



Machine learning-based particle accelerator modeling

Emmanuel Goutierre

► To cite this version:

Emmanuel Goutierre. Machine learning-based particle accelerator modeling. Artificial Intelligence [cs.AI]. Université Paris-Saclay, 2024. English. ⟨NNT : 2024UPASG106⟩. ⟨tel-04995631⟩

HAL Id: tel-04995631

<https://theses.hal.science/tel-04995631v1>

Submitted on 18 Mar 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

Machine Learning-Based Particle Accelerator Modeling

*Utilisation de Méthodes d'Apprentissage pour la
Modélisation d'un Accélérateur de Particules*

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580, Sciences et Technologies de l'Information et de la
Communication (STIC)

Spécialité de doctorat : Informatique

Graduate School : Informatique et Sciences du Numérique

Référent : Faculté des Sciences d'Orsay

Thèse préparée dans les unités de recherche **IJCLab (Université Paris-Saclay, CNRS) et
LISN (Université Paris-Saclay, CNRS),**

sous la direction de **Johanne COHEN**, Directrice de recherche,
le co-encadrement de **Christelle BRUNI**, Chargée de recherche
Hayg GULER, Ingénieur de recherche

Thèse soutenue à Paris-Saclay, le 19 décembre 2024, par

Emmanuel GOUTIERRE

Composition du jury

Membres du jury avec voix délibérative

Fatiha SAÏS

Professeure, LISN, Université Paris-Saclay

Annika EICHLER

Professeure, DESY, Technische Universität Hamburg (TUHH)

Patrick GALLINARI

Professeur, ISIR, Sorbonne Université

Nicolas BASKIOTIS

Maître de conférences, ISIR, Sorbonne Université

Adnan GHRIBI

Ingénieur de recherche, Grand Accélérateur National d'Ions
Lourds (GANIL)

Présidente

Rapporteure & Examinatrice

Rapporteur

Examineur

Examineur

Titre : Utilisation de Méthodes d'Apprentissage pour la Modélisation d'un Accélérateur de Particules

Mots clés : Apprentissage Automatique, Simulation d'Accélérateur de Particules, Modélisation par Modèle de Substitution, Modélisation de Nuage de Particules, Modélisation de Systèmes Dynamiques,

Résumé : Les accélérateurs de particules reposent sur des simulations de haute précision pour optimiser la dynamique du faisceau. Ces simulations sont coûteuses en ressources de calcul, rendant leur analyse en temps réel difficilement réalisable. Cette thèse propose de surmonter cette limitation en explorant le potentiel de l'apprentissage automatique pour développer des modèles de substitution des simulations d'accélérateurs de particules.

Ce travail se concentre sur ThomX, une source Compton compacte, et introduit deux modèles de substitution : LINACNET et *Implicit Neural ODE* (INODE). Ces modèles sont entraînés sur une base de données développée dans le cadre de cette thèse, couvrant une grande variété de conditions opérationnelles afin d'assurer leur robustesse et leur capacité de généralisation.

LINACNET offre une représentation complète du nuage de particules en prédisant les coordonnées de toutes les macro-particules du faisceau

plutôt que de se limiter à ses observables. Cette modélisation détaillée, couplée à une approche séquentielle prenant en compte la dynamique cumulative des particules tout au long de l'accélérateur, garantit la cohérence des prédictions et améliore l'interprétabilité du modèle.

INODE, basé sur le cadre des *Neural Ordinary Differential Equations* (NODE), vise à apprendre les dynamiques implicites régissant les systèmes de particules sans avoir à résoudre explicitement les équations différentielles pendant l'entraînement. Contrairement aux méthodes basées sur NODE, qui peinent à gérer les discontinuités, INODE est conçu théoriquement pour les traiter plus efficacement.

Ensemble, LINACNET et INODE servent de modèles de substitution pour ThomX, démontrant leur capacité à approximer la dynamique des particules. Ce travail pose les bases pour développer et améliorer la fiabilité des modèles basés sur l'apprentissage automatique en physique des accélérateurs.

Title: Machine Learning-Based Particle Accelerator Modeling

Keywords: Machine Learning, Particle Accelerator Simulation, Surrogate Modeling, Particle Cloud Modeling, Dynamical System Modeling,

Abstract: Particle accelerators rely on high-precision simulations to optimize beam dynamics. These simulations are computationally expensive, making real-time analysis impractical. This thesis seeks to address this limitation by exploring the potential of machine learning to develop surrogate models for particle accelerator simulations.

The focus is on ThomX, a compact Compton source, where two surrogate models are introduced: LINACNET and Implicit Neural ODE (INODE). These models are trained on a comprehensive database developed in this thesis that captures a wide range of operating conditions to ensure robustness and generalizability.

LINACNET provides a comprehensive representation of the particle cloud by predicting all coordinates of the macro-particles, rather than focusing solely on beam observables. This de-

tailed modeling, coupled with a sequential approach that accounts for cumulative particle dynamics throughout the accelerator, ensures consistency and enhances model interpretability.

INODE, based on the Neural Ordinary Differential Equation (NODE) framework, seeks to learn the implicit governing dynamics of particle systems without the need for explicit ODE solving during training. Unlike traditional NODEs, which struggle with discontinuities, INODE is theoretically designed to handle them more effectively.

Together, LINACNET and INODE serve as surrogate models for ThomX, demonstrating their ability to approximate particle dynamics. This work lays the groundwork for developing and improving the reliability of machine learning-based models in accelerator physics.

Synthèse en Français

Les accélérateurs de particules nécessitent des simulations de haute précision pour optimiser la dynamique du faisceau. Cependant, ces simulations sont coûteuses en ressources de calcul, rendant leur utilisation en temps réel difficilement envisageable. Cette thèse explore l'utilisation de l'apprentissage automatique pour développer des modèles de substitution capables d'approximer fidèlement ces simulations tout en réduisant considérablement leur coût computationnel.

L'étude porte sur ThomX, une source Compton compacte, et introduit deux modèles de substitution : LINACNET et INODE (Implicit Neural ODE). Ces modèles sont entraînés sur une base de données simulée couvrant un large éventail de conditions opérationnelles afin d'assurer leur robustesse et leur capacité de généralisation.

LINACNET repose sur une approche novatrice de modélisation séquentielle du nuage de particules. Contrairement aux approches de modélisation classiques qui ne prédisent que les observables globales du faisceau, LINACNET capture la dynamique détaillée des particules en prédisant l'évolution de l'ensemble du nuage de particules à chaque étape de l'accélérateur linéaire de ThomX. Cette approche permet de mieux prendre en compte les effets cumulés des interactions entre les particules et des variations du faisceau tout au long de son trajet dans l'accélérateur. L'intégration d'une représentation complète du nuage de particules améliore non seulement la précision des prédictions, mais aussi l'interprétabilité physique des résultats, offrant ainsi un outil plus adapté aux besoins des physiciens des accélérateurs.

INODE, quant à lui, s'appuie sur le cadre théorique des Neural Ordinary Differential Equations (NODE). Il cherche à apprendre les dynamiques implicites des systèmes de particules en modélisant directement l'équation différentielle sous-jacente qui régit leur évolution. Contrairement aux méthodes classiques basées sur NODE, qui peinent à gérer les discontinuités, INODE introduit une approche permettant de mieux traiter ces phénomènes tout en évitant de résoudre explicitement les équations différentielles durant l'apprentissage. Cela lui confère une plus grande flexibilité et adaptabilité à des systèmes dynamiques complexes, rendant cette approche particulièrement prometteuse pour la physique des accélérateurs.

La thèse est organisée comme suit :

- Le chapitre 0 introduit le contexte général et les motivations de cette étude. Il présente les enjeux liés à la simulation des accélérateurs de particules, les défis computationnels qu'elle pose, ainsi que l'intérêt des modèles de substitution basés sur l'apprentissage automatique. Les principales contributions de cette thèse y sont également résumées.
- Le chapitre 1 présente les principes fondamentaux de la physique des accélérateurs, en mettant l'accent sur les mécanismes d'accélération des particules et la caractérisation des faisceaux. Il introduit également les simulateurs d'accélérateurs de particules et les défis liés à leur utilisation.
- Le chapitre 2 dresse un état de l'art des modèles de substitution appliqués aux accélérateurs de particules. Il examine les approches existantes et met en évidence leurs limites, justifiant ainsi la nécessité d'explorer de nouvelles méthodes basées sur l'apprentissage automatique.
- Le chapitre 3 décrit la construction de la base de données utilisée pour entraîner LINACNET et INODE. Il détaille la méthodologie de génération des données, les paramètres de simulation et les critères de validation adoptés.
- Le chapitre 4 est consacré à la conception et à l'évaluation de LINACNET. Il explore les différentes stratégies d'apprentissage mises en œuvre pour modéliser le nuage de particules et analyse la performance du modèle en le comparant aux simulations traditionnelles.
- Le chapitre 5 introduit INODE et en développe le cadre théorique. Il présente les fondements mathématiques de l'approche, discute de ses avantages par rapport aux NODE classiques et propose une validation expérimentale du modèle.
- Le chapitre 6 conclut cette thèse en synthétisant les contributions principales et en ouvrant des perspectives pour l'amélioration et l'extension des modèles développés.

