulative density function

CERN European organization for nuclear research

CGH computer generated hologram

CNN convolutional neural network

CSR coherent synchrotron radiation

CTR coherent transition radiation

CW continuous wave

DDPG deep deterministic policy gradient

DC direct current

DESY Deutsches Elektronen-Synchrotron

DL deep learning

DQN deep Q-Learning

DR domain randomization

DRL deep reinforcement learning

EA Experimental Area

EI expected improvement

EIC Electron Ion Collider

ELU exponential linear unit

ER edge radiation

ES extremum seeking

EuXFEL European X-Ray Free-Electron Laser

FEL free electron laser

FLUTE far-infrared linac and test experiment

FPGA field-programmable gate array

FT Fourier transform

FFT fast Fourier transform

FWHM full width at half maximum

GA genetic algorithm

GP Gaussian process

GP-MPC Gaussian process model predictive control

GPU graphical processing unit

GS Gerchberg-Saxton

HPC high performance computing

IR infrared

ISR incoherent synchrotron radiation

KARA Karlsruhe research accelerator

KDE kernel density estimation

LC liquid crystal

LCLS Linac Coherent Light Source

LCoS liquid crystal on silicon

LHC large hadron collider

LHS latin hypercube sampling

linac linear accelerator

LSTM long short-term memory

MAE mean aboslute error

MAML model agnostic meta-learning

MC Monte-Carlo

MDP Markov decision process

ML machine learning

MLL marginal log likelihood

MLP multilayer perceptron

MPC model predictive control

MSE mean squared error

NN neural network

OOD out-of-distribution

PBO parallel Bayesian optimization

PDF probability density function

PINN physics-informed neural network

PITZ Photo Injector Test Facility at DESY in Zeuthen

POMDP partially observable Markov decision process

PPO proximal policy optimization

QE quantum efficiency

ReLU rectified linear unit

RBF radial basis function

RCDS robust conjugate direction search

RF radio frequency

RL reinforcement learning

RL4AA Reinforcement Learning for Autonomous Accelerator

RMS root mean square

RNN recurrent neural network

ROI region of interest

SAC soft actor-critic

SASE self-amplified spontaneous emission

SC space charge

SGD stochastic gradient descent

SHAP Shapley additive explanations

SLM spatial light modulator

TD temporal-difference

TD3 twin delayed deep deterministic policy gradient

THG third harmonic generation

Ti:Sa Titanium:Sapphire

TRPO trust region policy optimization

TPSA truncated power series algebra

UCB upper confidence bound

UV ultraviolet

YAG Yttrium Aluminum Garnet

ZOD zeroth order diffraction

List of Publications

This dissertation is largely based on the following publications.

- Jan Kaiser, **Chenran Xu**, Annika Eichler, Andrea Santamaria Garcia, Oliver Stein, Erik Bründermann, Willi Kuropka, Hannes Dinter, Frank Mayet, Thomas Vinatier, Florian Burkart, and Holger Schlarb. "Reinforcement learning-trained optimisers and Bayesian optimisation for online particle accelerator tuning". In: *Scientific Reports* 14.1 (2024), p. 15733. DOI: 10.1038/s41598-024-66263-y

- Jan Kaiser, **Chenran Xu**, Annika Eichler, and Andrea Santamaria Garcia. "Bridging the gap between machine learning and particle accelerator physics with high-speed, differentiable simulations". In: *Phys. Rev. Accel. Beams* 27 (5 May 2024), p. 054601. DOI: 10.1103/PhysRevAccelBeams.27.054601

- **Chenran Xu**, Erik Bründermann, Anke-Susanne Müller, Andrea Santamaria Garcia, and Sergey Tomin. "Bayesian Optimization for SASE Tuning at the European XFEL". in: *Proc. IPAC'23*. May 2023. DOI: 10.18429/JACoW-IPAC2023-THPL028

- **Chenran Xu**, Erik Bründermann, Anke-Susanne Müller, Andrea Santamaria Garcia, Jan Kaiser, and Annika Eichler. "Beam trajectory control with lattice-agnostic reinforcement learning". In: *Proc. IPAC'23*. May 2023. DOI: 10.18429/JACoW-IPAC2023-THPL029

- **Chenran Xu**, Erik Bründermann, Anke-Susanne Müller, Andrea Santamaria Garcia, Markus Schwarz, and Jens Schäfer. "Optimization Studies of Simulated THz Radiation at FLUTE". in: *Proc. IPAC'22*. July 2022. DOI: 10.18429/JACoW-IPAC2022-WEPOMS023

- **Chenran Xu**, Erik Bründermann, Anke-Susanne Müller, Andrea Santamaria Garcia, and Jens Schäfer. "Surrogate Modelling of the FLUTE Low-Energy Section". In: *Proc. IPAC'22*. July 2022, pp. 1182–1185. DOI: 10.18429/JACoW-IPAC2022-TUPOPT070

- Matthias Nabinger et al. "Transverse and Longitudinal Modulation of Photoinjection Pulses at FLUTE". in: *Proc. IPAC'22*. July 2022. DOI: 10.18429/JACoW-IPAC2022-TUPOPT068

- **Chenran Xu**, Erik Bründermann, Annika Eichler, Anke-Susanne Müller, Michael J. Nasse, Andrea Santamaria Garcia, Carl Sax, and Christina Widmann. "Machine Learning Based Spatial Light Modulator Control for the Photoinjector Laser at FLUTE". in: *Proc. IPAC'21*. Aug. 2021. DOI: 10.18429/JACoW-IPAC2021-WEPAB289

- Annika Eichler, Erik Bründermann, Florian Burkart, Jan Kaiser, Willi Kuropka, Andrea Santamaria Garcia, Oliver Stein, and **Chenran Xu**. "First Steps Toward an Autonomous Accelerator, a Common Project Between DESY and KIT". in: *Proc. IPAC'21*. Aug. 2021. DOI: 10.18429/JACoW-IPAC2021-TUPAB298

The following publications contain content related to this dissertation.

- **Chenran Xu**, Tobias Boltz, Akira Mochihashi, Andrea Santamaria Garcia, Marcel Schuh, and Anke-Susanne Müller. "Bayesian optimization of the beam injection process into a storage ring". In: *Phys. Rev. Accel. Beams* 26 (3 Mar. 2023), p. 034601. DOI: 10.1103/PhysRevAccelBeams.26.034601

- Ryan Roussel, Auralee L. Edelen, Tobias Boltz, Dylan Kennedy, Zhe Zhang, Fuhao Ji, Xiaobiao Huang, Daniel Ratner, Andrea Santamaria Garcia, **Chenran Xu**, Jan Kaiser, Angel Ferran Pousa, Annika Eichler, Jannis O. Lübsen, Natalie M. Isenberg, Yuan Gao, Nikita Kuklev, Jose Martinez, Brahim Mustapha, Verena Kain, Christopher Mayes, Weijian Lin, Simone Maria Liuzzo, Jason St. John, Matthew J. V. Streeter, Remi Lehe, and Willie Neiswanger. "Bayesian optimization algorithms for accelerator physics". In: *Phys. Rev. Accel. Beams* 27 (8 Aug. 2024), p. 084801. DOI: 10.1103/PhysRevAccelBeams.27.084801

- **Chenran Xu**, Edmund Blomley, Anke-Susanne Müller, Andrea Santamaria Garcia, and Merritt Zhang. "Integration of an Optimizer Framework into the Control System at KARA". in: *Proc. ICALEPCS'23*. Jan. 2024. DOI: 10.18429/JACoW-ICALEPCS2023-TUPDP030

- Ryan Roussel, Gregor Charleux, Auralee Edelen, Annika Eichler, Juan Pablo Gonzalez-Aguilera, Axel Huebl, Jan Kaiser, Remi Lehe, Andrea Santamaria Garcia, and **Chenran Xu**. "Advancements in backwards differentiable beam dynamics simulations for accelerator design, model calibration, and machine learning". In: *Proc. LINAC2024*. Aug. 2024, pp. 559–562. DOI: 10.18429/JACoW-LINAC2024-THPB068

- Simon Hirlaender, Lukas Lamminger, Sabrina Pochaba, Jan Kaiser, **Chenran Xu**, Andrea Santamaria Garcia, Luca Scomparin, and Verena Kain. "Towards few-shot reinforcement learning in particle accelerator control". In: *Proc. IPAC'24*. May 2024. DOI: 10.18429/JACoW-IPAC2024-TUPS60

- Simon Hirlaender, Sabrina Pochaba, Lamminger Lukas, Andrea Santamaria Garcia, **Chenran Xu**, Jan Kaiser, Annika Eichler, and Verena Kain. "Deep Meta Reinforcement Learning for Rapid Adaptation In Linear Markov Decision Processes: Applications to CERN's AWAKE Project". In: *Combining, Modelling and Analyzing Imprecision, Randomness and Dependence*. Cham: Springer Nature Switzerland, 2024, pp. 175–183. DOI: 10.1007/978-3-031-65993-5_21

- Luca Scomparin, Andrea Santamaria Garcia, Andreas Kopmann, Anke-Susanne Mueller, **Chenran Xu**, Edmund Blomley, Erik Bründermann, Johannes Steinmann, Jürgen Becker, Marcel Schuh, Michelle Caselle, Timo Dritschler, Akira Mochihashi,

and Marc Weber. "Preliminary results on the reinforcement learning-based control of the microbunching instability". In: *Proc. IPAC'24*. May 2024. DOI: 10.18429/JACoW-IPAC2024-TUPS61

- Andrea Santamaria Garcia, Luca Scomparin, **Chenran Xu**, Simon Hirlaender, Sabrina Pochaba, Annika Eichler, Jan Kaiser, and Michael Schenk. "The reinforcement learning for autonomous accelerators collaboration". In: *Proc. IPAC'24*. May 2024. DOI: 10.18429/JACoW-IPAC2024-TUPS62

- Andrea Santamaria Garcia, Erik Bründermann, Michelle Caselle, Giovanni De Carne, Anke-Susanne Müller, Luca Scomparin, and **Chenran Xu**. "How Can Machine Learning Help Future Light Sources?" In: *Proc. FLS'23*. 67. Jan. 2024. DOI: 10.18429/JACoW-FLS2023-TH3D3

- Luca Scomparin, Michele Caselle, Andrea Santamaria Garcia, **Chenran Xu**, Edmund Blomley, Timo Dritschler, Akira Mochihashi, Marcel Schuh, Johannes L. Steinmann, Erik Bründermann, Andreas Kopmann, Jürgen Becker, Anke-Susanne Müller, and Marc Weber. *Microsecond-Latency Feedback at a Particle Accelerator by Online Reinforcement Learning on Hardware*. under review. 2024. arXiv: 2409.16177 [physics.acc-ph]

Bibliography

[1] Andrew Sessler and Edmund Wilson. *Engines of Discovery*. Vol. 34. 6. WORLD SCIENTIFIC, Apr. 2014, pp. 38–41. ISBN: 978-981-4417-18-1. DOI: 10.1142/8552.

[2] Nanshun Huang et al. "Features and futures of X-ray free-electron lasers". In: *The Innovation* 2.2 (May 2021), p. 100097. ISSN: 26666758. DOI: 10.1016/j.xinn.2021.100097.

[3] P. Emma et al. "First lasing and operation of an ångstrom-wavelength free-electron laser". In: *Nature Photonics 2010 4:9* 4.9 (Aug. 2010), pp. 641–647. ISSN: 1749-4893. DOI: 10.1038/nphoton.2010.176.

[4] Tetsuya Ishikawa et al. "A compact X-ray free-electron laser emitting in the sub-ångström region". In: *Nature Photonics 2012 6:8* 6.8 (June 2012), pp. 540–544. ISSN: 1749-4893. DOI: 10.1038/nphoton.2012.141.

[5] Heung Sik Kang et al. "Hard X-ray free-electron laser with femtosecond-scale timing jitter". In: *Nature Photonics 2017 11:11* 11.11 (Oct. 2017), pp. 708–713. ISSN: 1749-4893. DOI: 10.1038/s41566-017-0029-8.

[6] W. Decking et al. "A MHz-repetition-rate hard X-ray free-electron laser driven by a superconducting linear accelerator". In: *Nature Photonics* 14.6 (June 2020), pp. 391–397. ISSN: 1749-4885. DOI: 10.1038/s41566-020-0607-z.