Contents

0	Introduction	5
0.1	General Motivation	5
0.2	Challenges	6
0.3	Context of This Work	7
0.4	Contributions	7
0.5	Outline	8
1	Problem Statement	11
1.1	General Principles of Accelerator Physics	11
1.1.1	Particle Accelerators	12
1.1.2	Particle Accelerator Components	14
1.1.3	Beam Characterization	20
1.1.4	Diagnostics	23
1.2	Particle Accelerator Simulators	25
2	State of the Art for Surrogate Modeling: The case of Particle Accelerators	29
2.1	Surrogate Modeling: The Context of Particle Accelerators	30
2.2	Data Acquisition for Particle Accelerator Simulators Modeling	31
2.3	Artificial Intelligence in Particle Accelerator Modeling	32
2.3.1	Offline Optimization of Accelerator Operations	33
2.3.2	Online Control of Accelerator Operations	33
2.3.3	Fault Detection and Diagnostics	34
2.4	Data Structure in Particle Accelerator Simulations	34
2.5	The Multi-Objective Nature of Particle Accelerator Simulator Modeling	36
2.5.1	Scalarization Methods	37
2.5.2	Gradient-Based Adaptive Methods	38
3	Towards Particle Accelerator Simulator Modeling: Database and Supervised Learning Framework	41
3.1	Database Generation	41
3.1.1	Simulation Overview	42
3.1.2	Macro-Particle Tradeoff	44
3.1.3	Control Parameter Sampling	45
3.1.4	Simulation Results Overview	48
3.2	Performance Indicators	50
3.3	Conclusion	60

4	Particle Cloud Modeling: Explicit Sequential Supervised Learning	61
4.1	Particle Cloud Modeling	62
4.1.1	Modeling a Particle Cloud	62
4.1.2	Computing a Loss for a Particle Cloud Prediction	66
4.2	Beam Propagation Modeling in a Sequence of Accelerator Segments	68
4.2.1	Modeling	68
4.2.2	A Multi-Objective Problem	69
4.2.3	A Multi-Task Learning Solution	70
4.3	Validation	74
4.3.1	Goals of Experiments	74
4.3.2	First Segment Modeling	75
4.3.3	Multi-segment modeling	85
5	Dynamical System Modeling: an Implicit ODE Approach	91
5.1	Machine Learning for Dynamical System Modeling	92
5.2	INODE Method	93
5.2.1	From NODE to INODE : Learning an ODE in an Implicit Space	93
5.2.2	Adjusting for Control Parameters	94
5.2.3	Evaluation Process with INODE	95
5.2.4	Summary of the INODE Method	95
5.3	INODE Theoretical Foundation	96
5.3.1	Implicit Trajectories	98
5.3.2	Universal Approximation Theorem	100
5.4	INODE Experiments	103
5.5	Conclusion and Future Works	106
6	Conclusion	109
6.1	Context and Challenges of Particle Accelerator Simulation	110
6.2	Contributions	110
6.2.1	LINACNET	110
6.2.2	INODE	111
6.3	Limitations and Future Work	111
6.3.1	Applicability of the Surrogate Models	111
6.3.2	Particle Cloud Representation	111
6.3.3	Computation Intensive Device Requirements	112
6.3.4	Looking for more adapted operators for INODE	112
6.3.5	Experimental Validation of INODE	112
6.3.6	Merging INODE and LINACNET	112
A	Sampling Strategies in Surrogate Modeling	113
A.1	Random Sampling	113
A.2	Quasi-Monte Carlo Methods	114
A.3	Active Learning	114

B INode Mathematical Formalism 117

B.1 Background 117

B.2 Technical proofs 122

 B.2.1 Existence and Uniqueness Theorem 122

 B.2.2 Universal Approximation Theorem 124

 B.2.3 Folk Knowledge 127

Bibliography 129

0 -Introduction

Contents

0.1 General Motivation	5
0.2 Challenges	6
0.3 Context of This Work	7
0.4 Contributions	7
0.5 Outline	8

0.1 .General Motivation

Particle accelerators are devices that use electromagnetic fields to increase the energy of charged particles, accelerating them to fractions of the speed of light. These particles form beams whose properties must be carefully tuned for the desired application. These applications include the study of the fundamental properties of matter, materials science, spectrometry, radiation therapy, and food sterilization [57].

Particle accelerators are among the most complex and expensive machines built by humans. Numerous and diverse components are necessary for an accelerator's proper functioning (accelerating cavity, magnet, radio-frequency system, vacuum chamber, etc., each of which must be adjusted with the utmost precision [73]. With the increasing complexity of particle accelerator systems, modeling and simulation tools have been developed to assist physicists in making decisions [22, 45, 25, 50]. These simulation codes play a role at several stages of an accelerator development. Initially, simulations are used during the design phase to validate and compare accelerator plans. Then, during the optimization phase, simulations are used to find the optimal theoretical settings to produce the best possible beam according to the criteria set for the intended application [55]. Finally, during the machine's operation phase, simulations are used as a reference to compare with experimental results.

However, these simulations have several limitations, the most important of which is computational time. Indeed, the more numerical precision required for the beam simulation, the more physical phenomena are taken into account, and the longer the simulation time becomes, which can extend to several days on dedicated computing centers for the most prolonged simulations [24, 77]. The real-time simulation used during an accelerator's operations is reserved

for rare cases where the simulations can be simplified or when large high-performance computing resources are available in real-time.

Another significant limitation of particle accelerator simulators is the reality gap, which refers to the discrepancy between simulated models and real-world performance. While simulations are designed to reproduce the physical phenomena and operating conditions of particle accelerators as accurately as possible, they are inherently simplified and idealized. These simplifications can lead to discrepancies when the simulated predictions are compared with experimental results. Factors such as uncertainties in the input parameter algorithms contribute to this gap [8].

One of the areas for improvement of simulators lies in the fact that each change in the accelerator's configuration requires a new complete simulation without being able to take advantage of previous simulations already performed. Machine learning methods offer a way to use the entire history of simulations to establish a surrogate model of the simulator [20]. By learning to reproduce the behavior of the simulator linking the control parameters of the simulation to the observables predicted by the simulator, these surrogate models offer a potential solution to extrapolate the results of a simulation for new control parameters very quickly without having to bear the burden of running a complete simulation again.

0.2 .Challenges

A surrogate model combines data from the phenomenon being modeled and a learning algorithm, both of which are essential to the quality of the model. While many generic learning algorithms exist, the representation of the data provided plays a crucial role in the quality of the final model. Thus, a thorough understanding of the workings of a particle accelerator simulator is necessary to construct an adequate representation.

Particles in interaction. A particle beam is a collection of electrically charged particles that interact with each other and propagate through an accelerator [73]. These particles interact with the electromagnetic environment created by the accelerator components. They also interact with each other: each particle exerts an electromagnetic force on the others due to its charge [53].

Representing all inter-particle interactions within a beam could lead to a highly complex model that does not meet the expectations of deployability, real-time execution, or learning feasibility. Conversely, a model that ignores these interactions might oversimplify the problem. Therefore, it will be appropriate to study how to represent these interactions in a surrogate model and to verify whether the representation of these interactions is crucial to the

quality of such a model.

Propagating Beam The set of particles propagates through the various constituent elements of an accelerator. Each element has a specific role and effect on the beam, which has different properties depending on its position in the accelerator.

The surrogate model must be able to reproduce all these different phenomena. To achieve this, it will also be necessary to consider whether and how to represent this sequence of elements in the model and to determine the appropriate level of detail.

0.3 .Context of This Work

This work is part of developing a particle accelerator named ThomX [10], being developed in Orsay, France. ThomX aims to utilize inverse Compton scattering to produce a beam of hard X-rays. It is a compact machine, occupying an area of $70m^2$, and consists of a linear accelerator, a transfer line and a storage ring that allows the interaction of the electron beam with a laser to generate X-rays. The accelerator began operation in 2021 and is currently in the commissioning phase. The first component to be commissioned is the ThomX's linear accelerator (LINAC), which will be used to illustrate all the discussions in this thesis.

Nonetheless, this work is part of a much broader perspective. The perspective is to develop general methods and establish the basic components for implementing surrogate models for a wide range of particle accelerators. These include not only ThomX but also next-generation accelerator projects such as the PERLE Energy Recovery Linac (ERL) [34].

Moreover, the development of these models must be accompanied by tools to ensure confidence in the proposed models. Thus, a surrogate model must not only provide predictions of the accelerator's observables but also include the information necessary for physicists to evaluate their relevance and consistency. This aligns with the broader context of trustworthy AI, which emphasizes the reliability and transparency of AI-driven solutions.

0.4 .Contributions

This thesis aims to provide the essential components for the modeling of particle accelerator simulators. The study of the various challenges mentioned in Section 0.2 leads us to propose a surrogate model called LINACNET . This model successfully reproduces the behavior of the Astra [22] simulator across a wide range of values representing the potential configurations of the ThomX Linac. LINACNET provides the complete distribution of particles at different

points in the accelerator along with the physical output values of the simulator. These distributions allow for assessing the predicted scalar values' relevance and evaluating the predictions' credibility along the accelerator.

While LINACNET aims to learn the full set of particles at different points along the accelerator, we also explore an alternative approach that focuses on learning the governing Ordinary Differential Equation (ODE) derived from a set of control parameters. This approach leads to a second model, called Implicit Neural ODE (INODE), which is inspired by Neural Ordinary Differential Equations [12] (NODE). Implicit Neural ODE improves over NODE by learning an underlying ODE without the need to use ODE solver during training.

0.5 .Outline

This manuscript is organized as follows.

Chapter 1 provides a comprehensive introduction to the fundamentals of accelerator physics, including particle acceleration mechanisms and beam dynamics. It also details the operating principles of particle accelerator simulators, laying the foundation for the development of a surrogate model.

Chapter 2 reviews the current state of the art in surrogate models for particle accelerator simulators. It examines several existing models, comparing their methodologies, strengths, and limitations. In addition, it analyzes the learning tools and techniques that facilitate the creation of models inspired by simulator operating mechanisms. This review helps identify gaps in current research and areas where LINACNET and INODE can contribute novel solutions.

Chapter 3 outlines the experimental setup used to validate LINACNET and INODE. It provides a detailed description of the simulation environment, including the configurations and parameters of the ThomX Linac, which were used to generate a comprehensive database that emulates various operational scenarios of ThomX. In addition, the chapter explains the evaluation criteria used to assess the performance of the model, ensuring that the surrogate model accurately replicates the behavior of the simulator over a wide range of conditions.

Chapter 4 describes the general framework for developing a surrogate model for particle accelerator simulators. It introduces LINACNET, explains its conceptual basis inspired from point cloud modeling [11] and multi-task learning [63] and the different solutions considered during its development. The chapter examines the algorithms and machine learning techniques used, highlighting how they mimic the behavior of traditional simulators. It provides a detailed analysis of the model's performance, comparing its predictions with those of the Astra simulator. The chapter highlights LINACNET's accuracy in predicting particle distributions and physical output values, demonstrating its relevance and reliability as a surrogate model.

Chapter 5 introduces the Implicit Neural ODE (INODE) method as a tool for modeling dynamic systems. It provides a detailed mathematical foundation, highlighting how INODE can bypass the need for an ODE solver during the training process while still effectively learning the governing ODE of the dynamic system.

Finally, Chapter 6 offers perspectives on the potential applications of LINACNET and INODE in particle accelerator research and development. It discusses how the model can be integrated into existing accelerator projects and its potential to support future innovations. The chapter also identifies areas for further research and potential improvements to enhance the performance and applicability of LINACNET and INODE .

1 -Problem Statement

Contents

1.1 General Principles of Accelerator Physics	11
1.1.1 Particle Accelerators	12
1.1.2 Particle Accelerator Components	14
1.1.3 Beam Characterization	20
1.1.4 Diagnostics	23
1.2 Particle Accelerator Simulators	25

Initially, it is essential to identify the operating principles of the phenomenon being modeled to highlight its specific characteristics that can be used to construct a surrogate model. Therefore, Section 1.1 will provide some general principles of accelerator physics, and Section 1.2 will discuss the principles of tools used for simulating particle accelerators. These sections will use the ThomX accelerator and its simulation as examples. The aim of these sections is not to provide an exhaustive description of the principles of particle accelerators and their simulation but to highlight key elements. Reference works will be mentioned for interested readers who wish to delve deeper into the topics discussed.