[7] Eduard Prat et al. "A compact and cost-effective hard X-ray free-electron laser driven by a high-brightness and low-energy electron beam". In: *Nature Photonics 2020 14:12* 14.12 (Nov. 2020), pp. 748–754. ISSN: 1749-4893. DOI: 10.1038/s41566-020-00712-8.

[8] Zhentang Zhao et al. "Status of the SXFEL Facility". In: *Applied Sciences 2017, Vol. 7, Page 607* 7.6 (June 2017), p. 607. ISSN: 2076-3417. DOI: 10.3390/APP7060607.

[9] Auralee Edelen et al. "Opportunities in Machine Learning for Particle Accelerators". In: (Nov. 2018).

[10] Andrea Santamaria Garcia et al. "How Can Machine Learning Help Future Light Sources?" In: *Proc. FLS'23*. 67. Jan. 2024. DOI: 10.18429/JACoW-FLS2023-TH3D3.

[11] C. Adolphsen et al. "European Strategy for Particle Physics – Accelerator R&D Roadmap". In: (Jan. 2022). DOI: 10.23731/CYRM-2022-001.

[12] S. Gourlay et al. "Snowmass'21 Accelerator Frontier Report". In: (Sept. 2022).

[13] S Biedron et al. "Snowmass21 Accelerator Modeling Community White Paper". In: (Mar. 2022).

[14] C. Allaire et al. “Artificial Intelligence for the Electron Ion Collider (AI4EIC)”. In: *Computing and Software for Big Science* 8.1 (Dec. 2024), pp. 1–21. ISSN: 25102044. DOI: 10.1007/S41781-024-00113-4/FIGURES/4.

[15] R Roussel et al. “XOPT: a simplified framework for optimization of accelerator problems using advanced algorithms”. In: *Proc. IPAC’23*. 2023. ISBN: 9783954502318. DOI: 10.18429/JACoW-IPAC2023-THPL164.

[16] A. Ferran Pousa et al. “Bayesian optimization of laser-plasma accelerators assisted by reduced physical models”. In: *Physical Review Accelerators and Beams* 26.8 (Aug. 2023), p. 084601. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.26.084601.

[17] Z Zhang et al. “BADGER: THE missing optimizer in ACR”. In: *Proc. IPAC’22*. 2022. ISBN: 9783954502271. DOI: 10.18429/JACoW-IPAC2022-TUPOST058.

[18] Nico Madysa. *Generic Optimisation Frontend and Framework (GeOFF)*.

[19] C. Emma et al. “Machine learning-based longitudinal phase space prediction of particle accelerators”. In: *Physical Review Accelerators and Beams* 21.11 (Nov. 2018), p. 112802. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.21.112802.

[20] J. Zhu et al. “High-Fidelity Prediction of Megapixel Longitudinal Phase-Space Images of Electron Beams Using Encoder-Decoder Neural Networks”. In: *Physical Review Applied* 16.2 (Aug. 2021), p. 024005. ISSN: 2331-7019. DOI: 10.1103/PhysRevApplied.16.024005.

[21] Alexander Scheinker et al. “An adaptive approach to machine learning for compact particle accelerators”. In: *Scientific Reports* 11.1 (Sept. 2021), p. 19187. ISSN: 2045-2322. DOI: 10.1038/s41598-021-98785-0.

[22] A. Wolski et al. “Transverse phase space tomography in an accelerator test facility using image compression and machine learning”. In: *Physical Review Accelerators and Beams* 25.12 (Dec. 2022), p. 122803. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.25.122803.

[23] R. Roussel et al. “Phase Space Reconstruction from Accelerator Beam Measurements Using Neural Networks and Differentiable Simulations”. In: *Physical Review Letters* 130.14 (Apr. 2023), p. 145001. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.130.145001.

[24] M. Kranjčević et al. “Multiobjective optimization of the dynamic aperture using surrogate models based on artificial neural networks”. In: *Physical Review Accelerators and Beams* 24.1 (Jan. 2021), p. 014601. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.24.014601.

[25] E. Fol et al. “Detection of faulty beam position monitors using unsupervised learning”. In: *Physical Review Accelerators and Beams* 23.10 (Oct. 2020), p. 102805. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.23.102805.

[26] G Azzopardi and G Ricci. “NEW MACHINE LEARNING MODEL APPLICATION FOR THE AUTOMATIC LHC COLLIMATOR BEAM-BASED ALIGNMENT”. In: *ICALEPCS21*. 2022. ISBN: 9783954502219. DOI: 10.18429/JACoW-ICALEPCS2021-THPV040.

[27] Sichen Li and Andreas Adelmann. “Time series forecasting methods and their applications to particle accelerators”. In: *Physical Review Accelerators and Beams* 26.2 (Feb. 2023), p. 024801. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.26.024801.

[28] J. Duris et al. “Bayesian Optimization of a Free-Electron Laser”. In: *Physical Review Letters* 124.12 (Mar. 2020), p. 124801. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.124.124801.

[29] Alexander Scheinker et al. “Model-independent tuning for maximizing free electron laser pulse energy”. In: *Physical Review Accelerators and Beams* 22.8 (Aug. 2019), p. 082802. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.22.082802.

[30] Johannes Kirschner et al. “Tuning particle accelerators with safety constraints using Bayesian optimization”. In: *Physical Review Accelerators and Beams* 25.6 (June 2022), p. 062802. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.25.062802.

[31] Chenran Xu et al. “Bayesian optimization of the beam injection process into a storage ring”. In: *Phys. Rev. Accel. Beams* 26 (3 Mar. 2023), p. 034601. DOI: 10.1103/PhysRevAccelBeams.26.034601.

[32] A. Awal et al. “Optimization of the injection beam line at the Cooler Synchrotron COSY using Bayesian Optimization”. In: *Journal of Instrumentation* 18.04 (Apr. 2023), P04010. ISSN: 1748-0221. DOI: 10.1088/1748-0221/18/04/P04010.

[33] Auralee Edelen et al. “Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems”. In: *Physical Review Accelerators and Beams* 23.4 (Apr. 2020), p. 044601. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.23.044601.

[34] Sören Jalas et al. “Bayesian Optimization of a Laser-Plasma Accelerator”. In: *Physical Review Letters* 126.10 (Mar. 2021), p. 104801. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.126.104801.

[35] Verena Kain et al. “Sample-efficient reinforcement learning for CERN accelerator control”. In: *Physical Review Accelerators and Beams* 23.12 (Dec. 2020), p. 124801. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.23.124801.

[36] Jan Kaiser, Oliver Stein, and Annika Eichler. “Learning-based Optimisation of Particle Accelerators Under Partial Observability Without Real-World Training”. In: *PMLR* 162 (2022), pp. 10575–10585.

[37] Weija Wang et al. “Accelerated Deep Reinforcement Learning for Fast Feedback of Beam Dynamics at KARA”. In: *IEEE Transactions on Nuclear Science* 68.8 (Aug. 2021), pp. 1794–1800. ISSN: 0018-9499. DOI: 10.1109/TNS.2021.3084515.

[38] Luca Scomparin et al. “Preliminary results on the reinforcement learning-based control of the microbunching instability”. In: *Proc. IPAC’24*. May 2024. DOI: 10.18429/JACoW-IPAC2024-TUPS61.

[39] Jason St. John et al. “Real-time artificial intelligence for accelerator control: A study at the Fermilab Booster”. In: *Physical Review Accelerators and Beams* 24.10 (Oct. 2021), p. 104601. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.24.104601.

[40] N Thuerey et al. “Physics-based Deep Learning”. In: (2022).

[41] Facundo Sapienza et al. “Differentiable Programming for Differential Equations: A Review”. In: (2024).

[42] Jan Kaiser et al. “Bridging the gap between machine learning and particle accelerator physics with high-speed, differentiable simulations”. In: *Phys. Rev. Accel. Beams* 27 (5 May 2024), p. 054601. DOI: 10.1103/PhysRevAccelBeams.27.054601.

[43] Andrzej Wolski. *Beam Dynamics in High Energy Particle Accelerators*. IMPERIAL COLLEGE PRESS, Apr. 2014. ISBN: 978-1-78326-277-9. DOI: 10.1142/p899.

[44] Helmut Wiedemann. *Particle Accelerator Physics*. Graduate Texts in Physics. Cham: Springer International Publishing, 2015. ISBN: 978-3-319-18316-9. DOI: 10.1007/978-3-319-18317-6.

[45] S Y Lee. *Accelerator Physics*. 4th. WORLD SCIENTIFIC, Jan. 2019. ISBN: 978-981-327-467-9. DOI: 10.1142/11111.

[46] Kevin Li. “Collective Effects – an introduction”. In: *Proceedings of the CERN-Accelerator-School course on Introduction to Accelerator Physics* (July 2021).

[47] Joseph Duris et al. “Tunable isolated attosecond X-ray pulses with gigawatt peak power from a free-electron laser”. In: *Nature Photonics* 14.1 (Jan. 2020), pp. 30–36. ISSN: 1749-4885. DOI: 10.1038/s41566-019-0549-5.

[48] Praveen Kumar Maroju et al. “Attosecond pulse shaping using a seeded free-electron laser”. In: *Nature* 578.7795 (Feb. 2020), pp. 386–391. ISSN: 0028-0836. DOI: 10.1038/s41586-020-2005-6.

[49] Alexander Wu Chao et al. *Handbook of Accelerator Physics and Engineering*. WORLD SCIENTIFIC, May 2013, pp. 1–830. ISBN: 978-981-4415-84-2. DOI: 10.1142/8543.

[50] M. J. Nasse et al. “FLUTE: A versatile linac-based THz source”. In: *Review of Scientific Instruments* 84.2 (Feb. 2013). ISSN: 0034-6748. DOI: 10.1063/1.4790431.

[51] M. J. Nasse et al. “First Electron Beam at the Linear Accelerator FLUTE at KIT”. In: *Proceedings of the10th International Particle Accelerator Conference (IPAC 2019), Melbourne, AUS, May 10-24, 2019. Ed.: M. Boland* 10 (2019), p. 882. DOI: 10.18429/JACOW-IPAC2019-MOPTS018.

[52] F Burkart et al. “The ARES Linac at DESY”. In: *Proceedings of LINAC2022* (Sept. 2022), pp. 691–694. ISSN: 2226-0366. DOI: 10.18429/JACOW-LINAC2022-THPOJO01.

[53] Annika Eichler et al. “First Steps Toward an Autonomous Accelerator, a Common Project Between DESY and KIT”. In: *Proc. IPAC’21*. Aug. 2021. DOI: 10.18429/JACoW-IPAC2021-TUPAB298.

[54] Sergey Tomin et al. “Undulator linear taper control at the European X-Ray Free-Electron Laser facility”. In: *Physical Review Accelerators and Beams* 27.4 (Apr. 2024), p. 042801. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.27.042801.

[55] Ian. Goodfellow, Yoshua. Bengio, and Aaron. Courville. *Deep learning*. The MIT Press, 2016, p. 775. ISBN: 9780262337373.

[56] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. ISBN: 9780262256834. DOI: 10.7551/mitpress/3206.001.0001.

[57] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[58] Ciyou Zhu et al. "Algorithm 778: L-BFGS-B". In: *ACM Transactions on Mathematical Software* 23.4 (Dec. 1997), pp. 550–560. ISSN: 0098-3500. DOI: 10.1145/279232.279236.

[59] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *NIPS'19*. 2019, pp. 8026–8037. DOI: 10.5555/3454287.3455008.

[60] Jason Ansel et al. "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation". In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. Vol. 2. New York, NY, USA: ACM, Apr. 2024, pp. 929–947. DOI: 10.1145/3620665.3640366.

[61] Martín Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: (Mar. 2016).

[62] Roy Frostig, Matthew James Johnson, and Chris Leary. "Compiling machine learning programs via high-level tracing". In: *Systems for Machine Learning*. 2018.

[63] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (Dec. 2014).

[64] Ryan Roussel et al. "Bayesian optimization algorithms for accelerator physics". In: *Phys. Rev. Accel. Beams* 27 (8 Aug. 2024), p. 084801. DOI: 10.1103/PhysRevAccelBeams.27.084801.

[65] Eric Brochu, Vlad M. Cora, and Nando de Freitas. *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*. Dec. 2010.

[66] Niranjan Srinivas et al. "Gaussian process optimization in the bandit setting: No regret and experimental design". In: *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*. 2010, pp. 1015–1022. ISBN: 9781605589077. DOI: 10.1109/TIT.2011.2182033.

[67] Donald R Jones, Matthias Schonlau, and William J Welch. "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492. ISSN: 09255001. DOI: 10.1023/A:1008306431147.

[68] Daniel James Lizotte. "Practical Bayesian Optimization". PhD thesis. University of Alberta, 2008, p. 177. ISBN: 978-0-494-46365-9.

[69] Adi Hanuka et al. "Physics model-informed Gaussian process for online optimization of particle accelerators". In: *Physical Review Accelerators and Beams* 24.7 (July 2021), p. 072802. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.24.072802.

[70] Favour M. Nyikosa, Michael A. Osborne, and Stephen J. Roberts. "Bayesian Optimization for Dynamic Problems". In: (Mar. 2018).

[71] Andreas Krause and Cheng Soon Ong. "Contextual Gaussian process bandit optimization". In: *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*. 2011. ISBN: 9781618395993.

[72] N Kuklev et al. "Online Accelerator Tuning with Adaptive Bayesian Optimization". In: *Proc. 5th Int. Particle Accel. Conf. (NAPAC'22)*. Oct. 2022, pp. 842–845. DOI: 10.18429/JACoW-NAPAC2022-THXD4.

[73] N Kuklev et al. "Robust adaptive Bayesian optimization". In: *Proc. IPAC'23*. 2023. ISBN: 9783954502318. DOI: 10.18429/JACoW-IPAC2023-THPL007.

[74] F. Irshad, S. Karsch, and A. Döpp. "Multi-objective and multi-fidelity Bayesian optimization of laser-plasma acceleration". In: *Physical Review Research* 5.1 (Jan. 2023), p. 013063. ISSN: 2643-1564. DOI: 10.1103/PhysRevResearch.5.013063.

[75] Ryan Roussel et al. "Turn-key constrained parameter space exploration for particle accelerators using Bayesian active learning". In: *Nature Communications* 12.1 (Sept. 2021), p. 5612. ISSN: 2041-1723. DOI: 10.1038/s41467-021-25757-3.

[76] Chenran Xu et al. "Bayesian Optimization for SASE Tuning at the European XFEL". In: *Proc. IPAC'23*. May 2023. DOI: 10.18429/JACoW-IPAC2023-THPL028.

[77] Johannes Kirschner et al. "Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces". In: *36th International Conference on Machine Learning, ICML 2019*. Vol. 2019-June. International Machine Learning Society (IMLS), 2019, pp. 5959–5971. ISBN: 9781510886988.

[78] Jannis O. Lübsen et al. "A Safe Bayesian Optimization Algorithm for Tuning the Optical Synchronization System at European XFEL". In: *IFAC-PapersOnLine* 56.2 (Jan. 2023), pp. 3079–3085. ISSN: 2405-8963. DOI: 10.1016/J.IFACOL.2023.10.1438.

[79] Ryan Roussel, Adi Hanuka, and Auralee Edelen. "Multiobjective Bayesian optimization for online accelerator tuning". In: *Physical Review Accelerators and Beams* 24.6 (June 2021), p. 062801. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.24.062801.

[80] S. Jalas et al. "Tuning curves for a laser-plasma accelerator". In: *Physical Review Accelerators and Beams* 26.7 (July 2023), p. 071302. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.26.071302.

[81] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 0028-0836. DOI: 10.1038/nature14236.

[82] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 0028-0836. DOI: 10.1038/nature16961.

[83] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (Oct. 2017), pp. 354–359. ISSN: 0028-0836. DOI: 10.1038/nature24270.

[84] Jonas Degrave et al. “Magnetic control of tokamak plasmas through deep reinforcement learning”. In: *Nature* 602.7897 (Feb. 2022), pp. 414–419. ISSN: 0028-0836. DOI: 10.1038/s41586-021-04301-9.

[85] Elia Kaufmann et al. “Champion-level drone racing using deep reinforcement learning”. In: *Nature* 620.7976 (Aug. 2023), pp. 982–987. ISSN: 0028-0836. DOI: 10.1038/s41586-023-06419-4.

[86] John Schulman et al. “High-Dimensional Continuous Control Using Generalized Advantage Estimation”. In: (June 2015).

[87] John Schulman et al. “Trust Region Policy Optimization”. In: *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, June 2015, pp. 1889–1897.

[88] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Mar. 2004. ISBN: 9780521833783. DOI: 10.1017/CBO9780511804441.

[89] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: (July 2017).

[90] Scott Fujimoto, Herke Van Hoof, and David Meger. “Addressing Function Approximation Error in Actor-Critic Methods”. In: (2018).

[91] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *35th International Conference on Machine Learning, ICML 2018* 5 (Jan. 2018), pp. 2976–2989.

[92] Niky Bruchon et al. “Basic Reinforcement Learning Techniques to Control the Intensity of a Seeded Free-Electron Laser”. In: *Electronics* 9.5 (May 2020), p. 781. ISSN: 2079-9292. DOI: 10.3390/electronics9050781.

[93] F. H. 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”. In: *Physical Review Accelerators and Beams* 23.12 (Dec. 2020), p. 122802. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.23.122802.

[94] Francesco Maria Velotti et al. “Towards automatic setup of 18 MeV electron beamline using machine learning”. In: *Machine Learning: Science and Technology* 4.2 (Apr. 2023), p. 025016. ISSN: 2632-2153. DOI: 10.1088/2632-2153/ACCE21.

[95] Jan Kaiser et al. “Reinforcement learning-trained optimisers and Bayesian optimisation for online particle accelerator tuning”. In: *Scientific Reports* 14.1 (2024), p. 15733. DOI: 10.1038/s41598-024-66263-y.

[96] Xiaolong Chen et al. “Orbit correction based on improved reinforcement learning algorithm”. In: *Physical Review Accelerators and Beams* 26.4 (Apr. 2023), p. 044601. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.26.044601.

[97] Xiaolong Chen et al. “Trend-Based SAC Beam Control Method with Zero-Shot in Superconducting Linear Accelerator”. In: (May 2023).

[98] Tobias Boltz. “Micro-Bunching Control at Electron Storage Rings with Reinforcement Learning”. PhD thesis. 2021. DOI: 10.5445/IR/1000140271.

[99] Luca Scomparin et al. *Microsecond-Latency Feedback at a Particle Accelerator by Online Reinforcement Learning on Hardware*. under review. 2024. arXiv: 2409.16177 [physics.acc-ph].

[100] Simon Hirlaender and Niky Bruchon. "Model-free and Bayesian Ensembling Model-based Deep Reinforcement Learning for Particle Accelerator Control Demonstrated on the FERMI FEL". In: (Dec. 2020). DOI: 10.48550/arxiv.2012.09737.

[101] Simon Hirlaender et al. "Ultra Fast Reinforcement Learning Demonstrated at CERN AWAKE". In: *Proceedings of the 14th International Particle Accelerator Conference*. 2023. DOI: 10.18429/JACoW-IPAC2023-THPL038.

[102] Simon Hirlaender et al. "Towards few-shot reinforcement learning in particle accelerator control". In: *Proc. IPAC'24*. May 2024. DOI: 10.18429/JACoW-IPAC2024-TUPS60.

[103] Klaus Floettmann. *ASTRA: A Space Charge Tracking Algorithm*.

[104] Chenran Xu et al. "Optimization Studies of Simulated THz Radiation at FLUTE". In: *Proc. IPAC'22*. July 2022. DOI: 10.18429/JACoW-IPAC2022-WEPOMS023.

[105] Chenran Xu et al. "Surrogate Modelling of the FLUTE Low-Energy Section". In: *Proc. IPAC'22*. July 2022, pp. 1182–1185. DOI: 10.18429/JACoW-IPAC2022-TUPOPT070.

[106] Rohit K. Tripathy and Ilias Bilionis. "Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification". In: *Journal of Computational Physics* 375 (Dec. 2018), pp. 565–588. ISSN: 00219991. DOI: 10.1016/j.jcp.2018.08.036.

[107] Lukáš Bajer, Zbyněk Pitra, and Martin Holeňa. "Benchmarking Gaussian Processes and Random Forests Surrogate Models on the BBOB Noiseless Testbed". In: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, July 2015, pp. 1143–1150. ISBN: 9781450334884. DOI: 10.1145/2739482.2768468.

[108] Jinyu Wan, Paul Chu, and Yi Jiao. "Neural network-based multiobjective optimization algorithm for nonlinear beam dynamics". In: *Physical Review Accelerators and Beams* 23.8 (Aug. 2020), p. 081601. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.23.081601.

[109] Ryan Sandberg et al. "Synthesizing Particle-In-Cell Simulations through Learning and GPU Computing for Hybrid Particle Accelerator Beamlines". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. Vol. 1. New York, NY, USA: ACM, June 2024, pp. 1–11. DOI: 10.1145/3659914.3659937.

[110] F. Mayet et al. "Predicting the transverse emittance of space charge dominated beams using the phase advance scan technique and a fully connected neural network". In: *Physical Review Accelerators and Beams* 25.9 (Sept. 2022), p. 094601. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.25.094601.

[111] Alexander Scheinker et al. “Demonstration of Model-Independent Control of the Longitudinal Phase Space of Electron Beams in the Linac-Coherent Light Source with Femtosecond Resolution”. In: *Physical Review Letters* 121.4 (July 2018), p. 044801. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.121.044801.

[112] Tobias Boltz et al. “More Sample-Efficient Tuning of Particle Accelerators with Bayesian Optimization and Prior Mean Models”. In: (Feb. 2024).

[113] Thorsten Hellert et al. “Application of deep learning methods for beam size control during user operation at the Advanced Light Source”. In: *Physical Review Accelerators and Beams* 27.7 (July 2024), p. 074602. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.27.074602.

[114] T Schmelzer et al. “DIAGNOSTICS AND FIRST BEAM MEASUREMENTS AT FLUTE”. In: *IPAC 2019* (2019). DOI: 10.18429/JACoW-IPAC2019-WEPGW010.

[115] Scott Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: (May 2017).

[116] T. Schmelzer et al. “Systematic Beam Parameter Studies at the Injector Section of FLUTE”. In: *12th International Particle Accelerator Conference* (Aug. 2021), pp. 2837–2839. ISSN: 2673-5490. DOI: 10.18429/JACOW-IPAC2021-WEPAB103.

[117] Moloud Abdar et al. “A review of uncertainty quantification in deep learning: Techniques, applications and challenges”. In: *Information Fusion* 76 (Dec. 2021), pp. 243–297. ISSN: 15662535. DOI: 10.1016/j.inffus.2021.05.008.

[118] Markus Schwarz et al. “Analytic calculation of the electric field of a coherent THz pulse”. In: *Physical Review Special Topics - Accelerators and Beams* 17.5 (May 2014), p. 050701. ISSN: 1098-4402. DOI: 10.1103/PhysRevSTAB.17.050701.

[119] Markus Schwarz et al. “Analytic calculation of electric fields of coherent THz pulses”. In: *IPAC’14*. July 2014. ISBN: 9783954501328. DOI: 10.18429/JACoW-IPAC2014-MOPRO067.

[120] I. Agapov et al. “OCELOT: A software framework for synchrotron light source and FEL studies”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 768 (Dec. 2014), pp. 151–156. ISSN: 01689002. DOI: 10.1016/j.nima.2014.09.057.

[121] Minghao Song et al. “Storage ring nonlinear dynamics optimization with multi-objective multi-generation Gaussian process optimizer”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 976 (Oct. 2020), p. 164273. ISSN: 01689002. DOI: 10.1016/j.nima.2020.164273.

[122] Javier González et al. “Batch Bayesian Optimization via Local Penalization”. In: (May 2015).

[123] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems* 2020-December (May 2020). ISSN: 10495258.

[124] J. Gonzalez-Aguilera et al. “Towards fully differentiable accelerator modeling”. In: *Proc. IPAC’23*. May 2023, pp. 2673–5490. DOI: 10.18429/JACoW-IPAC2023-WEPA065.

[125] Ji Qiang. "Differentiable self-consistent space-charge simulation for accelerator design". In: *Physical Review Accelerators and Beams* 26.2 (Feb. 2023), p. 024601. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.26.024601.