1.1 .General Principles of Accelerator Physics

This section aims to provide the reader with the general operating principles of a particle accelerator, using ThomX as a case study.

Section 1.1.1 briefly describes the different types and uses of particle accelerators. Section 1.1.2 describes the various components of a particle accelerator, with particular attention to the elements present in the ThomX linac. It details components' specific roles and functionalities, such as the linear accelerator (Linac), transfer line, and storage ring. Section 1.1.3 provides the elements needed to characterize a beam within an accelerator. It discusses the key parameters and metrics used to evaluate beam quality, stability, and performance, essential for optimizing accelerator operation and achieving desired outcomes. Finally, Section 1.1.4 details the diagnostic elements installed in the ThomX Linac. This section covers the various instruments and techniques used to monitor and assess the accelerator's performance, ensuring accurate and reliable operation.

Interested readers are referred to Wiedemann [73] for a more in-depth overview of particle accelerator mechanisms.

1.1.1 .Particle Accelerators

1.1.1.1 .General Overview

A *particle accelerator* is a system that uses electromagnetic fields to interact with a set of charged particles. It consists of two main elements. The first is the *particle source*, which generates charged particles. These particles can be electrons, protons, or heavier ions, depending on the intended experiment or application. The extraction of particles from matter can be achieved through several methods, including gas ionization, thermionic emission from heated filaments, or laser-induced photoelectric effect. Once produced, these particles are often formatted into bunches and directed to the second key element of the accelerator, the *acceleration system*, which accelerates them to high speeds using electromagnetic fields.

Modern acceleration systems generally fall into two categories: *linear accelerators* and *circular accelerators*. In linear accelerators, the most commonly used system is placing a succession of cavities, generating an accelerating field that progressively enables particle bunches to gain energy. These systems produce high-intensity, low-dispersion beams. However, each bunch only passes through each accelerating cavity once. Building long (and expensive) accelerators is necessary to achieve high-energy beams.

Circular accelerators, such as synchrotrons and cyclotrons, use powerful magnetic fields to keep the particles on a circular trajectory while accelerating at each pass. This method allows for achieving very high energy levels without requiring the considerable length of linear accelerators. The accelerated particles circulate in a storage ring where they can be used for various experiments, such as colliding particles to study the fundamental properties of matter or producing synchrotron radiation for materials research. However, circular accelerators are limited by energy loss due to synchrotron radiation, which restricts their efficiency at ultra-high energies.

1.1.1.2 .Fundamental Laws

Various components are responsible for creating the electromagnetic field within a particle accelerator. Particle accelerators usually maintain a high vacuum to ensure the proper propagation of particles; the resulting field within the accelerator can be seen as the solution to Maxwell's equations in a vacuum, with boundary conditions given by the characteristics of the various components. Maxwell's equations in a vacuum are divided into four laws: Maxwell-Gauss equation (1.1), Maxwell-Thomson equation (1.2), Maxwell-Faraday equation (1.3) and Maxwell-Ampère equation (1.4).

$$\vec{\nabla} \cdot \vec{E}(x, y, z, t) = \frac{\rho(x, y, z, t)}{\epsilon_0} \quad (1.1)$$

$$\vec{\nabla} \cdot \vec{B}(x, y, z, t) = 0 \quad (1.2)$$

$$\vec{\nabla} \wedge \vec{E}(x, y, z, t) = -\frac{\partial \vec{B}}{\partial t}(x, y, z, t) \quad (1.3)$$

$$\vec{\nabla} \wedge \vec{B}(x, y, z, t) = \mu_0 \vec{j}(x, y, z, t) + \mu_0 \epsilon_0 \frac{\partial \vec{E}}{\partial t}(x, y, z, t) \quad (1.4)$$

where $\vec{E}(x, y, z, t)$ is the electric field, $\vec{B}(x, y, z, t)$ is the magnetic field, $\rho(x, y, z, t)$ is the volume charge density, ϵ_0 is the vacuum permittivity constant, μ_0 is the vacuum magnetic permeability and $\vec{j}(x, y, z, t)$ is the current density vector.

A charged particle q moving in such an electromagnetic field is subject to the Lorentz force (see eq. 1.5).

$$\vec{F}(t) = q \left(\vec{E}(x(t), y(t), z(t), t) + \vec{v}(t) \wedge \vec{B}(x(t), y(t), z(t), t) \right) \quad (1.5)$$

where $x(t), y(t), z(t)$ are the particle coordinates of the particle and $\vec{v}(t) = \left(\frac{dx}{dt}(t), \frac{dy}{dt}(t), \frac{dz}{dt}(t) \right)$ is its speed.

The particle's trajectory can be determined using Newton's law of motion (see Eq. 1.6).

$$\vec{F}(t) = m\vec{a}(t) \quad (1.6)$$

where $\vec{a}(t) = \frac{d\vec{v}}{dt}(t)$ is the particle's acceleration.

1.1.1.3 .Particle Accelerator Applications

Beyond fundamental research in particle physics, particle accelerators have applications in various fields, ranging from medicine to industry, archaeology and materials science [57].

In medicine, particle accelerators play a role in diagnosing and treating diseases. For example, radiography and positron emission tomography (PET) produce radioactive isotopes used as tracers. In cancer treatment, particle beam therapy, such as proton therapy, uses accelerated particle beams to target and destroy tumors with high precision.

In industry, particle accelerators have varied applications, including material processing, sterilizing medical tools and food items, and cross-linking

plastics to improve their physical properties. They are also used for the radiation inspection of welded structures, foundry parts, and aircraft components to detect internal defects without causing damage.

In archaeology and materials science, particle accelerators enable non-destructive analyses of samples to determine their chemical composition and structure. Carbon-14 dating, for instance, relies on accelerators to measure the remaining carbon-14 in organic samples, determining their age. In materials research, accelerators probe the internal structure of materials, study defects and interfaces, and understand materials' mechanical, electronic, and magnetic properties at the nanometer scale. This understanding is crucial for developing new materials with enhanced properties.

Finally, particle accelerators also contribute to environmental applications, such as analyzing air, water, and soil pollutants and monitoring and controlling industrial pollution sources.

1.1.1.4 .ThomX

ThomX [71] is a prototype of a compact Compton source (CCS). This accelerator aims to create hard X-rays (with energy between 45 and 90 keV) of high brilliance (number of photons emitted per second, per unit area of the source, per unit solid angle, and within an energy bandwidth of 0.1%) using Compton backscattering [10]. The principle of these facilities is to transfer a fraction of the energy from electrons to photons through the interaction between an electron beam created by the accelerator and a laser beam. These facilities provide higher quality X-rays than conventional X-ray tubes while being more compact than facilities using synchrotron radiation from circular accelerators.

ThomX can be schematically divided into four parts (see Fig 1.1).

- The *linear accelerator* (Linac) consists of the photo-injector where electrons are created and accelerated up to 5 MeV, and the accelerating section, which accelerates the electrons up to an energy of 50 MeV.
- The *transfer line* consists of magnets that deflect and transport the beam to bring it to the injection point.
- The *storage ring* is where the electron are stored and where the beam interacts with a laser beam at the interaction point to generate X-rays through the Fabry-Perot cavity.
- The *extraction line* allows for the disposal of the used electron beam and helps to diagnose the beam after interaction with the laser beam.

1.1.2 .Particle Accelerator Components

In this section, we present the various components of the ThomX Linac.

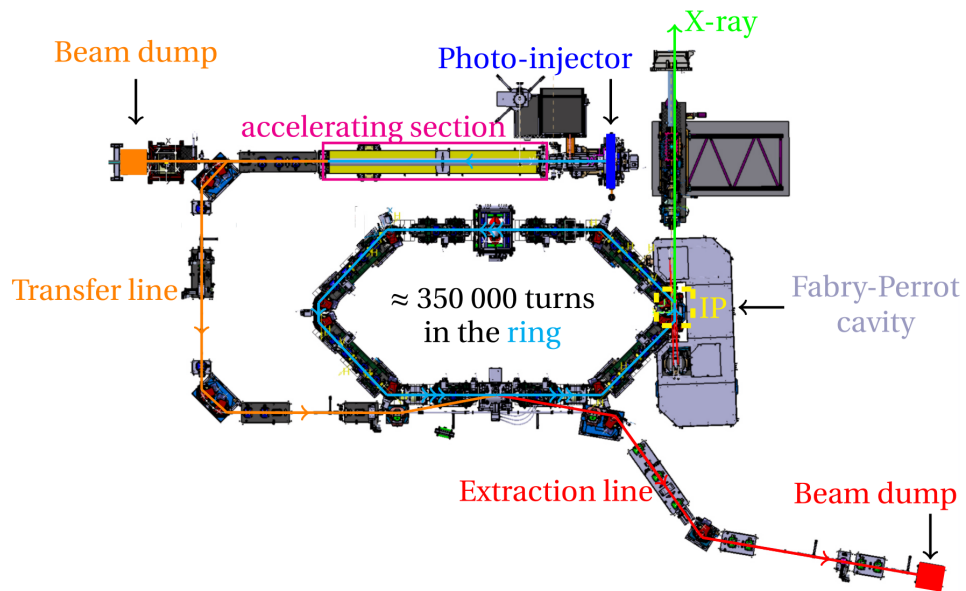


Figure 1.1: Drawing of ThomX seen from above

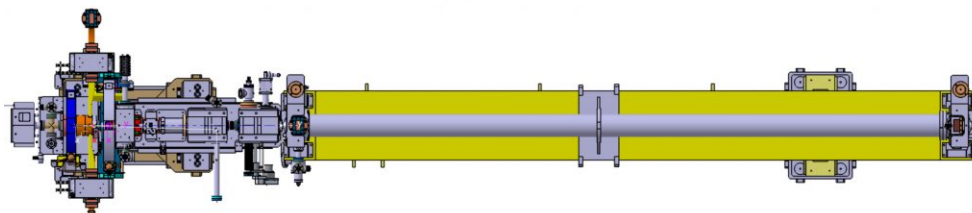


Figure 1.2: ThomX Linac. The first part (grey) is the electron gun, and the second (yellow) is the accelerating section.

Table 1.1: Characteristics of the ThomX Photocathode Laser

Wavelength	257nm.
Energy max	100 μ J.
Pulse duration	5ps.
Trigger Frequency	50Hz.