[126] J Wan et al. "Developing nested auto-differentiation tracking code for beam dynamics optimization". In: *Proc. IPAC'24*. 2024. ISBN: 9783954502479. DOI: 10.18429/JACoW-IPAC2024-WEPR59.

[127] Robbie Watt and Brendan O'Shea. "A differentiable simulation package for performing inference of synchrotron-radiation-based diagnostics". In: *Journal of Synchrotron Radiation* 31.2 (Mar. 2024), pp. 409–419. ISSN: 1600-5775. DOI: 10.1107/S1600577524000663.

[128] Karl L. Brown. "A First and Second Order Matrix Theory for the Design of Beam Transport Systems and Charged Particle Spectrometers". In: *Adv. Part. Phys.* (1968).

[129] Chenran Xu et al. "Machine Learning Based Spatial Light Modulator Control for the Photoinjector Laser at FLUTE". In: *Proc. IPAC'21*. Aug. 2021. DOI: 10.18429/JACoW-IPAC2021-WEPAB289.

[130] Matthias Nabinger et al. "Transverse and Longitudinal Modulation of Photoinjection Pulses at FLUTE". In: *Proc. IPAC'22*. July 2022. DOI: 10.18429/JACoW-IPAC2022-TUPOPT068.

[131] Luca Serafini and James B. Rosenzweig. "Envelope analysis of intense relativistic quasilaminar beams in rf photoinjectors:mA theory of emittance compensation". In: *Physical Review E* 55.6 (June 1997), pp. 7565–7590. ISSN: 1063-651X. DOI: 10.1103/PhysRevE.55.7565.

[132] O. J. Luiten et al. "How to Realize Uniform Three-Dimensional Ellipsoidal Electron Bunches". In: *Physical Review Letters* 93.9 (Aug. 2004), p. 094802. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.93.094802.

[133] Yuelin Li and John W. Lewellen. "Generating a Quasiellipsoidal Electron Beam by 3D Laser-Pulse Shaping". In: *Physical Review Letters* 100.7 (Feb. 2008), p. 074801. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.100.074801.

[134] Sol M. Gruner et al. "Energy recovery linacs as synchrotron radiation sources (invited)". In: *Review of Scientific Instruments* 73.3 (Mar. 2002), pp. 1402–1406. ISSN: 0034-6748. DOI: 10.1063/1.1420754.

[135] D. Filippetto et al. "Ultrafast electron diffraction: Visualizing dynamic states of matter". In: *Reviews of Modern Physics* 94.4 (Dec. 2022), p. 045004. ISSN: 0034-6861. DOI: 10.1103/RevModPhys.94.045004.

[136] Kwang-Je Kim. "Rf and space-charge effects in laser-driven rf electron guns". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 275.2 (Feb. 1989), pp. 201–218. ISSN: 01689002. DOI: 10.1016/0168-9002(89)90688-8.

[137] Feng Zhou et al. “Impact of the spatial laser distribution on photocathode gun operation”. In: *Physical Review Special Topics - Accelerators and Beams* 15.9 (Sept. 2012), p. 090701. ISSN: 10984402. DOI: 10.1103/PHYSREVSTAB.15.090701/FIGURES/7/MEDIUM.

[138] E.L. Saldin, E.A. Schneidmiller, and M.V. Yurkov. “Coherence properties of the radiation from X-ray free electron laser”. In: *Optics Communications* 281.5 (Mar. 2008), pp. 1179–1188. ISSN: 00304018. DOI: 10.1016/j.optcom.2007.10.044.

[139] G. Penco et al. “Experimental Demonstration of Electron Longitudinal-Phase-Space Linearization by Shaping the Photoinjector Laser Pulse”. In: *Physical Review Letters* 112.4 (Jan. 2014), p. 044801. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.112.044801.

[140] S. Bettoni et al. “Impact of laser stacking and photocathode materials on microbunching stability in photoinjectors”. In: *Physical Review Accelerators and Beams* 23.2 (Feb. 2020), p. 024401. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.23.024401.

[141] Randy Lemons et al. “Temporal shaping of narrow-band picosecond pulses via noncolinear sum-frequency mixing of dispersion-controlled pulses”. In: *Physical Review Accelerators and Beams* 25.1 (Jan. 2022), p. 013401. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.25.013401.

[142] Jared Maxson et al. “Adaptive electron beam shaping using a photoemission gun and spatial light modulator”. In: *Physical Review Special Topics - Accelerators and Beams* 18.2 (Feb. 2015), p. 023401. ISSN: 10984402. DOI: 10.1103/PHYSREVSTAB.18.023401/FIGURES/11/MEDIUM.

[143] Carmelo Rosales-Guzmán and Andrew Forbes. *How to Shape Light with Spatial Light Modulators*. SPIE, June 2017. ISBN: 9781510613010. DOI: 10.1117/3.2281295.

[144] S. Yu. Mironov et al. “Shaping of cylindrical and 3D ellipsoidal beams for electron photoinjector laser drivers”. In: *Applied Optics* 55.7 (Mar. 2016), p. 1630. ISSN: 0003-6935. DOI: 10.1364/AO.55.001630.

[145] James Good. “Experimental Characterization of the Photocathode Laser System for Advanced 3D Pulse Shaping at PITZ”. PhD thesis. 2019.

[146] C Koschitzki et al. “CHIRPED PULSE LASER SHAPING FOR HIGH BRIGHTNESS PHOTOINJECTORS”. In: (2022). DOI: 10.18429/JACoW-FEL2022-WEA04.

[147] Andreas Hoffmann et al. “Towards Implementation of 3D Amplitude Shaping at 515 nm and First Pulseshaping Experiments at PITZ”. In: *Photonics* 11.1 (Dec. 2023), p. 6. ISSN: 2304-6732. DOI: 10.3390/photonics11010006.

[148] Pierre Tournois. “Acousto-optic programmable dispersive filter for adaptive compensation of group delay time dispersion in laser systems”. In: *Optics Communications* 140.4-6 (Aug. 1997), pp. 245–249. ISSN: 00304018. DOI: 10.1016/S0030-4018(97)00153-3.

[149] A. M. Weiner. "Femtosecond pulse shaping using spatial light modulators". In: *Review of Scientific Instruments* 71.5 (May 2000), pp. 1929–1960. ISSN: 0034-6748. DOI: 10.1063/1.1150614.

[150] Yannick Schrödel et al. "Acousto-optic modulation of gigawatt-scale laser pulses in ambient air". In: *Nature Photonics* 18.1 (Jan. 2024), pp. 54–59. ISSN: 1749-4885. DOI: 10.1038/s41566-023-01304-y.

[151] Yuelin Li, Sergey Chemerisov, and John Lewellen. "Laser pulse shaping for generating uniform three-dimensional ellipsoidal electron beams". In: *Physical Review Special Topics - Accelerators and Beams* 12.2 (Feb. 2009), p. 020702. ISSN: 1098-4402. DOI: 10.1103/PhysRevSTAB.12.020702.

[152] A E Pollard et al. "Machine learning approach to temporal pulse shaping for the photoinjector laser at CLARA". In: *Proceedings of IPAC2022* (2022), pp. 2917–2920. DOI: 10.18429/JACoW-IPAC2022-THPOTK061.

[153] Futoshi Matsui et al. "Genetic-algorithm-based method to optimize spatial profile utilizing characteristics of electrostatic actuator deformable mirror". In: *Optical Review* 15.3 (May 2008), pp. 156–161. ISSN: 1340-6000. DOI: 10.1007/s10043-008-0025-9.

[154] G. Ha et al. "Bunch shaping in electron linear accelerators". In: *Reviews of Modern Physics* 94.2 (May 2022), p. 025006. ISSN: 0034-6861. DOI: 10.1103/RevModPhys.94.025006.

[155] Matthias Nabinger et al. "Efficient Terahertz Generation by Tilted-Pulse-Front Pumping in Lithium Niobate for the Split-Ring Resonator Experiment at FLUTE". In: *12th International Particle Accelerator Conference* (Aug. 2021), pp. 4299–4302. ISSN: 2673-5490. DOI: 10.18429/JACOW-IPAC2021-THPAB251.

[156] Eliot Bolduc et al. "Exact solution to simultaneous intensity and phase encryption with a single phase-only hologram". In: *Optics Letters* 38.18 (Sept. 2013), p. 3546. ISSN: 0146-9592. DOI: 10.1364/OL.38.003546.

[157] Ryoichi Horisaki, Ryosuke Takagi, and Jun Tanida. "Deep-learning-generated holography". In: *Applied Optics* 57.14 (May 2018), p. 3859. ISSN: 1559-128X. DOI: 10.1364/AO.57.003859.

[158] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9351 (2015), pp. 234–241. ISSN: 1611-3349. DOI: 10.1007/978-3-319-24574-4{_}28.

[159] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. 2015, pp. 448–456. DOI: 10.5555/3045118.3045167.

[160] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2323. ISSN: 00189219. DOI: 10.1109/5.726791.

[161] A. Marinelli et al. “Optical Shaping of X-Ray Free-Electron Lasers”. In: *Physical Review Letters* 116.25 (June 2016), p. 254801. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.116.254801.

[162] D. Cesar et al. “Electron beam shaping via laser heater temporal shaping”. In: *Physical Review Accelerators and Beams* 24.11 (Nov. 2021), p. 110703. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.24.110703.

[163] Marius Constantin Chirita Mihaila et al. “Transverse Electron-Beam Shaping with Light”. In: *Physical Review X* 12.3 (Sept. 2022), p. 031043. ISSN: 2160-3308. DOI: 10.1103/PhysRevX.12.031043.

[164] Xiaobiao Huang. *Beam-based Correction and Optimization for Accelerators*. Boca Raton: CRC Press, Dec. 2019. ISBN: 9780429434358. DOI: 10.1201/9780429434358.

[165] Xiaobiao Huang. “Robust simplex algorithm for online optimization”. In: *Physical Review Accelerators and Beams* 21.10 (Oct. 2018), p. 104601. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.21.104601.

[166] S Tomin et al. “Progress in automatic software-based optimization of accelerator performance”. In: *IPAC 2016 - Proceedings of the 7th International Particle Accelerator Conference*. 2016, pp. 3064–3066. ISBN: 9783954501472.

[167] Xiaobiao Huang et al. “An algorithm for online optimization of accelerators”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 726 (Oct. 2013), pp. 77–83. ISSN: 01689002. DOI: 10.1016/j.nima.2013.05.046.

[168] Xiaobiao Huang and James Safranek. “Online optimization of storage ring nonlinear beam dynamics”. In: *Physical Review Special Topics - Accelerators and Beams* 18.8 (Aug. 2015), p. 084001. ISSN: 1098-4402. DOI: 10.1103/PhysRevSTAB.18.084001.

[169] Alexander Scheinker, Xiaoying Pang, and Larry Rybarcyk. “Model-independent particle accelerator tuning”. In: *Physical Review Special Topics - Accelerators and Beams* 16.10 (Oct. 2013), p. 102803. ISSN: 1098-4402. DOI: 10.1103/PhysRevSTAB.16.102803.

[170] Peter I. Frazier. “A Tutorial on Bayesian Optimization”. In: (July 2018).

[171] David Meier et al. “Optimizing a superconducting radio-frequency gun using deep reinforcement learning”. In: *Physical Review Accelerators and Beams* 25.10 (Oct. 2022), p. 104604. ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.25.104604.

[172] Ke Li and Jitendra Malik. “Learning to Optimize”. In: (June 2016).

[173] Tianlong Chen et al. “Learning to Optimize: A Primer and A Benchmark”. In: *Journal of Machine Learning Research* 23 (2022), pp. 1–59.

[174] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. "Gaussian Processes for Data-Efficient Learning in Robotics and Control". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2 (Feb. 2015), pp. 408–423. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2013.218.

[175] Chenran Xu et al. "Beam trajectory control with lattice-agnostic reinforcement learning". In: *Proc. IPAC'23*. May 2023. DOI: 10.18429/JACoW-IPAC2023-THPL029.

[176] Simon Hirlaender et al. "Deep Meta Reinforcement Learning for Rapid Adaptation In Linear Markov Decision Processes: Applications to CERN's AWAKE Project". In: *Combining, Modelling and Analyzing Imprecision, Randomness and Dependence*. Cham: Springer Nature Switzerland, 2024, pp. 175–183. DOI: 10.1007/978-3-031-65993-5_21.