1.1.2.1 .Photocathode

In ThomX, the particle source is a photocathode. A *photocathode* is a specialized material with a work function that allows it to emit electrons when illuminated by light. This phenomenon is known as the *photoelectric effect*. The photoelectric effect occurs when a material—in this case, the photocathode—absorbs photons (the elementary particles of light) that have sufficient energy to release electrons from the material's surface. This required energy, specific to each material, is known as the work function (Φ). Mathematically, the energy of a photon (E_{photon}) is given by $E_{\text{photon}} = h\nu$, where h is Planck's constant and ν is the frequency of the incident light.

When E_{photon} is greater than or equal to Φ , electrons absorb the energy of the photons and are ejected from the material's surface. The kinetic energy E_{kin} of the ejected electrons is given by Einstein's equation for the photoelectric effect: $E_{\text{kin}} = h\nu - \Phi$. The electrons thus released can then be collected and used to generate an electron beam.

In the context of ThomX, the photocathode is made of copper and is illuminated by a laser (see table 1.1). The laser strikes the photocathode in the UV range with a wavelength $\lambda = 257\text{nm}$.¹ The photocathode stay illuminated during 5ps. during which electrons are released and is illuminated every 0.02s..

1.1.2.2 .RF Cavity

An *RF cavity* is a shaped metal container that confines electromagnetic fields at specific frequencies—those in the radio frequency (RF) spectrum, generally from a few hundred megahertz (MHz) to several gigahertz (GHz). The cavity's geometry and size are tailored to resonate at these frequencies, meaning it naturally oscillates with maximum amplitude at these frequencies. The standing electromagnetic waves in the cavity are a result of constructive interference between waves reflecting back and forth within the cavity. The electric field component of these waves oscillates sinusoidally with time and space, creating regions within the cavity where the field strength periodically reaches maximum and minimum values.

When a beam of charged particles, such as protons or electrons, enters

¹The frequency ν of the laser is given by the relationship: $\nu = \frac{c}{\lambda}$, where c is the speed of light in vacuum.

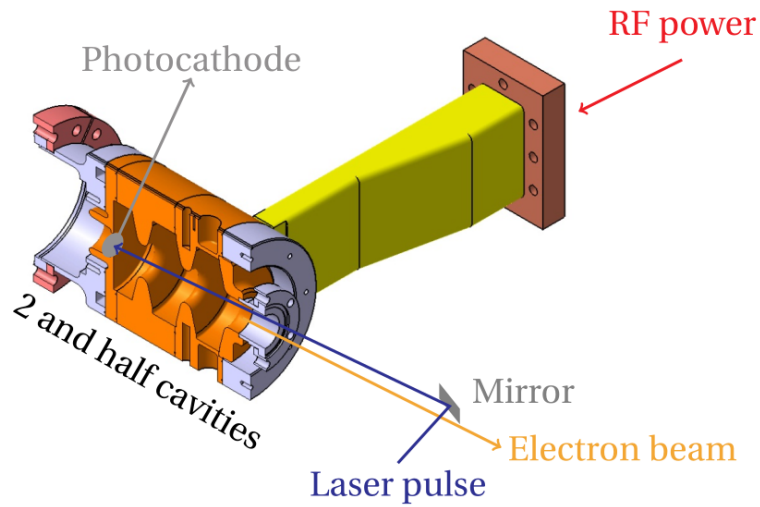


Figure 1.3: ThomX photo-injector, consisting of a photocathode and a two-and-a-half RF cavity gun.

Table 1.2

RF Frequency	2.99855GH.
Accelerating Gradient (E max)	80MV/m
Electron Energy at Exit	5MeV
Beam Charge	1nC

the cavity, it interacts with the oscillating electric field. This interaction's timing is critical and synchronized so that the particles encounter the electric field at phases that accelerate them. This synchronization ensures that as the particles pass through regions of the cavity where the electric field is oscillating, they consistently gain energy. However, the complexity increases due to the finite size of the bunches and the spread in particle velocities within each bunch. As a bunch traverses the cavity, particles at the front may experience a different phase of the electric field than those at the rear, leading to a spread in the energy gain across the bunch.

In a particle accelerator, multiple RF cavities are often combined in series to continuously impart energy to the particle beam, allowing the particles to reach very high energies.

1.1.2.3 .Electron gun

In the ThomX linac, a combination of two and a half cavities forms the *electron gun* (see Fig 1.3). This gun gives an initial impulse to the electrons, allowing them to reach an energy of 5MeV. at the exit. The characteristics of the electron beam at the electron gun are given in Table 1.2.

Table 1.3: Solenoid Parameters

Max solenoid Field	0.25598T.
Focusing Solenoid Length	56mm

The electric field at the cathode level affects the beam creation process in two main ways:

Schottky Effect The *Schottky effect* manifests when an external electric field is applied to the surface of the photocathode, leading to an effective reduction of this work barrier. Mathematically, the work barrier reduction $\Delta\Phi$ due to the Schottky effect can be expressed as:

$$\Delta\Phi = \sqrt{\frac{e^3 E}{4\pi\epsilon_0}}$$

where e is the elementary charge of an electron, E is the intensity of the electric field applied to the surface of the photocathode, and ϵ_0 is the vacuum permittivity. This reduction in the work barrier facilitates electron emission, as less energy is required for an electron to leave the material surface, thus changing the quantity and energy distribution of the emitted electrons.

Current Limitation by Space Charge Effect When the cathode emits electrons, they start to repel each other due to their negative charge. Their mutual repulsion creates an additional electrostatic field that adds to the external electric field applied by the cavity. This phenomenon is called the *space charge effect*. When the electron density is high enough, the resulting electrostatic field from the space charge can counteract the applied electric field that accelerates the electrons away from the cathode. For electrons located at the beam's edges or having a lower initial energy, this electrostatic field can repel them back towards the cathode or away from the beam axis.

1.1.2.4 .Solenoid

A *solenoid* is a type of electromagnet composed of a coil of wire through which an electric current passes. The solenoid is designed to produce a uniform, controlled magnetic field along its longitudinal axis. The primary purpose of a solenoid in a photo-injector is to focus the generated electron beam. When electrons are emitted from the cathode, they tend to diverge due to space charge forces, especially at low energies. The solenoidal magnetic field exerts a Lorentz force on the electrons. The radial component of this force acts to confine and focus the electron beam, counteracting the repulsive forces between electrons and thus minimizing beam divergence.

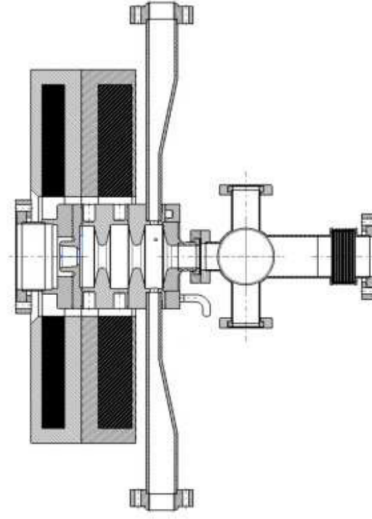


Figure 1.4: Schematic of the electron gun with its solenoids.

Table 1.4: Accelerating section Parameters

Length	4.5m.
Number of Cavities	135
Phase Advance per Cavity	$\frac{2\pi}{3}$
Accelerating Gradient	$14\text{MV} \cdot \text{m}^{-1}$
Electron Energy at Exit	50MeV

ThomX is equipped with two solenoids installed around the photo-injector (see Fig 1.4). One solenoid fulfils the purpose of focusing the beam, while the other cancels the magnetic field at the photocathode location to prevent low-energy electron gyration. The focusing solenoid is mounted on motors for beam-based alignment (an alignment technique working by analyzing the trajectory and properties of the beam itself rather than relying solely on external mechanical measurements).

1.1.2.5 .Accelerating section

The accelerating section of ThomX (see Fig 1.2) is a Traveling Wave Structure (TWS) consisting of 135 RF cavities with a phase advance per cavity of $\frac{2\pi}{3}$. In a TWS, the electromagnetic waves propagate along the structure. This propagation matches the velocity of the accelerated particles, allowing for continuous energy transfer to the particles over a long distance. Characteristics of the accelerating section are given in Table 1.4.

1.1.3 .Beam Characterization

This section describes the physical quantities that qualify electron beam propagation.

A *bunch* is a set of particle $\mathcal{D}(t) = \{\zeta_i(t), i \in \mathcal{S}(t)\}$ where ζ is a particle and $\mathcal{S}(t)$ is the set of particle composing the beam at time t , with $\forall t \in \mathbb{R}_+, \mathcal{S}(t) \subset \mathbb{N}$. Considering that some particles can be lost during the propagation of the beam in the accelerator, the set $\mathcal{S}$ of particles is evolving with time following with $\forall t \in \mathbb{R}_+, \forall \Delta t \in \mathbb{R}_+, \mathcal{S}(t + \Delta t) \subset \mathcal{S}(t)^2$.

Each electron ζ_i can be represented using a seven coordinate system. These seven coordinates for the particle i are its position in the relativistic 4-dimensional time-space $(x_i(t), y_i(t), z_i(t), t_i(t))$ and the three coordinates of its momentum $(p_{x,i}(t), p_{y,i}(t), p_{z,i}(t))$, where t is the clock in the laboratory frame of reference. In a linear accelerator, the z axis is the direction of the beam propagation and is called the longitudinal axis, and x and y are the transversal axes.

The total moment of a particle is the norm of the coordinates: $p_i(t) = \sqrt{p_{x,i}^2(t) + p_{y,i}^2(t) + p_{z,i}^2(t)}$. Their kinetic energy is derived from this total moment: $E_{\text{kin},i}(t) = \sqrt{p_i^2(t) c^2 + m_e^2 c^4} - m_e c^2$, where m_e is the electron mass and c is the speed of light in vacuum.

The following coordinates characterize the angles made by the particles with the axis: $x'_i(t) = \frac{p_{x,i}(t)}{p_i(t)}$ and $y'_i(t) = \frac{p_{y,i}(t)}{p_i(t)}$.

We can also define the Lorentz factor: $\gamma_i(t) = 1 + \frac{E_{\text{kin},i}(t)}{m_e c^2}$ as well as the velocities $v_{x,i}(t) = \frac{p_{x,i}(t)}{\gamma m_e}$, $v_{y,i}(t) = \frac{p_{y,i}(t)}{\gamma m_e}$ and $v_{z,i}(t) = \frac{p_{z,i}(t)}{\gamma m_e}$ as well as their normalized counterparts $\beta_{x,i}(t) = \frac{v_{x,i}(t)}{c}$, $\beta_{y,i}(t) = \frac{v_{y,i}(t)}{c}$ and $\beta_{z,i}(t) = \frac{v_{z,i}(t)}{c}$.