[177] S. Tomin, L. Fröhlich, and M. Scholz. "STATUS OF AUTOMATED OPTIMIZATION PROCEDURES AT THE EUROPEAN XFEL ACCELERATOR". In: *Proc. IPAC'19* (2019). DOI: 10.18429/JACoW-IPAC2019-TUZZPLM2.

[178] Maximilian Balandat et al. "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization". In: *Advances in Neural Information Processing Systems* 2020-December (Oct. 2019). ISSN: 10495258.

[179] Antonin Raffin, Jens Kober, and Freek Stulp. "Smooth Exploration for Robotic Reinforcement Learning". In: (May 2020).

[180] Siddharth Mysore et al. "Regularizing Action Policies for Smooth Control with Reinforcement Learning". In: (Dec. 2020).

[181] Mark Towers et al. *Gymnasium*. May 2024. DOI: 10.5281/zenodo.8127025.

[182] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[183] EPICS collaboration. *EPICS - Experimental Physics and Industrial Control System*.

[184] *DOOCS: The Distributed Object-Oriented Control System Framework*.

[185] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8.

[186] Eva Panofski et al. "Commissioning Results and Electron Beam Characterization with the S-Band Photoinjector at SINBAD-ARES". In: *Instruments* 5.3 (Aug. 2021), p. 28. ISSN: 2410-390X. DOI: 10.3390/instruments5030028.

[187] W. Kuropka et al. "Full PIC simulation of a first ACHIP experiment @ SINBAD". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 909 (Nov. 2018), pp. 193–195. ISSN: 01689002. DOI: 10.1016/j.nima.2018.02.042.

[188] F. Mayet et al. "Simulations and plans for possible DLA experiments at SINBAD". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 909 (Nov. 2018), pp. 213–216. ISSN: 01689002. DOI: 10.1016/j.nima.2018.01.088.

[189] Josh Tobin et al. “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World”. In: (Mar. 2017). DOI: 10.1109/IROS.2017.8202133.

[190] OpenAI et al. “Solving Rubik’s Cube with a Robot Hand”. In: (Oct. 2019).

[191] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020.

[192] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366. ISSN: 08936080. DOI: 10.1016/0893-6080(89)90020-8.

[193] Michael McCloskey and Neal J. Cohen. “Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem”. In: *Psychology of Learning and Motivation - Advances in Research and Theory*. Vol. 24. C. Academic Press, Jan. 1989, pp. 109–165. DOI: 10.1016/S0079-7421(08)60536-8.

[194] Jacob Beck et al. “A Survey of Meta-Reinforcement Learning”. In: (Jan. 2023).

[195] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *34th International Conference on Machine Learning, ICML 2017* 3 (Mar. 2017), pp. 1856–1868. DOI: 10.48550/arxiv.1703.03400.

[196] Simon Hirlaender et al. *Tutorial on Meta-Reinforcement Learning and GP-MPC at the RL4AA’24 Workshop*. Version v1.0.2. Mar. 2024. DOI: 10.5281/zenodo.10887397.

[197] Ryan Roussel et al. “Advancements in backwards differentiable beam dynamics simulations for accelerator design, model calibration, and machine learning”. In: *Proc. LINAC2024*. Aug. 2024, pp. 559–562. DOI: 10.18429/JACoW-LINAC2024-THPB068.

[198] Andrea Santamaria Garcia et al. “The reinforcement learning for autonomous accelerators collaboration”. In: *Proc. IPAC’24*. May 2024. DOI: 10.18429/JACoW-IPAC2024-TUPS62.

[199] Chenran Xu, Andrea Santamaria Garcia, and Jan Kaiser. *Tutorial on Applying Reinforcement Learning to the Particle Accelerator ARES*. Version v1.0.1. Mar. 2024. DOI: 10.5281/zenodo.10777477.

[200] Chenran Xu et al. “Integration of an Optimizer Framework into the Control System at KARA”. In: *Proc. ICALEPCS’23*. Jan. 2024. DOI: 10.18429/JACoW-ICALEPCS2023-TUPDP030.

[201] J. A. Nelder and R. Mead. “A Simplex Method for Function Minimization”. In: *The Computer Journal* 7.4 (Jan. 1965), pp. 308–313. ISSN: 0010-4620. DOI: 10.1093/comjnl/7.4.308.

[202] Alexander Scheinker. “100 years of extremum seeking: A survey”. In: *Automatica* 161 (Mar. 2024), p. 111481. ISSN: 00051098. DOI: 10.1016/j.automatica.2023.111481.

[203] Alexander Scheinker, En-Chuan Huang, and Charles Taylor. "Extremum Seeking-Based Control System for Particle Accelerator Beam Loss Minimization". In: *IEEE Transactions on Control Systems Technology* 30.5 (Sept. 2022), pp. 2261–2268. ISSN: 1063-6536. DOI: 10.1109/TCST.2021.3136133.

AI Assistance Disclosure

In the preparation of this dissertation, generative AI tools, including OpenAI's ChatGPT and Grammarly, were used to enhance writing style and to check spelling and grammar. All AI-generated content was carefully reviewed and edited, and I take full responsibility for the final content and conclusions presented in this dissertation.

A. Appendix

A.1. Feature importance study of the Surrogate Model

In Section 4.1, a surrogate model is trained for the low-energy section of FLUTE using a fully-connected neural network. Although the surrogate model can make fast predictions with high accuracy, it is still important to get insights into the model's decision-making process.

As the NN is a non-linear black-box model, simple methods like correlation analysis, which assumes a global linear dependency, will be biased to explain the trained model. Shapley additive explanations (SHAP) was proposed to provide a more accurate explanation of the model's decision-making process by estimating the Shapley value derived from cooperative game theory. In recent years, it has become a popular method to explain the predictions of NNs and other machine learning models. The SHAP values can be calculated separately for each of the output parameters. For each input parameter, they attribute to the change in the output prediction when conditioning on the input parameter. In other words, they quantify the importance of each input parameter to the model's prediction.

In Fig. A.1, the SHAP values for the surrogate model are visualized for the output beam parameters. The order of the input parameters is sorted by their importance to the model's prediction. The color of the points indicates the values of the input parameter, where red denotes a high (positive) value and blue is a low (negative) value. Their horizontal positions denote their impact on the respective output, i.e. a positive impact leads to a large output value. By analyzing the SHAP values, one can easily identify the most important input parameters for each output parameter and use these parameters for further optimization or online operation. For example, the bunch length is most sensitive to the gun phase. The gun gradient comes second and has a negative impact on the bunch length, which means a higher gun gradient leads to a shorter bunch length. The bunch energy is predominantly determined by the gun gradient while the energy spread is largely influenced by the gun phase, which determines the slope of the accelerating field that the electron bunch experiences. The percentage of the remaining particles is mostly influenced by the gun phase and insensitive to other incoming parameters. This is because particles may be lost in the photo-injector if the gun phase is not properly tuned. Lastly, it is visible that the beam emittance depends on all the input parameters, with the bunch charge being the most important one, due to the strong space charge effects in the low-energy section of the accelerator. This non-linear dependence also makes the beam emittance optimization for photo-injectors a challenging and recurring task in many accelerator facilities.

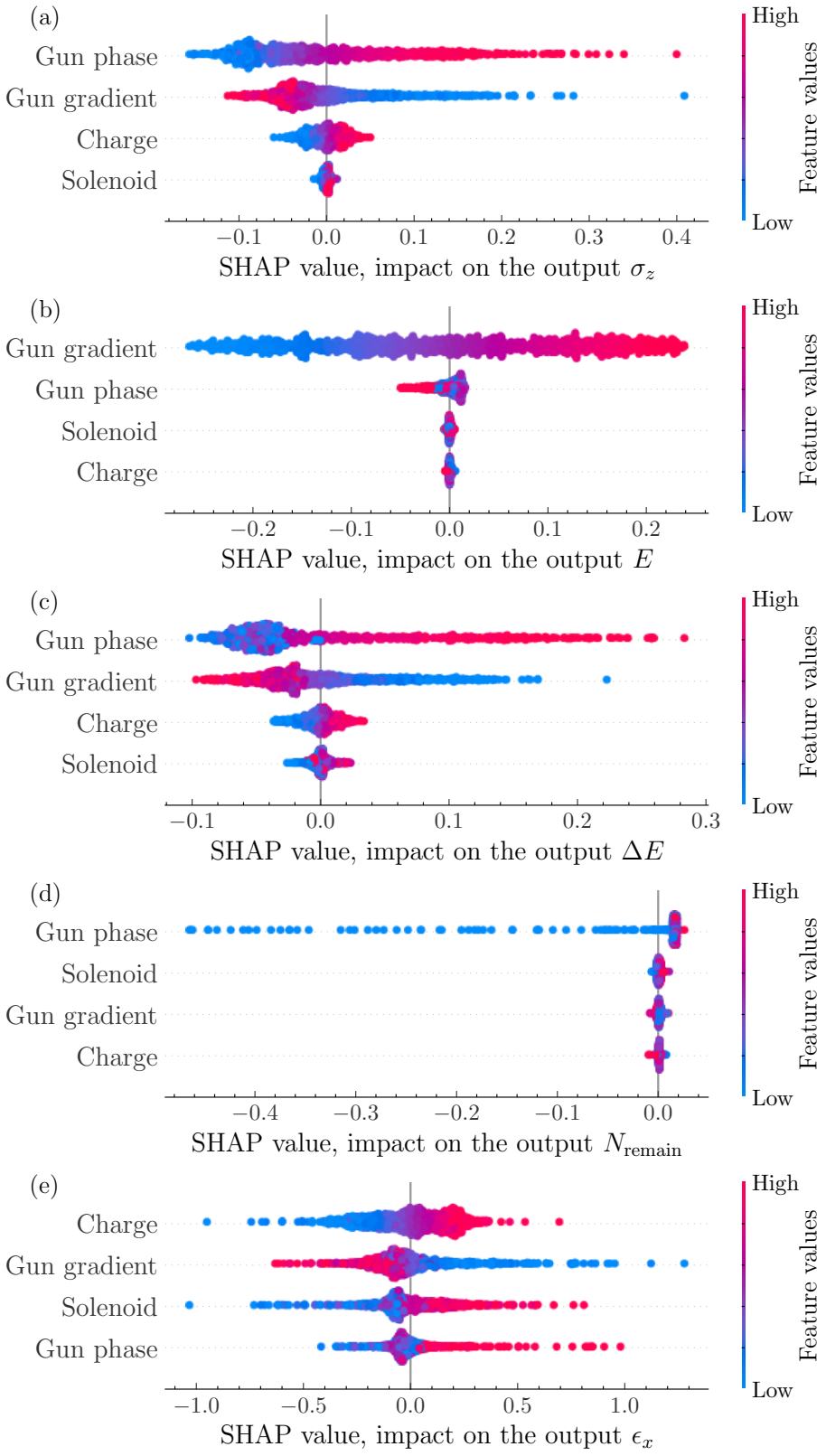


Figure A.1.: Feature importance and dependence of the surrogate model.

A.2. Kernel Density Estimation

In Section 4.3, kernel density estimation (KDE) is used instead of histogram as a technique to obtain the charge density profile from the macroparticles while maintaining the differentiability. Some more detailed explanations of this method are provided below.

kernel density estimation (KDE) is a non-parametric method to estimate the probability density function (probability density function (PDF)) of a random variable. Let $\{x_1, x_2, \ldots, x_n\}$ be a set of random samples drawn from a univariate distribution with an unknown PDF f_{PDF}. The KDE estimate of the PDF is given by

$$\hat{f}_{\mathrm{PDF}}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_i}{h}), \tag{A.1}$$

where K is the kernel function, and h is the bandwidth parameter. The kernel function is a non-negative function, similar to what is used in Bayesian optimization (BO). The most common choice is the Gaussian kernel, where $K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$. The bandwidth parameter h controls the smoothness of the estimated PDF. In the case of the Gaussian kernel, the bandwidth parameter becomes the standard deviation of the Gaussian distribution. In short, the kernel values are calculated for each sample, and the estimated PDF is the sum of all the individual kernels.