Using these coordinates, we can characterize the whole bunch.

Beam Position The *beam position* is defined as the barycenter of the particles:

²As a first approximation, we will not consider secondary emission of particles due to particle matter interactions

$$x_{avr}(t) = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} x_i(t)$$

$$y_{avr}(t) = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} y_i(t)$$

$$z_{avr}(t) = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} z_i(t)$$

Beam Charge The *total charge* of a beam is the sum of the charges of each particle in the beam: $Q(t) = |\mathcal{S}(t)| e^-$ where e^- is the elementary charge of an individual electron.

Beam Size The *beam size* is defined as the standard deviation of the bunch in each dimension:

$$x_{std}(t) = \sqrt{\frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} (x_i(t) - x_{avr}(t))^2}$$

$$y_{std}(t) = \sqrt{\frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} (y_i(t) - y_{avr}(t))^2}$$

$$z_{std}(t) = \sqrt{\frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} (z_i(t) - z_{avr}(t))^2}$$

Beam Direction The *beam direction* is given by its angle in the transverse plane:

$$x'_{avr}(t) = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} x'_i(t)$$

$$y'_{avr}(t) = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} y'_i(t)$$

Beam Divergence The *beam divergence* is an angular measure of the increase in beam diameter with time and is defined as follows:

$$x'_{\text{std}}(t) = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} (x'_i(t) - x'_{\text{avr}}(t))^2$$

$$y'_{\text{std}}(t) = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} (y'_i(t) - y'_{\text{avr}}(t))^2$$

Beam Energy and Energy Spread Instead of working with the longitudinal coordinates, it is more common to work with the beam energy and its spread:

$$E_{\text{kin,avr}}(t) = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} E_{\text{kin},i}(t)$$

$$E_{\text{kin,std}}(t) = \sqrt{\frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} (E_{\text{kin},i}(t) - E_{\text{kin,avr}}(t))^2}$$

Normalized Emittance The *normalized emittance* of an electron beam quantifies the phase space volume occupied by the electrons in the beam, indicating the beam's quality in terms of its spread in position and momentum. It is invariant under linear beam optics transformations, making it a crucial measure for comparing beam qualities across different sections of an accelerator or different accelerators.

For notation purposes, let's define $\gamma\beta_{x,\text{avr}} = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} \gamma_i(t) \beta_{x,i}(t)$ and

$$\gamma\beta_{x,\text{std}} = \sqrt{\frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} (\gamma_i(t) \beta_{x,i}(t) - \gamma\beta_{x,\text{avr}}(t))^2}.$$

Then $\text{cov}(x(t), \gamma(t) \beta_x(t)) = \frac{1}{|\mathcal{S}(t)|} \sum_{i \in \mathcal{S}(t)} (x_i(t) - x_{\text{avr}}(t)) (\gamma_i(t) \beta_{x,i}(t) - \gamma\beta_{x,\text{avr}}(t))$

Finally, the normalized emittance is defined as the determinant of the variance-covariance matrix between x and $\gamma\beta_x$

$$\epsilon_{x,\text{norm}}(t) = \sqrt{\det \left(\text{cov} \left(\begin{pmatrix} x(t) \\ \gamma(t) \beta_x(t) \end{pmatrix} \right) \right)}$$

$$= \sqrt{x_{\text{std}}^2 * (\gamma\beta_{x,\text{std}})^2 - \text{cov}(x(t), \gamma(t) \beta_x(t))^2}$$

The same process can be used to define $\epsilon_{y,\text{norm}}(t)$ and $\epsilon_{z,\text{norm}}(t)$.

1.1.4 .Diagnostics

1.1.4.1 .Beam Position Monitor (BPM)

A *Beam Position Monitor (BPM)* in a linear particle accelerator is an instrumental device designed to precisely determine the transverse position of a charged particle beam as it propagates through the accelerator structure.

The fundamental principle behind a BPM is the detection of the electromagnetic fields generated by the charged particles in the beam. As these particles move, they induce electric and magnetic fields, which can be detected by sensors placed around the beam pipe. The BPM typically consists of a set of electrodes, positioned around the circumference of the beam pipe. These electrodes are sensitive to the electric field components perpendicular to the beam direction.

The most common configuration of a BPM includes four electrodes placed symmetrically around the beam pipe. This setup allows for the detection of the beam's position in two dimensions: horizontal (x) and vertical (y). The signals from these electrodes are processed to determine the beam's centroid position relative to the geometric center of the BPM.

Mathematically, the position of the beam can be inferred from the signals V from the four electrodes by using the following equations:

$$x_{avr} = k_x \frac{V_{right} - V_{left}}{V_{right} + V_{left}}$$
$$y_{avr} = k_y \frac{V_{top} - V_{bottom}}{V_{top} + V_{bottom}}$$

where k_x and k_y are calibration factors that depends on the geometry of the BPM and the beam's properties. The scale factor is often determined through calibration procedures that involve moving the beam through known positions within the BPM and measuring the corresponding signals. This process establishes a direct relationship between signal ratios and physical displacements. This calibration process is critical for achieving high accuracy in beam position measurements, as it accounts for all the complex interactions between the beam, the BPM electrodes, and the surrounding environment.

On the ThomX accelerator, a BPM is positioned between the electron gun and the accelerating section. There is another one after the RF section. There are used to ensure good beam transport (their signal sum is similar to a charge measurement), and would be used, together with screens, to find and measure any mechanical misalignment, in a so called beam based alignment.

1.1.4.2 .Integrated Current Transformer (ICT)

An *Integrated Current Transformer (ICT)*, in the context of a linear particle accelerator, is a diagnostic device employed to measure the charge of the

particle beam passing through the accelerator. The principle of operation is rooted in Faraday's law of electromagnetic induction, which states that a change in magnetic flux through a loop induces an electromotive force around the loop.

When a charged particle beam, such as electrons or protons, traverses the ICT, it constitutes a moving electric charge, which, according to Maxwell's equations, generates a magnetic field. This magnetic field varies as the beam moves through the ICT, inducing a voltage in the ICT's coil that encircles the beam pipe. The magnitude of this induced voltage is proportional to the time derivative of the beam current with the following relationship $V(t) = \mu_0 A \frac{dI}{dt}(t)$ where A is the area through which the field is passing and $I(t)$ is the instantaneous current of the beam.

To obtain the total charge of the beam, one integrates the induced voltage signal over the time interval of the beam pulse : $Q = \int_{t_1}^{t_2} I(t) dt$ where t_1 and t_2 mark the beginning and end of the beam pulse, respectively.

On the ThomX linear accelerator, two Integrated Current Transformers are positioned: one follows the electron gun, and the other is located after the accelerating section.

1.1.4.3 .Faraday Cup

A *Faraday Cup* is a diagnostic instrument commonly used in linear particle accelerators to measure the charge of particle beams.

The Faraday Cup consists of a conductive cup-shaped container, made of a metal, designed to intercept and collect the charged particles from the beam. When a particle beam enters the cup, the particles are absorbed by the material of the cup, causing a transfer of charge to the cup. This transfer of charge generates an electric current, which can then be measured using an external electrical circuit connected to the cup. By measuring the current, one can extrapolate the total charge of the beam. This process is inherently destructive, as it effectively terminates the particles' trajectory and prevents their further use in experimental procedures.

On the ThomX accelerator, a Faraday Cup is positioned at the end of the linac to perform these measurements.

1.1.4.4 .Beam Loss Monitor (BLM)

A *Beam Loss Monitor* (BLM) in the context of a linear particle accelerator is a diagnostic device designed to detect and quantify the loss of beam particles along the accelerator's length.

The principle of operation for a BLM is fundamentally based on the interaction between the lost beam particles and the materials surrounding the beam pipe. When particles deviate from the intended path, they collide with the accelerator's components or the beam pipe itself. These interactions produce

secondary particles which can be detected. BLMs are positioned along the linac to monitor these secondary emissions.

BLM data is essential for understanding the beam dynamics and optimizing the accelerator's performance. By analyzing the patterns and locations of beam losses, physicists can identify and rectify alignment errors, RF phase instabilities, or focusing deficiencies in the accelerator structure.

1.1.4.5 .Screen Station

A *Screen Station* (SST) intercepts the particle beam using a screen or a set of screens. These screens are typically made of materials such as scintillators, which emit light when struck by charged particles, or other materials that can produce secondary emissions or alterations when interacted with by a high-energy particle beam. When the beam strikes the screen, the particles induce emissions (e.g., photons in the case of a scintillator) that are proportional to the intensity of the beam at the point of interaction. By capturing and analyzing this emitted radiation, usually with a camera or other optical detection system positioned outside the beam path, one can infer the spatial distribution of the beam at the SST location. The data obtained from an SST allows for the reconstruction of the beam profile, which includes information about the beam's transverse dimensions (width and height), shape and position.

1.2 .Particle Accelerator Simulators

Particle accelerator simulators are essential tools used to predict how particle beams behave as they travel through an accelerator. These tools allow researchers to model complex interactions between particles and the fields that guide and accelerate them, which is critical for optimizing the design and operation of accelerators [7, 45]. In this context, simulators provide a virtual environment to test different setups, fine-tune parameters, and identify potential issues before building or operating the physical accelerator.

One of the key aspects in simulating particle accelerators, especially for systems like linear accelerators (linacs), is the need to model space charge effects. These effects occur when particles within a beam, all carrying the same charge, repel each other due to Coulomb forces. At lower beam energies or when dealing with dense particle beams, space charge effects can strongly influence beam behavior, leading to issues like beam spreading or reduced beam quality [53]. Accurately capturing these effects is essential for predicting beam dynamics, and simulators like ASTRA [22] (A Space Charge Tracking Algorithm) are designed to handle these conditions effectively.

ASTRA utilizes the Particle-in-Cell (PIC) method, a computational technique designed to simulate systems with interacting particles, particularly charged particles such as those found in particle accelerators [30]. The method is

well-suited for modeling the complex interactions between particle beams and electromagnetic fields, especially in scenarios where space charge effects are significant. By combining particle motion tracking with field calculations on a spatial grid, the PIC method allows for efficient and accurate simulations of these interactions. The steps involved in this method can be outlined in a detailed process, which follows.