Since the estimated PDF is a continuous function, it remains differentiable, which is a prerequisite for the gradient-based optimization algorithms presented in the simulation study.

The KDE process is visualized in Fig. A.2. Here, a mixture of two Gaussian distributions is used as the underlying distribution

$$f_{\mathrm{PDF}}(x) = \frac{1}{3} \left(\mathcal{N}(\mu = -0.5, \sigma = 0.5) + 2 \cdot \mathcal{N}(\mu = 1.5, \sigma = 0.5)\right). \tag{A.2}$$

The purple lines show the values of the 100 samples drawn from the distribution. The dashed gray lines represent the kernel function values for each sample. The estimated PDF using the KDE is shown in red, which is the sum of all the individual kernels. As shown in the figure, the KDE can capture the double-peak structure of the underlying distribution. The comparison of the results obtained using KDE and the conventional histogram method is shown in Fig. A.3. The KDE result (red) demonstrates a smooth approximation compared to the discrete histogram values (blue), approximating the underlying PDF more closely.

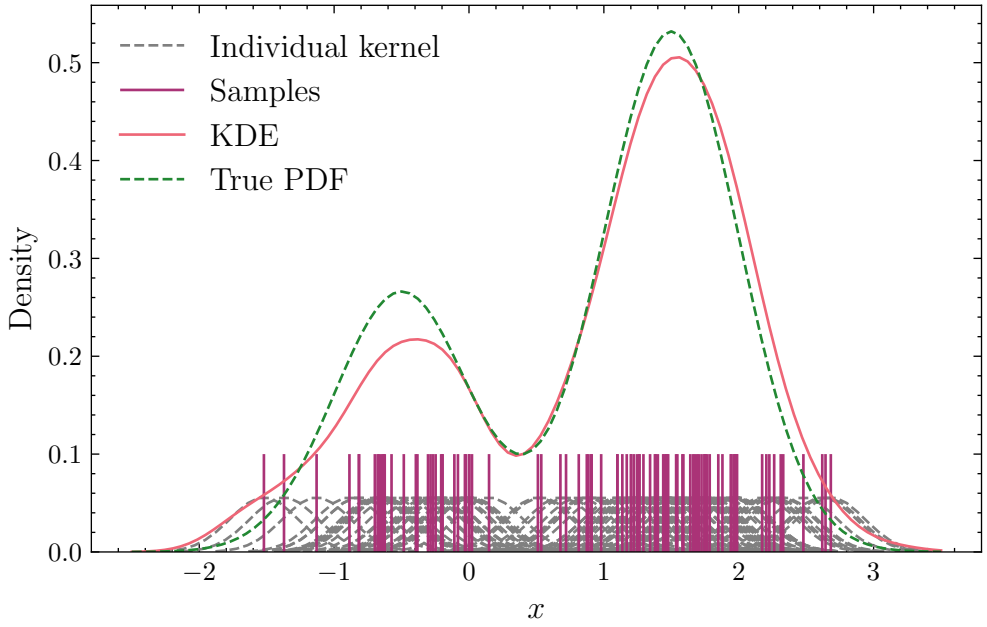


Figure A.2.: Working principle of KDE. Purple lines denote 100 samples obtained from the underlying PDF in green. For each sample, the kernel function value (gray) is calculated. The estimated PDF using the KDE (red) is the sum of all the individual kernels.

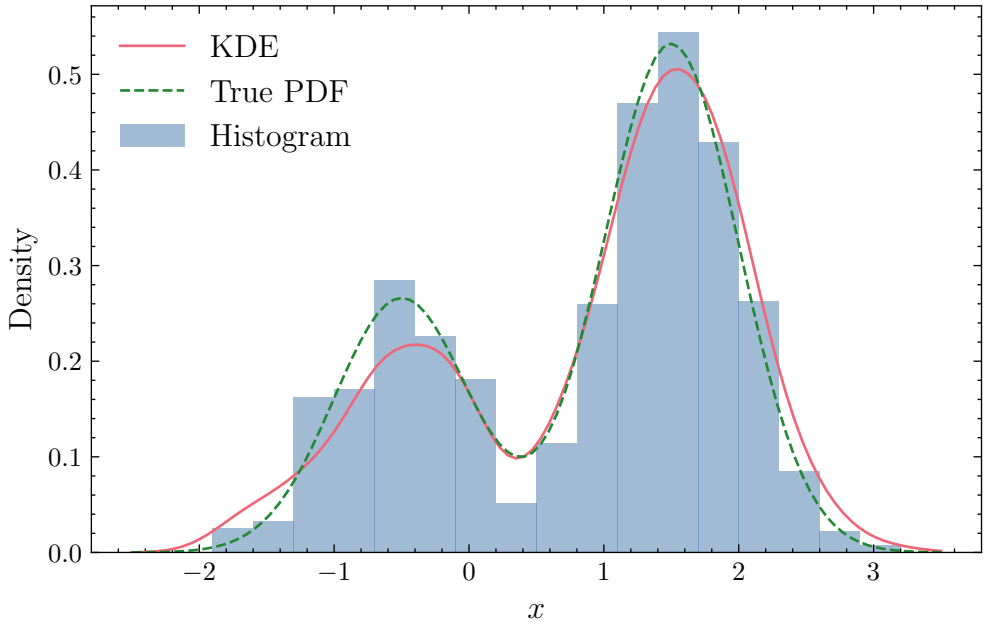


Figure A.3.: Density estimation using Gaussian KDE (red) and histogram method (blue). The random variable has a double peak structure and is distributed according to the PDF (dashed green).

A.3. Laser Modulation Results with Convolutional Neural Network

In Section 5.3, a CNN is trained to assist the transverse laser profile shaping using an SLM. Some additional results of the shaping results are shown in Fig. A.4, generating a digit, a Gaussian profile, and a flattop profile. Although the CNN corrected images show some improvement compared to the ones without correction, the modulation still shows visible discrepancies. For example, in the flattop profile, the zeroth-order diffraction is not fully compensated. Further improvement can be expected when scaling up the image resolution, using another NN structure, or controlling with physics-informed actuators like the Zernike polynomials as proposed in Section 5.5.

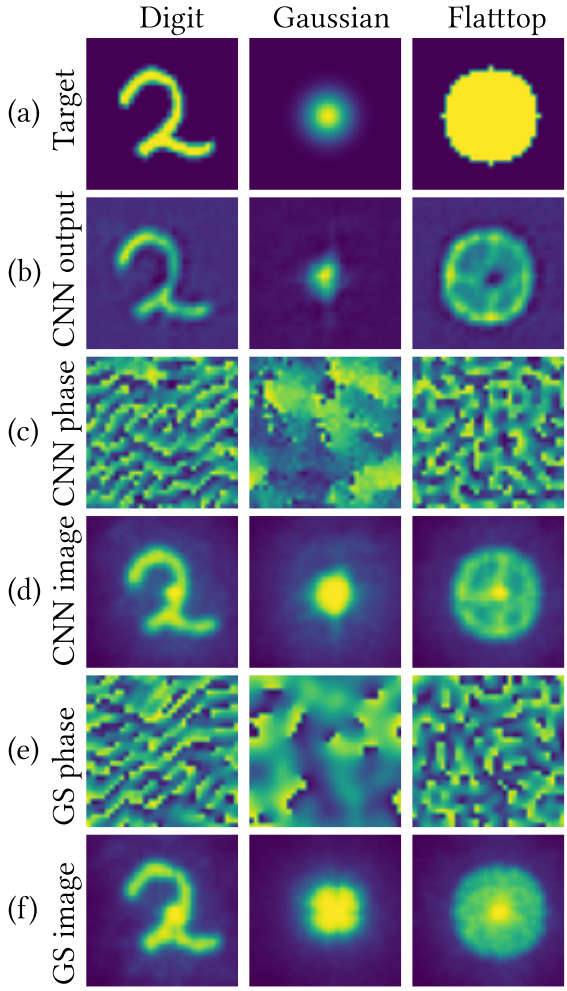


Figure A.4.: Experimental results of the CNN-assisted laser shaping. (a) Target patterns/input of the CNN: digit (left), Gaussian (middle), and flattop (right). (b) Output patterns of the CNN. (c) Holograms corresponding to the CNN predictions and (d) the camera-captured patterns. (e) GS-only holograms and (f) the captured images. Figure adapted from [129].

A.4. Laser Modulation with Zernike Polynomials

Zernike polynomials are a set of polynomials defined on a 2D unit disk in the Cartesian coordinate system. Fig. A.5 shows the first 10 terms of the Zernike polynomials. They are defined as

$$\begin{aligned} Z_n^m(\rho,\varphi) &= R_n^m(\rho)\cos m\varphi \\ Z_n^{-m}(\rho,\varphi) &= R_n^m(\rho)\cos m\varphi, \end{aligned} \tag{A.3}$$

where (ρ, φ) are the radius and azimuthal angle in the polar representation.

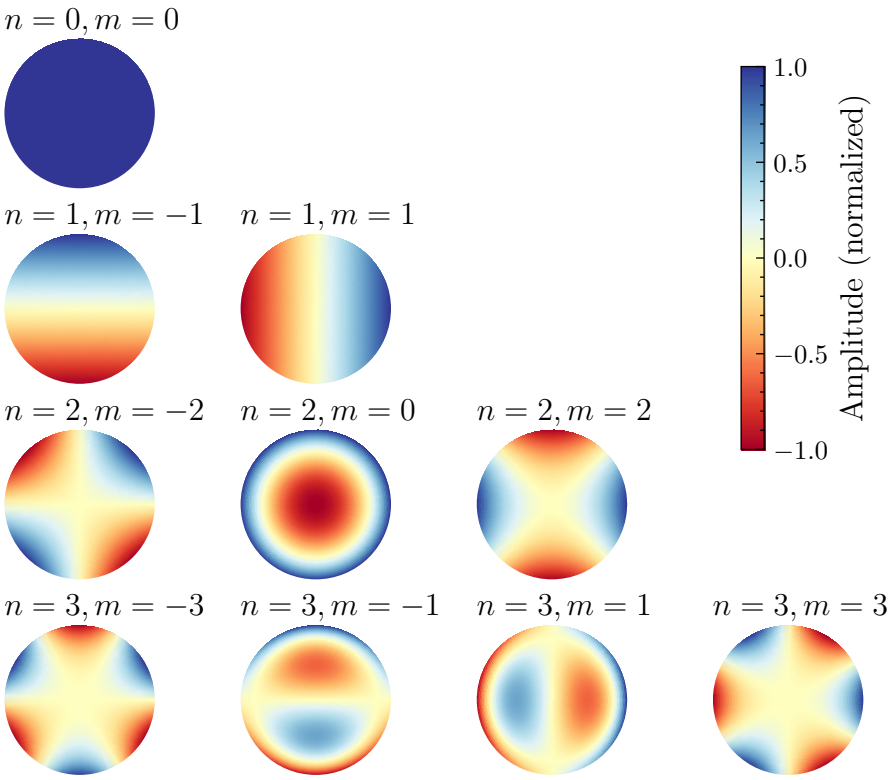


Figure A.5.: Visualization of the first 10 Zernike polynomials $n \in \{0, 1, 2, 3\}$.

The radial polynomials are defined as

$$R_n^m(\rho) = \begin{cases} \sum_{k=0}^{\frac{n-m}{2}} \frac{(-1)^k \cdot (n-k)!}{k! \cdot \left(\frac{n+m}{2}-k\right)! \cdot \left(\frac{n-m}{2}-k\right)!} \cdot \rho^{n-2k} & \text{, if } n-2m \in \mathbb{Z} \\ 0 & \text{, if } n-2m+1 \in \mathbb{Z} \end{cases} \tag{A.4}$$

They form an orthogonal basis over the unit disk and each Zernike polynomial corresponds to a specific aberration mode. Therefore, an arbitrary wavefront can be decomposed into a sum of Zernike polynomials. By neglecting higher-order terms, the dimensions of the actuators can be effectively reduced for wavefront correction using SLMs as discussed in Section 5.5. Figure A.6 shows additional experimental results of using individual Zernike terms for the laser shaping and the corresponding laser images at the cathode and the virtual cathode. The double spot for the $n = 0$ reference pulse on the virtual cathode was

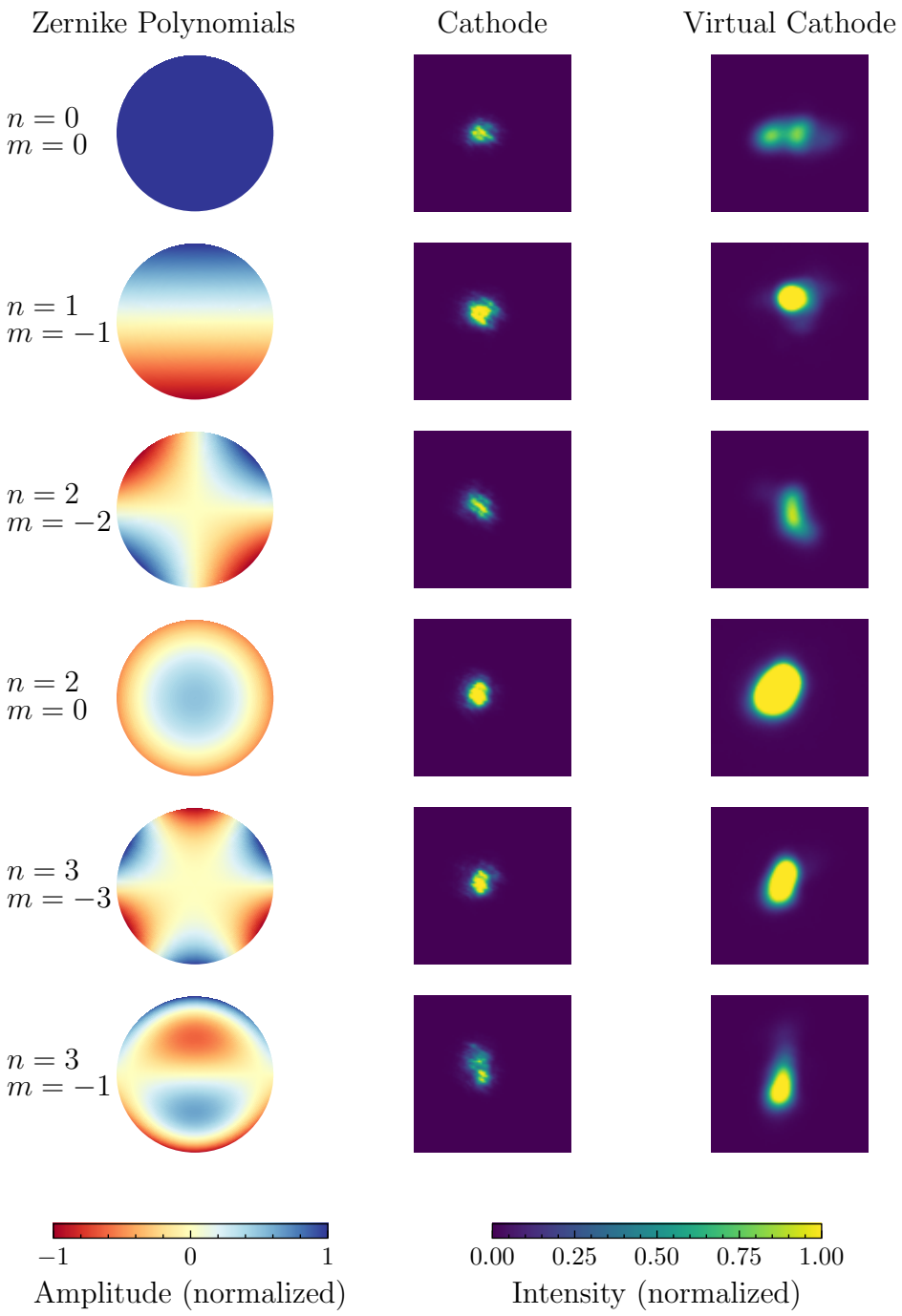


Figure A.6.: Modulated laser images using Zernike polynomials captured at the cathode and the virtual cathode.

not a result of the modulation, but an artifact due to the double reflection on the mirror coupling to the virtual cathode. The effects of different orders of the Zernike polynomial can be visually identified, such as the $n = 1$ term creates a tilt that shifts the pulse, the $n = 2, m = -2$ introduces astigmatism, and the $n = 2, m = 0$ term (de-)focuses the laser.

A.5. Non-Machine Learning Algorithms for Accelerator Tuning

A.5.1. Nelder-Mead Simplex

The Nelder-Mead simplex method [201] is an efficient model-free optimization algorithm. Due to its effectiveness, it has been applied widely for online accelerator optimization tasks where an accurate model is absent. In this dissertation, it is used as a baseline method for benchmarking the performance of the proposed ML methods. Without loss of generality, here a maximization task is considered. In this section, the method is briefly introduced. The outline of the Nelder-Mead simplex algorithm is shown in Algorithm 3, and the details of each step are described in the following part.

The algorithm is controlled by four hyperparameters, which are the coefficients of *reflection* (α), *expansion* (γ), *contraction* (β) and *shrinkage* (δ). These parameters satisfy the following conditions

$$\alpha > 0, \quad \gamma > 1, \quad 0 < \beta < 1, \quad 0 < \delta < 1 \quad . \tag{A.5}$$

Commonly, the following parametrization are used

$$(\alpha, \gamma, \beta, \delta) = \left(1, 2, \frac{1}{2}, \frac{1}{2}\right). \tag{A.6}$$

1. **Initialization**

   The Nelder-Mead algorithm is based on a non-degenerate simplex S, i.e. a convex polytope with $n+1$ vertices in the n-dimensional parameter space. The initial simplex can be chosen randomly. Commonly, the initial simplex is expanded around an initial input point x_1

   $$\boldsymbol{x}_{i+1} = \boldsymbol{x}_1 + l_i \cdot \boldsymbol{e}_i \,, \quad i \in \{1, \ldots, n\} \,, \tag{A.7}$$

   where $\boldsymbol{e}_i$ is the unit vector and l_i is the characteristic lengthscale of each input dimension.

2. **Ordering**

   The vertices of the simplex S are sorted and relabeled, so that

   $$f(\boldsymbol{x}_1) < f(\boldsymbol{x}_2) < \cdots < f(\boldsymbol{x}_{n+1}) \,, \tag{A.8}$$

   where $\boldsymbol{x}_1$ is the point with the least objective value and $\boldsymbol{x}_{n+1}$ is the best point.

3. **Centroid**

   The centroid $\boldsymbol{c}$ of all vertices except for $\boldsymbol{x}_1$ is calculated

   $$\boldsymbol{c} = \frac{1}{n} \sum_{i=2}^{n+1} \boldsymbol{x}_i. \tag{A.9}$$

   This point is the basis of each transformation.

4. **Transformation**

   The simplex is reshaped and updated in one of the following ways. The transformations in a 2-dimensional space are illustrated in Figure A.7.

   a) **Reflection**

   The reflected point $\boldsymbol{x}_r$ is calculated as

   $$\boldsymbol{x}_r = \boldsymbol{c} + \alpha(\boldsymbol{c} - \boldsymbol{x}_1), \tag{A.10}$$

   Then, the point $\boldsymbol{x}_r$ is evaluated. If $f(\boldsymbol{x}_2) < f(\boldsymbol{x}_r) < f(\boldsymbol{x}_3)$, then $\boldsymbol{x}_1$ is replaced by $\boldsymbol{x}_r$ and the simplex is updated. If this is not the case, other transformations are performed.

   b) **Expansion**

   If the reflected point is better than the current best point $f(\boldsymbol{x}_r) > f(\boldsymbol{x}_{n+1})$, the expansion point is computed

   $$\boldsymbol{x}_e = \boldsymbol{c} + \gamma(\boldsymbol{x}_r - \boldsymbol{c}). \tag{A.11}$$

   If $f(\boldsymbol{x}_r) < f(\boldsymbol{x}_e)$, then $\boldsymbol{x}_1$ is replaced by the expanded point $\boldsymbol{x}_e$. Otherwise, if $f(\boldsymbol{x}_e) < f(\boldsymbol{x}_r)$, the reflected point $\boldsymbol{x}_r$ is accepted and the algorithm moves to the next iteration.

   c) **Contraction**

   - **Outside** If $f(\boldsymbol{x}_1) < f(\boldsymbol{x}_r) < f(\boldsymbol{x}_2)$, the simplex is contracted outside

     $$\boldsymbol{x}_{c,(\text{out})} = \boldsymbol{c} + \beta(\boldsymbol{x}_r - \boldsymbol{c}). \tag{A.12}$$

     If the contracted point is better than the reflected $f(\boldsymbol{x}_r) < f(\boldsymbol{x}_{c,(\text{out})})$, $\boldsymbol{x}_{c,(\text{out})}$ is accepted. Otherwise, a shrink step is performed.

   - **Inside** If $f(\boldsymbol{x}_r) < f(\boldsymbol{x}_1)$, the simplex is contracted inside

     $$\boldsymbol{x}_{c,(\text{in})} = \boldsymbol{c} + \beta(\boldsymbol{x}_1 - \boldsymbol{c}). \tag{A.13}$$

     If the contracted point is better than the worst $f(\boldsymbol{x}_1) < f(\boldsymbol{x}_{c,(\text{in})})$, $\boldsymbol{x}_{c,(\text{in})}$ is accepted. Otherwise, a shrink step is performed.

   d) **Shrinkage**

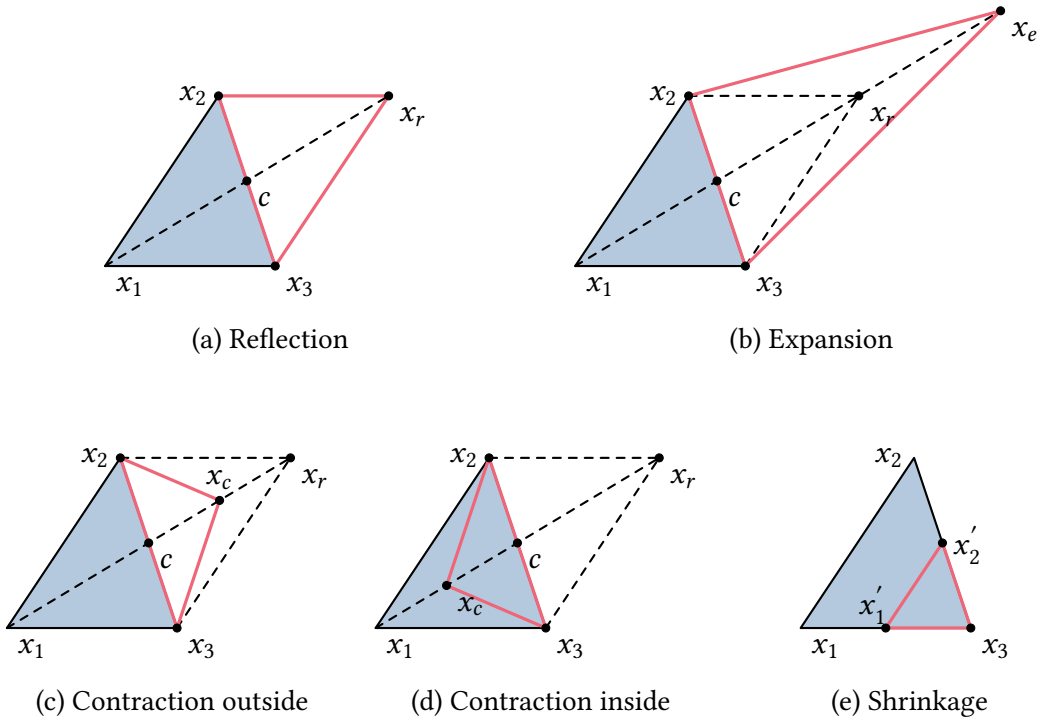
   If none of the above transformations succeed, the shrink transformation is performed. Only the best point $\boldsymbol{x}_{n+1}$ is kept, and all other vertices are changed

   $$\boldsymbol{x}_i = \boldsymbol{x}_{n+1} + \delta(\boldsymbol{x}_i - \boldsymbol{x}_{n+1}), \quad i \in \{1, \ldots, n\}, \tag{A.14}$$

   This is the most time-consuming transformation. Since for n dimensional problem, n new vertices are calculated and the objective function needs to be evaluated n times.