1. **Particles and Grids:** In a PIC simulation, particles are represented as "macro-particles," each of which aggregates the properties of a large number of real particles. The macro-particles have associated quantities such as charge q and mass m , and they move through a continuous phase space $((\mathbf{x}, \mathbf{v}))$, where $\mathbf{x}$ is position and $\mathbf{v}$ is velocity. The simulation space is divided into a discrete mesh or grid, typically regular in shape, where the electric and magnetic fields will be computed. Importantly, the particles do not reside on the grid; the grid is used solely to compute the fields, while the particles move continuously within this space.
2. **Charge Deposition to the Grid:** As the particles move through space, their charge is assigned (or "deposited") to the surrounding grid points to create a charge density field $\rho(\mathbf{x})$ on the grid. The charge from each particle is distributed to the nearest grid points using a weighting function $W(\mathbf{x}_p, \mathbf{x}_g)$, which ensures charge conservation. The total charge density at each grid point $\mathbf{x}_g$ is the sum of contributions from all nearby particles:

$$\rho(\mathbf{x}_g) = \sum_p q_p W(\mathbf{x}_p, \mathbf{x}_g),$$

where q_p is the charge of particle p , and W is a weighting function that distributes the particle's charge to nearby grid points.

3. **Solving Field Equations:** Once the charge density $\rho(\mathbf{x})$ is computed on the grid, the electric field $\mathbf{E}(\mathbf{x})$ and the magnetic field $\mathbf{B}(\mathbf{x})$ is obtained by solving Maxwell's equation (see Section 1.1.1.2). These equations are solved numerically on the grid using techniques such as finite-difference methods. These fields are what cause the particles to either accelerate or deflect as they move through the accelerator.
4. **Interpolation of Fields to Particle Positions:** After the electric and magnetic fields are computed at the grid points, they must be interpolated back to the position of each particle. This means that for every particle, the simulator calculates the electric and magnetic forces acting on it based on the nearby grid points.
5. **Updating Particle Motion:** The motion of each particle is then updated using the Lorentz force law (see Eq. (1.5)). Typically, numerical

integration schemes like Runge-Kutta method are used to update particle positions and velocities over discrete time steps. Once the particle motions are updated, the cycle repeats, calculating the new charge distribution and fields at the next time step.

The accuracy and efficiency of a particle accelerator simulation depend on how key inputs are defined. These inputs can be divided into three main categories: the initial particle distribution, the configuration of accelerator components, and the numerical parameters that govern the simulation's execution.

- **Initial Particle Distribution:** The initial particle distribution in a simulation reflects the beam as it is produced by the particle source of the accelerator, typically modeled using macro-particles to represent groups of real particles. A Gaussian distribution is often used in position and velocity space because it closely approximates the spread of particles emitted from sources like electron guns or photo-injectors. Key parameters such as beam energy and emittance are usually derived from experimental data or design specifications, with emittance reflecting the beam's quality and focusability.
- **Accelerator Components and the Lattice:** The configuration of the accelerator's magnetic and electric elements, collectively referred to as the lattice, defines the path that the beam follows. This lattice is composed of dipoles, quadrupoles, and RF cavities, each playing a specific role in bending, focusing, and accelerating the beam. In the simulation, these elements are specified by providing detailed field maps or using idealized models based on design parameters. These inputs allow the simulator to recreate the precise environment through which the beam propagates, ensuring that the interactions between the beam and the lattice are accurately modeled.
- **Numerical Inputs:** Numerical parameters such as grid resolution, the number of macro-particles, and time step size govern the simulation's accuracy and runtime. A higher grid resolution and more macro-particles improve the precision of field calculations and space charge effects, but also increase computational demands. Smaller time steps ensure accurate tracking of particle motion, particularly in rapidly changing fields, but can significantly extend the runtime. Therefore, striking a balance between resolution, particle count, and time steps is key to achieving accurate results without excessive computation time.

The outputs from particle accelerator simulations provide comprehensive information about the beam's behavior as it travels through the accelerator.

These outputs can be grouped into two essential aspects: the detailed particle-level data and the derived beam properties.

- The **particle distribution** data offers a comprehensive view of the beam, capturing the position, velocity, and energy of each macro-particle. This data represents the full state of the beam, making it valuable for in-depth analysis of beam dynamics and interactions. However, storing this data at multiple points along the accelerator consumes significant disk space, and its complexity makes it more difficult to analyze efficiently.
- In contrast, the derived **beam properties** (such as emittance, beam size, and energy spread, see Section 1.1.3) provide more easily interpretable metrics that are useful for assessing overall beam quality. These properties are the key indicators physicists rely on to evaluate performance and guide adjustments to the accelerator. However, they offer only a condensed view of the beam, potentially overlooking finer details captured in the full particle distribution.

Particle accelerator simulators, like ASTRA, are essential for accurately modeling beam behavior through the Particle-in-Cell method, capturing complex interactions like space charge effects. By carefully defining the initial particle distribution, configuring the accelerator's lattice, and selecting appropriate numerical parameters, these simulations can offer detailed particle-level data and key beam properties. However, the high computational cost of running these simulations, especially with fine resolution or multiple runs, presents challenges. Surrogate models, built upon the principles outlined here, are an avenue to provide an efficient alternative, approximating key aspects of the full simulator while reducing computational requirements.

2 -State of the Art for Surrogate Modeling: The case of Particle Accelerators

Contents

2.1 Surrogate Modeling: The Context of Particle Accelerators	30
2.2 Data Acquisition for Particle Accelerator Simulators Modeling	31
2.3 Artificial Intelligence in Particle Accelerator Modeling	32
2.3.1 Offline Optimization of Accelerator Operations	33
2.3.2 Online Control of Accelerator Operations	33
2.3.3 Fault Detection and Diagnostics	34
2.4 Data Structure in Particle Accelerator Simulations	34
2.5 The Multi-Objective Nature of Particle Accelerator Simulator Modeling	36
2.5.1 Scalarization Methods	37
2.5.2 Gradient-Based Adaptive Methods	38

In the field of particle accelerator physics, complex simulations are essential for predicting beam dynamics, optimizing machine settings, and guiding experimental designs. However, these simulations are computationally expensive, often requiring significant time and resources. This is where surrogate modeling comes into play, offering a computationally efficient alternative by approximating the behavior of detailed simulators. Surrogate models enable faster evaluations, making them invaluable for tasks like optimization, uncertainty quantification, and control.

The goal of this chapter is to present the state of the art in surrogate modeling within the context of particle accelerators. A comprehensive review of the entire surrogate modeling literature is beyond the scope of this discussion, so the focus will be narrowed to several key areas relevant to the modeling of particle accelerator systems.

First, Section 2.1 briefly recalls the context of particle accelerators and explores why surrogate models are increasingly needed in this domain. Next, Section 2.2 delves into the process of creating such a surrogate model, based on interaction with a particle accelerator simulator to construct a relevant

database. Following this, Section 2.3 examines the contributions of artificial intelligence (AI) within the accelerator physics community.

Another important aspect is the nature of the data involved in particle accelerator simulations. These simulations generate high-dimensional, complex datasets, and effectively representing this data is a key challenge for machine learning models. Section 2.4 reviews the types of data typically produced by particle accelerator simulators and explore the machine learning models best suited to handling these data in surrogate modeling tasks.

Finally, Section 2.5 discusses the concept of multi-task learning as a strategy for addressing the various objectives that arise in surrogate modeling for particle accelerators.

2.1 .Surrogate Modeling: The Context of Particle Accelerators

Particle accelerators are complex systems requiring precise simulation of beam dynamics, particle interactions, and electromagnetic fields (see Section 1.2 for details on the simulation process). These high-fidelity simulations are computationally intensive [77], often involving the resolution of complex differential equations across large parameter spaces. For instance, modeling beam behavior in particle accelerators can take a few minutes to hours to days of computation per simulation, making traditional methods impractical for large-scale design exploration and iterative optimizations. For the simulation of the ThomX Linac using Astra [22], each run with a reasonable set of numerical parameters takes approximately 10 minutes to complete.¹

The need to explore diverse accelerator configurations and optimize performance makes surrogate models indispensable. By approximating the behavior of high-fidelity simulations, surrogate models drastically reduce computational costs [3]. This allows researchers to perform rapid parameter sweeps and evaluate multiple design alternatives that would otherwise be computationally prohibitive. Surrogate models, such as those based on Gaussian Processes [28] or Neural Networks [41], achieve high accuracy while reducing simulation times, enabling efficient exploration of complex, multi-dimensional parameter spaces.

One key application of surrogate models is in optimization tasks. Particle accelerator design involves balancing multiple objectives, such as minimizing beam emittance, reducing energy dispersion, and optimizing charge transmission. Traditional optimization methods, which require a significant number of evaluations, are infeasible when each evaluation involves a full-scale simulation. Surrogate models enable quicker evaluations, allowing for more iterations and better refinement of designs. They have been shown to improve perfor-

¹Further details on the selection of these numerical parameters are provided in Chapter 4 3.

mance in real-world scenarios, such as beamline optimization, by significantly accelerating convergence to optimal solutions [20].

Additionally, surrogate models play a critical role in uncertainty quantification. Accurately quantifying uncertainties is essential for evaluating the robustness and reliability of accelerator designs [1, 15]. Surrogate models propagate uncertainties from inputs to outputs, enabling researchers to assess the confidence levels of their predictions without the need for repeated high-cost simulations. This capability is particularly useful in the context of risk management, where decision-making is based on both performance and the associated uncertainties of the system.

As the scale and complexity of particle accelerators increase, the reliance on surrogate models is likely to grow. Their potential to deliver cost-effective, high-fidelity approximations positions them as promising tools for improving both the design and operational efficiency of future particle accelerators.

2.2 .Data Acquisition for Particle Accelerator Simulators Modeling

In traditional machine learning (ML) frameworks, models are typically trained on pre-existing datasets. However, in the context of surrogate modeling for particle accelerators, the problem is twofold: first, determining how to select data points (control parameters) to query from a high-fidelity simulator; and second, how to effectively train a model based on these data points. Careful interaction with the simulator is essential to ensure that the data used to train the surrogate model accurately captures the system’s behavior.

This process can be approached in two primary ways:

1. **Decoupling data selection and model learning:** In this approach, a database is generated in advance, aiming for a uniform coverage of the potential control parameter space. The model is then trained on this pre-constructed database. Data points are typically selected randomly across the input space. ²
2. **Coupling data selection with model learning through Active Learning:** This strategy involves an iterative process where the most informative data points are selected for evaluation, improving model accuracy with fewer samples [64, 65, 56]. ³

²Random sampling may result in uneven coverage, with clusters of points and gaps [23]. To mitigate this, Quasi-Monte Carlo methods such as Latin Hypercube Sampling (LHS) [44], Sobol [67], or Hammersley [27] sequences are commonly employed. These methods tend to reduce the expected discrepancy — a formal measure of sampling uniformity — though their advantages diminish in high-dimensional spaces. Further details are provided in Appendix A.

³Further details on Active Learning are provided in Appendix A

While Active Learning is appealing due to its efficiency in selecting data points, it introduces additional complexity. Specifically, it requires a new database to be generated for each model under consideration, which increases experimental complexity. Furthermore, active learning is not without theoretical challenges, as it can introduce sampling bias [16]. Due to these practical limitations and theoretical concerns, this work adopts the first approach: generating a fixed database and then training a model on this dataset.

A detailed description of the database is provided in Chapter 3. However, for clarity in the remainder of this chapter, we briefly summarize the structure of a data point in the database.

A data point consists of a set of control parameters (defining a potential configuration of the particle accelerator) and the corresponding simulation results. The control parameters are represented as a real-valued vector, while the simulation results are a vector of observable values (such as those outlined in Chapter 1.1.3) recorded at multiple positions along the accelerator. Additionally, the full description of the simulated beam may accompany these observables.

This database raises two key challenges:

1. How to accurately represent the full description of the simulated beam in the model? This issue is discussed in detail in Section 2.4.
2. How to account for predictions at multiple positions along the accelerator? A reliable surrogate model must not only provide accurate predictions at each position but also ensure coherence between predictions at different positions. This leads to a multi-objective problem that requires simultaneous optimization. Solutions to these multi-objective challenges are explored in Section 2.5.

2.3 .Artificial Intelligence in Particle Accelerator Modeling

Machine learning (ML) has become an essential tool in the operation and optimization of particle accelerators [19, 21]. It allows for data analysis, enabling more precise control and optimization of accelerator parameters [20]. ML techniques such as regression [20], classification [54], and reinforcement learning [35] have been employed to enhance performance and operational efficiency.

Surrogate models approximate the behavior of complex systems, offering faster computations while maintaining high accuracy. In particle accelerators, they are particularly valuable due to the high computational cost of direct simulations [8]. These models are crucial for real-time performance applications, such as control systems and optimization tasks.

2.3.1 .Offline Optimization of Accelerator Operations

Offline optimization in particle accelerators involves determining the optimal parameters to achieve specific objectives, such as minimizing beam emittance, reducing energy dispersion, and maximizing charge transmission. These tasks are typically performed on simulators before actual operations to establish the best conditions. Traditional optimization methods like the Nelder-Mead Simplex Algorithm [32, 49, 52] and Genetic Algorithms have been widely used [26, 7, 6, 31, 70, 5]. For instance, the Nelder-Mead algorithm has been effectively applied at CERN to optimize ion beam transmission, demonstrating significant improvements in beam handling [49].

Surrogate models play a crucial role in offline optimization by approximating complex simulations, thus significantly reducing computational costs. This allows for more extensive exploration of parameter spaces and quicker convergence to optimal solutions. For example, surrogate models based on Gaussian Processes [18, 28, 55] have been employed to accelerate the optimization of beamline configurations, resulting in efficient and accurate parameter tuning.

Additionally, modern approaches often involve training a surrogate model, such as a neural network, to capture the behavior of the accelerator. These surrogate models can then be combined with optimization algorithms like Genetic Algorithms (GAs) or Bayesian Optimization to perform surrogate optimization. This method has been shown to enhance the performance and robustness of the optimization process, leading to better overall results in accelerator tuning [42, 20, 38].

2.3.2 .Online Control of Accelerator Operations

Real-time control of particle accelerators is essential for maintaining optimal performance and addressing dynamic changes in operational conditions. Extremum Seeking Control (ESC) [4] is a prominent technique used for real-time optimization [61]. ESC works by continuously adjusting control parameters to maximize or minimize a performance criterion, based on the system's response to small periodic perturbations. This method has been successfully implemented at the Facility for Advanced Accelerator Experimental Tests (FACET) at SLAC, where it provided real-time diagnostics of electron bunch profiles [59] and has been enhanced to incorporate safety constraints [60, 74].

Surrogate models enhance online control by offering rapid approximations of system behavior, enabling real-time adjustments without requiring extensive computational resources [62, 58].

Additionally, reinforcement learning (RL) methods have emerged as powerful tools for online control of particle accelerators [35, 36]. RL algorithms learn optimal control policies through interaction with the accelerator environment, effectively adapting to dynamic conditions and uncertainties.

2.3.3 .Fault Detection and Diagnostics

Fault detection and diagnostics are critical for ensuring the reliability and safety of particle accelerators. Surrogate models contribute to this area by predicting potential faults and diagnosing issues based on operational data. By learning from historical data, these models can identify patterns indicative of faults, allowing for proactive maintenance and minimizing downtime.

For example, machine learning techniques have been employed to predict particle accelerator failures using binary classifiers, providing early warnings and enabling timely interventions [54]. This predictive capability helps maintain the high performance and safety standards in accelerator operations.

2.4 .Data Structure in Particle Accelerator Simulations

This section explores the effective utilization of data contained within the simulated particle beam. At the heart of a simulation lies the particle beam, which serves as a fundamental entity in accelerator studies.

As described in Chapter 1.2, a simulated particle beam consists of a collection of macro-particles, each represented by a 9D coordinate vector.⁴ These coordinates are recorded at multiple positions along the accelerator.

The particle beam encapsulates a rich representation of the underlying data, serving as the connection between the simulation's control parameters and the resulting observables. This representation contains far more information than the observables alone. Leveraging the full scope of this data enhances the robustness of the model by providing a more comprehensive foundation for its predictions. Furthermore, the model's trustworthiness is strengthened, as all predicted observables emerge from a coherent underlying cause: the prediction of the macro-particles themselves.

Given that the particle beam consists of a set of 9D data points, its representation falls within the scope of point cloud modeling.

Point cloud modeling is crucial for the accurate representation and simulation of multidimensional data. Traditional approaches to processing point clouds often involve transforming the data into structured forms such as voxel grids or meshes. While these methods can be effective, they frequently result in substantial data loss and computational inefficiencies, particularly given the high dimensionality and sparsity of particle cloud data.

We present in the following the main approaches to model a point cloud.

- **Direct Processing of Point Clouds:** The advent of PointNet [11] revolutionized point cloud processing by directly consuming unordered point

⁴The 9 dimensions include three spatial coordinates, three momentum coordinates, one time coordinate (since particles are relativistic and possess intrinsic time), one charge coordinate, and a flag indicating whether the particle is *alive* (i.e., it has reached the position where its coordinates are recorded) or *dead* (i.e., lost during tracking).

sets. Unlike previous methods that required transforming point clouds into structured formats, PointNet operates directly on raw point sets, thus preserving the original geometric information and reducing computational overhead. PointNet applies shared multilayer perceptrons (MLPs) to each point independently, transforming the input point $\mathbf{p}_i$:

$$\mathbf{h}_i = \gamma(\mathbf{p}_i) = \text{MLP}(\mathbf{p}_i),$$

where γ denotes the transformation function. To capture the global structure, PointNet aggregates these point-wise features using a symmetric function such as max pooling:

$$\mathbf{h}_{\text{global}} = \text{MAX}(\{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n\}).$$

While PointNet effectively captures the overall structure of point clouds, it has limitations in capturing fine-grained local features. This led to the development of PointNet++ [51], which enhances PointNet by introducing a hierarchical learning approach. PointNet++ segments the point cloud into overlapping local regions and applies PointNet recursively to capture local geometric structures at multiple scales:

$$\mathbf{h}_i^{(k)} = \text{PointNet}(\mathcal{N}(\mathbf{p}_i^{(k-1)})),$$

where k denotes the level in the hierarchy, and $\mathcal{N}(\mathbf{p}_i)$ represents the local neighborhood around point $\mathbf{p}_i$.

- **Convolutional Approaches for Point Clouds:** Building on the need to better capture local features, KPConv [69] was introduced in 2019, which adapts traditional convolution operations to point clouds. PConv uses kernel points distributed in the local neighborhood to perform the convolution operation:

$$\mathbf{h}_i = \sum_{j=1}^K w_j \phi(\mathbf{p}_i - \mathbf{k}_j),$$

where w_j are learnable weights, ϕ is a nonlinear transformation, and K is the number of kernel points. By aligning the kernel points with the local geometry of the point cloud, KPConv provides flexibility and improves the ability to capture spatial relationships.

- **Attention Mechanisms:** Expanding on the idea of capturing spatial relationships dynamically, the Point Transformer model was introduced [78], which leverages self-attention mechanisms. In Point Transformer, each point $\mathbf{p}_i$ attends to all other points in the set, dynamically weighing their importance through attention coefficients:

$$\mathbf{h}_i = \sum_{j=1}^n \alpha_{ij} (\mathbf{W}_v \mathbf{p}_j),$$

where α_{ij} is calculated as:

$$\alpha_{ij} = \frac{\exp(\mathbf{q}_i \cdot \mathbf{k}_j)}{\sum_{l=1}^n \exp(\mathbf{q}_i \cdot \mathbf{k}_l)},$$

with $\mathbf{q}_i = \mathbf{W}_q \mathbf{p}_i$ and $\mathbf{k}_j = \mathbf{W}_k \mathbf{p}_j$ being the query and key projections, respectively. This mechanism effectively captures long-range dependencies and contextual relationships within the point cloud.

Fast Point Transformer [47] enhanced this approach, optimizing the attention mechanism to reduce computational complexity.

- **Graph-Based Neural Networks:** Recognizing the importance of explicitly modeling relationships between points, Wang et al. proposed DGCNN [72], which constructs dynamic graphs for point clouds. In DGCNN, edges are defined based on nearest neighbors in the feature space, and edge convolutions are performed:

$$\mathbf{h}_i = \max_{j \in \mathcal{N}(i)} \phi(\mathbf{h}_i, \mathbf{h}_j - \mathbf{h}_i),$$

where $\mathcal{N}(i)$ denotes the neighborhood of point $\mathbf{p}_i$, and ϕ is a nonlinear transformation. The graph structure is dynamically updated as the features evolve, effectively capturing local geometric structures.

- **Hybrid Approaches:** To leverage the strengths of both point-based and voxel-based representations, Shi et al. introduced PV-RCNN [66]. Initially, the point cloud is voxelized, and 3D convolutional layers extract voxel features. These voxel features are then combined with point features to enhance the overall feature representation:

$$\mathbf{h}_{\text{combined}} = f_{\text{point}}(\mathbf{p}) + g_{\text{voxel}}(\mathbf{v}),$$

where f_{point} and g_{voxel} are functions representing the point-based and voxel-based feature extraction networks, respectively. This hybrid approach effectively balances detailed local feature extraction with broader contextual understanding.

2.5 .The Multi-Objective Nature of Particle Accelerator Simulator Modeling

As outlined in Section 2.2, the simulation output is recorded at multiple locations along the accelerator, each representing a distinct objective. Specifically, for the p -th position, the goal is to accurately predict the output at that point based on the initial control parameters. Additionally, we can define intermediate objectives, requiring the surrogate model to predict the output

at the p -th position given the state of the particle beam at a preceding location. The first set of objectives ensures the model's accuracy, while the second set guarantees internal consistency across the accelerator's sections.