Algorithm 3 Pseudo-code of the Nelder-Mead simplex algorithm

1: Construct the initial simplex S_0
2: **for** $t = 1, 2, \ldots$ **do**
3: Order the vertices of the current simplex S_{t-1}
4: Calculate the centroid c
5: Transform and update the simplex S_t
6: **end for**



(a) Reflection (b) Expansion

(c) Contraction outside (d) Contraction inside (e) Shrinkage

Figure A.7.: Transformations of the Nelder-Mead simplex method. In a 2-dimensional parameter space, the simplex is a triangle. In each panel, the blue-shaded region is the original simplex, and the red lines depict the transformed simplex. The vertices are already ordered, so that $f(x_1) < f(x_2) < f(x_3)$. In all five transformations, the worst point x_1 is replaced by a better point. In the case of shrinkage, only the best point x_3 is kept.

In practice, the Nelder-Mead algorithm mostly takes one or two evaluations for each iteration and converges relatively fast. However, when the dimension of the parameter space gets higher, the shrink transformation scales badly.

Another problem with the Nelder-Mead algorithm is its robustness against noise. When applied to tasks with noisy objective signals, the Nelder-Mead tends to get stuck in local optima and continues to contract/shrink. In that case, the algorithm needs to be restarted to allow further exploration of the parameter space.

A.5.2. Extremum Seeking

Extremum seeking (ES) generally describes the problem of optimizing the performance of a dynamic system by tracking its extremum [202]. In the past decades, ES algorithms have been extensively applied for the online control and optimization of time-varying dynamic systems, such as particle accelerator operations [169, 111, 29].

In Section 6.3, the ES algorithm is used as a baseline to compare the performances of RL and BO algorithms. In the following, this method is shortly introduced based on the description in [203]. It aims to maximize an arbitrary cost function of the form

$$C(\boldsymbol{p}, \boldsymbol{x}, t) = \sum_i h_i(\hat{y}_i(\boldsymbol{p}, \boldsymbol{x}, t), \boldsymbol{p}), \tag{A.15}$$

where $\boldsymbol{x}$ are the observable parameters, such as beam sizes and energies, $\boldsymbol{p}$ are the actuators that can be controlled, and t is the time. The measured signals $\hat{y}_i$ contain noise and h_i are the customizable functions based on the noisy measurements and the actuator settings themselves. For such a dynamic system, the input parameter p_i in step n can be updated by

$$p_i(n+1) = p_i(n) + \Delta_t \sqrt{\alpha_i \omega_i} \cos\left(\omega_i n \Delta_t + k_i C(n)\right), \tag{A.16}$$

where C is the objective value, α_i the amplitude, ω_i the dittering frequency, and Δ_t the timestep for each update. When reaching equilibrium, the parameters will oscillate around fixed points with amplitudes of $s = \sqrt{\alpha/\omega}$.

The evaluation study uses the implementation provided by the Xopt [15] Python package. According to the original paper, the input parameters were normalized to $[-1, 1]$, the dithering frequencies were chosen to be $\omega_i = r_i \omega_0$ with r_i linearly spanned in the range $[1.0, 1.75]$, and $\Delta_t = \frac{2\pi}{10 \max \omega_i}$. BO was used within the WandB framework to perform the hyperparameter tuning across the full set of trails, minimizing the best mean aboslute error (MAE) after 150 steps. The hyperparameter values used for the evaluated ES algorithm are listed in Table A.1. It is worth noting that in the evaluation study in this dissertation, the oscillation amplitude is decayed exponentially by a factor λ to improve the convergence to the target beam parameters. While the original implementation used ES without decay to achieve stable long-term control, the decay rate helps convergence when using ES in an optimization setting. Tuning the oscillation amplitude without a decay rate λ leads to either local optimization or large oscillation at the final steps, resulting in a worse final MAE.

Table A.1.: Hyperparamter values used for the ES algorithm.

Hyperparamter	Value
Feedback gain k	3.7
Oscillation size $s = \sqrt{\alpha/\omega}$	0.11
Decay rate λ	0.987

A.6. FLUTE Reinforcement Learning Training Configurations

The parameters used for training the RL agent for the FLUTE tuning task are shown in Table A.2. They are grouped in environment parameters and RL algorithm parameters. The naming conventions for the parameters are consistent with the ones used in the Stable Baselines3 [185]. If not listed, parameters take their default values as implemented in the Stable Baselines3. The proximal policy optimization (PPO) implementation supports training with vectorized environments, where n_{envs} denotes the number of environments used in parallel. The n_{steps} denotes the number of steps each environment runs before an update. In total, the PPO updates its policy whenever the rollout buffer is filled with $n_{\text{envs}} * n_{\text{steps}}$ samples. The training speed on this simple task can be improved by reducing these numbers and increasing the learning rate.

Table A.2.: Parameters used for the FLUTE reinforcement learning training

	Parameter	Value
Environment	w_{beam}	1.0
	$w_{\text{improvement}}$	0.5
	w_{action}	0.2
	Max episode step	50
RL algorithm	Learning rate	5×10^{-4}
	n_{steps}	128
	n_{envs}	10
	Total timesteps	50 000

A.7. Benchmarking Bayesian Optimization Implementations in the Xopt Package on EA Tuning Task

In this dissertation, a custom version of the BO algorithm is implemented for the benchmarking test at the ARES accelerator in Section 6.3. To ensure that this in-house BO version matches other available variants and implementations of BO, it is compared to the state-of-the-art Xopt package [15], which has been widely adopted in the particle accelerator community.

For benchmarking purposes, two BO versions are evaluated from the Xopt backend, one using a hard step-size constraint and the other one using the proximal-biasing [75] technique as a soft step-size constraint. Both versions use upper confidence bound (UCB) acquisition with $\beta = 2$ and perform the default pre-processing steps, i.e. normalizing the input to $[0, 1]$ and standardizing the objective values. The hard version used the same step-size limit of 0.1 (in the normalized space) as the BO and RL versions used in this study. The proximal weight was set to be 0.5, which means the acquisition drops to $1/e$ over 10 % of the action space. This was obtained from a hyperparameter tuning, optimizing for the best MAE within 150 steps. Note that this is higher than the original value 0.1 proposed in [75], which was used for a much smoother objective landscape. In the studied ARES task, smaller proximal weights would mostly lead to premature convergence, due to a large number of local optima.

The comparison results are shown in the supplementary materials in [95]. While in simulation, the custom BO implementation falls right in between the two provided by Xopt, on the real ARES accelerator, the custom implementation consistently performs the best. It proves that the BO implementation used in this dissertation represents the state of the art in BO for particle accelerator tuning. It is worth noting that the proximal BO produced smoother action than the other versions, as also observed in the application in the EuXFEL shown in Section 6.1. This smooth convergence could be favorable in other situations where smooth actions are more critical, for example, to reduce the mechanical stress of actuators or tuning systems with a non-negligible hysteresis effect.

A.8. Lattice-agnostic Reinforcement Learning Training Configurations

Table A.3 lists the hyperparameters used for training the lattice-agnostic RL agent in Section 6.4.1. The naming conventions are consistent with the ones used in the Stable Baselines3 package [185]. The values are empirically tuned with the help of the WandB package [191]. The hyperparameters not listed in the table take their default values. Soft actor-critic (SAC) is used for the RL training. Parameters are tuned using the experiment tracking package Weights and Biases.

Table A.3.: Parameters used for lattice-agnostic reinforcement learning training

Parameter	Value
Discount factor γ	0.99
Learning rate α	0.0003
Replay buffer size	10 000
Batch size	100
Gradient steps	1
Total timesteps	500 000
Max episode length	50

A.9. Code Availability

- The code showcasing the Cheetah applications is available at the Cheetah repository `https://github.com/desy-ml/cheetah` and the accompanying repository with the demonstrations `https://github.com/desy-ml/cheetah-demos`.
- The code for the BO beamtime at the European X-Ray Free-Electron Laser (EuXFEL) is available at the GitHub repository `https://github.com/cr-xu/bo-4-euxfel`.
- The code for benchmarking the BO and RL at the ARES accelerator is available at `https://github.com/desy-ml/rl-vs-bo`.
- The code for the meta-RL study is available in the tutorial repository [196].
- The rest of the code is available in the KIT Gitlab instance `https://gitlab.kit.edu/kit/ibpt/ai4accelerators/`, which can be accessed upon reasonable request.

Acknowledgments

The successful completion of this thesis would not have been possible without the support of many individuals. I would like to take the opportunity to extend my heartfelt gratitude to everyone who contributed to this journey.

First and foremost, I would like to thank Prof. Anke-Susanne Müller for providing the opportunity to conduct my doctoral study at IBPT, and allowing me to freely pursue all these projects that interest me.

I want to thank Prof. Torben Ferber for kindly taking over the task of being the second reviewer of this thesis on a cross-disciplinary topic.

I would also like to thank Erik Bründermann who not only advised me scientifically, but also provided guidance on administrative perspectives and pursuing a scientific career in general.

I want to thank Andrea Santamaria Garcia for being my scientific advisor, providing invaluable insights on everything from science to life itself. I cherished all the discussions, meetings that we sat through together, and, of course, many memorable road trips that we shared. Thank you for the countless hours spent proofreading my manuscripts and for helping me (hopefully now) embrace better writing habits. Your aesthetic critiques of my plots and slides were indeed spot-on as well. Without your guidance, this dissertation would be surely less enjoyable to read. I am deeply grateful for your mentorship. None of this would be possible without your support!

I also acknowledge the support of the Doctoral School "Karlsruhe School of Elementary Particle and Astroparticle Physics: Science and Technology" (KSETA).

I feel extremely fortunate to have Jan Kaiser along the path, both as a fellow doctoral and a dear friend. We started around the same time and worked in collaboration for a majority part of the projects. As a traditional saying goes, *"By using people as a mirror, one can see one's gains and losses."* I am continually in awe of his passion for the field, boundless energy, and incredible productivity. Being able to chat about all the little details and brainstorming bizarre ideas with him keep me motivated and excited about this subject, which remains to be relatively niche. My coding skills have also significantly improved, thanks to the watchful eyes of a computer scientist.

I would also like to thank Luca Scomparin for the fun memories, and opening up the world of embedded system and low-level programming for me. Combining RL methods with edge computing for real-time accelerator control is arguably the coolest project that I have worked on.

In the process of conducting this thesis, I also received assistance from colleagues in other groups. I would like to thank the colleagues from DESY, especially Prof. Annika Eichler, Florian Burkart, and Oliver Stein, for the fruitful collaborations. I would also like to thank Sergey Tomin for insightful discussions and the possibility to conduct experiments

at the European XFEL. I want to also thank Simon Hirlaender at Salzburg for his insights and guidance on reinforcement learning topics.

I want to thank all the present and former colleagues in the THz group at IBPT for the friendly and enjoyable working environment. In particular, I want to thank everyone who has contributed to this thesis in various ways and helped proofread the content. A special thank you to Tobias Boltz. He presented to me the possibility of combining AI/ML methods with accelerator physics and convinced me to conduct the master thesis on this topic. I would not have treaded this path without his encouragement and passion in this field. I want to thank Edmund Blomley, Julian Gethmann, and Johanees Steinmann who taught me a lot about software, hardware, and control systems. I also want to thank Markus Schwarz and Michael Nasse for the technical discussions and support.

I want to thank my fellow doctoral students Jens Schäfer, Sebastian Maier, Micha Reißig, Matthias Nabinger, Marvin Noll, and Felipe Donoso Aguirre for all the lengthy discussions and cheerful moments we shared during the coffee breaks. In addition, I want to thank Matthias also for his expertise in laser and for his patience while working with a lab newbie like me. The SLM experiments at FLUTE were made possible thanks to his support. A special thank you also to Sebastian Maier for being my office mate over four years and enduring my increasing number of online meetings. Thanks to his punctuality and self-discipline, I somehow managed to never miss a registration or submission deadline.

Finally, I want to express my gratitude to my dear family for their unwavering faith in me. To my parents, thank you for always giving me the freedom and supporting my rather unconventional life choices, from "I want to major in physics... in Germany", to "I feel like doing a PhD in physics", to "I want to continue on research". To my sister, thank you for being always so cheerful and loving. And lastly, to Zhenlin, my dear partner, thank you for being there with me from the beginning. Thank you for putting up with me through all the up and downs. We have taken the first step in faith, and together, we will continue forward through whatever comes next.