Handling these multiple objectives, while potentially non-conflicting, introduces a level of complexity that must be managed. Traditional Machine Learning frameworks typically optimize a single loss function, which is minimized via gradient descent, ultimately converging to a single solution. However, when dealing with multiple objectives, a more rigorous definition of a solution is required.

Mathematically, let $\mathcal{L}_i(\theta)$ represent the loss function for task i with model parameters θ . The overall objective in Multi-Objective Optimization (MOO) for Multi-Task Learning (MTL) can be formulated as:

$$\min_{\theta} [\mathcal{L}_1(\theta), \mathcal{L}_2(\theta), \dots, \mathcal{L}_n(\theta)]$$

A solution θ^* is Pareto optimal if there is no other solution θ such that $\mathcal{L}_i(\theta) \leq \mathcal{L}_i(\theta^*)$ for all i and $\mathcal{L}_j(\theta) < \mathcal{L}_j(\theta^*)$ for at least one j . This concept is used to identify a set of trade-off solutions optimal for all tasks involved, forming the Pareto front.

The hypervolume metric quantifies the volume covered by the Pareto front, assessing the diversity and quality of solutions. Mathematically, the hypervolume HV of a Pareto front P is defined as:

$$HV(P, r) = \lambda \left(\bigcup_{p \in P} \mathcal{H}(r, p) \right)$$

where λ denotes the Lebesgue measure, r is a reference point, p represents a Pareto optimal solution and $\mathcal{H}(r, p)$ is the hyperrectangle formed by r and p .

A larger hypervolume indicates that the Pareto front covers a more extensive area of the objective space.

A key challenge in Multi-Objective Optimization (MOO) lies in balancing trade-offs between the competing objectives. This is typically addressed by reducing the multi-objective problem to a single-objective one through one of two main strategies: (a) **Scalarization Methods**, which assign weights to each loss function, either predetermined or dynamically adjusted based on the other losses. The gradient of the resulting weighted loss is then computed to guide model updates (see Section 2.5.1). (b) **Gradient-Based Adaptive Methods**, where the gradients for each task are computed individually and then aggregated based on their interrelationships (see Section 2.5.2).

2.5.1 .Scalarization Methods

One approach to solving the MOO problem is scalarization, where the multi-objective problem is converted into a single-objective problem by assigning

weights to each task. The weighted sum method is the most straightforward scalarization technique, expressed as:

$$\mathcal{L}_{\text{weighted}}(\theta) = \sum_{i=1}^n \alpha_i \mathcal{L}_i(\theta)$$

where α_i are the weights assigned to each task. Although simple, choosing appropriate weights α_i can be challenging.

Moreover, a significant limitation of fixed scalarization is its inability to handle non-convex Pareto fronts effectively. In cases where the Pareto front is non-convex, a fixed set of weights α_i may not be able to capture all optimal trade-offs. This results in some regions of the Pareto front being unattainable, as the weighted sum method can only represent solutions on the convex hull of the Pareto front. Consequently, this restricts the diversity of solutions and limits the ability to find truly optimal or near-optimal solutions across the entire Pareto front.

The Tchebycheff method improves on this by minimizing the weighted maximum deviation from a reference point z^* (typically set to 0 when minimizing a loss):

$$\mathcal{L}_{\text{Tchebycheff}}(\theta) = \max_i \{\alpha_i |\mathcal{L}_i(\theta) - z_i^*|\}$$

This approach emphasizes the task that is performing the worst, ensuring more balanced performance across tasks.

2.5.2 .Gradient-Based Adaptive Methods

Gradient-based optimization methods also play a significant role in MTL. Multi-Objective Gradient Descent (MOGD) extends traditional gradient descent to handle multiple objectives by combining gradients from different tasks. Given the gradients $\nabla \mathcal{L}_i(\theta)$ for each task i , the MOGD update rule is:

$$\theta \leftarrow \theta - \eta \sum_{i=1}^n \alpha_i \nabla \mathcal{L}_i(\theta)$$

where η is the learning rate and α_i are the weights for each task.

However, gradients from different tasks can conflict, meaning they can point in opposing directions, leading to inefficient learning or suboptimal solutions. Addressing these conflicting gradients is crucial for effective MTL.

Several methods have been developed to handle conflicting gradients:

One notable method, GradNorm [13], tackles the issue of task imbalances by dynamically normalizing gradient magnitudes. This is achieved by adapting task-specific weights $\alpha_i(t)$ during training to ensure that tasks with larger gradients do not dominate the learning process. The method minimizes discrepancies in gradient magnitudes by continuously adjusting $\alpha_i(t)$ to equalize the influence of all tasks, promoting more balanced training progress.

Building on this foundation, Sener and Koltun tackled MTL by explicitly formulating it as a multi-objective optimization (MOO) problem [63]. Their approach leverages the Multiple Gradient Descent Algorithm (MGDA) to find a common descent direction that takes all tasks into account. MGDA solves the following optimization problem:

$$\min_{\alpha_1, \dots, \alpha_n} \left\| \sum_{i=1}^n \alpha_i \nabla_{\theta_{sh}} \mathcal{L}_i(\theta_{sh}, \theta_i) \right\|^2 \quad \text{subject to} \quad \sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0$$

Here, θ_{sh} represents the shared parameters across tasks, while θ_i are task-specific parameters. This formulation ensures that the optimization step finds a common descent direction by minimizing the weighted sum of gradients, with the solution lying in the convex hull of the task gradients. By selecting the minimum-norm element ω from this convex hull, MGDA ensures that all tasks contribute equally to the final gradient descent step. A key feature of this approach is that when ω lies in the interior of the convex hull, the Fréchet derivatives for each task are equal, guaranteeing that each task has an equal influence on the shared parameters. To efficiently find the minimum-norm solution in high-dimensional spaces, Sener and Koltun utilized a Frank-Wolfe-based optimization algorithm [33].

Another approach, PCGrad [76], proposed by Yu et al., specifically addresses the problem of gradient interference. In multi-task settings, conflicting gradients—those with negative cosine similarity—can hinder effective learning. PCGrad mitigates this by adjusting gradients when conflicts arise. The process begins by identifying conflicting gradients between tasks i and j :

$$\text{CosSim}(\nabla \mathcal{L}_i, \nabla \mathcal{L}_j) = \frac{\nabla \mathcal{L}_i \cdot \nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_i\| \|\nabla \mathcal{L}_j\|}$$

If the cosine similarity is negative, PCGrad projects the gradient of task i onto the normal plane of task j , removing the conflicting component:

$$\nabla \mathcal{L}_i^{\text{proj}} = \nabla \mathcal{L}_i - \frac{\nabla \mathcal{L}_i \cdot \nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|^2} \nabla \mathcal{L}_j$$

This modification allows the gradients to remain in harmony, preventing negative interactions while preserving task-specific learning. The modified gradients are then applied during the optimization step, ensuring that tasks do not interfere with each other's progress, particularly in cases where their objectives are in conflict.

Extending this idea further, Liu et al. introduced Conflict-Averse Gradient Descent (CAGrad) [43], which seeks to find a unified gradient that both reduces conflicts and allows progress across all tasks. CAGrad formulates this as an optimization problem at each training step, where the unified gradient

g minimizes the deviation from individual task gradients while ensuring no conflicts arise. The objective is to solve:

$$\min_g \sum_{i=1}^n \|\nabla \mathcal{L}_i - g\|^2 \quad \text{subject to} \quad g \cdot \nabla \mathcal{L}_i \geq 0 \quad \forall i$$

The constraint ensures that the unified gradient does not conflict with any individual task gradient, allowing it to align with all tasks while minimizing interference. This approach differs from PCGrad in that it directly optimizes a unified direction that works for all tasks, rather than resolving conflicts through projection. The result is a gradient that advances all tasks, making CAGrad particularly effective in situations where tasks frequently exhibit opposing objectives.

Despite these advances, both Kurin et al. [40] and Xin et al. [75] raised questions about the necessity of such complex multi-task optimization techniques. Their evaluations showed that simpler methods, such as unitary scalarization where each task is equally weighted with $\alpha_i = 1$, often perform on par with, or even better than, specialized approaches. Moreover, they argue that many of these optimizers function as regularization methods, suggesting that the benefits they offer may not justify their complexity and computational cost. Their empirical evaluations advocate for simpler methods that reduce computational overhead while maintaining robust performance in MTL settings.

3 -Towards Particle Accelerator Simulator Modeling: Database and Supervised Learning Framework

Contents

3.1 Database Generation	41
3.1.1 Simulation Overview	42
3.1.2 Macro-Particle Tradeoff	44
3.1.3 Control Parameter Sampling	45
3.1.4 Simulation Results Overview	48
3.2 Performance Indicators	50
3.3 Conclusion	60

This chapter presents the framework for our investigation, including the database used and the evaluation metrics for assessing model performance. It is organized as follows:

Section 3.1 provides a detailed overview of our first contribution, which consists of a database created for our experiments. Unlike the custom datasets often used in the existing literature, our approach involves randomizing a wide range of hyperparameters to create a comprehensive dataset. Furthermore, this database is made publicly available¹ to facilitate benchmarking and broader research applications. This section details the simulation parameters, the sampling distributions for the control parameters and the resulting dataset generated by the simulator, providing a thorough understanding of the experimental data.

Section 3.2 describes the performance metrics used to evaluate the accuracy of a surrogate model. This section also presents reference values that serve as benchmarks for consistent and objective model evaluation.

3.1 .Database Generation

This section presents the database created as part of our investigation into modeling the Linac of the ThomX accelerator. The database is the result of extensive simulations conducted using the ASTRA (A Space-charge TRacking Algorithm) simulator [22] (see Chapter 1.2), a Particle-In-Cell (PIC) simulator

¹<https://zenodo.org/records/11084340>

widely used for low-energy, high-intensity electron beams². The database serves as the basis for the development and evaluation of surrogate models, as well as the conduct of experiments.

The subsections of this section are organised in a way that facilitates a comprehensive understanding of the database generation process. Subsection 3.1.1 provides a detailed account of the process of a single simulation, specifying the input parameters and the resulting outputs generated by ASTRA. This subsection also introduces the notations that will be used throughout the remainder of this work. Subsection 3.1.2 discusses how we choose the parameters controlling the precision of the simulations. Subsection 3.1.3 describes the methodology used for sampling the control parameters, highlighting the approach adopted to ensure a diverse and comprehensive dataset. Finally, Subsection 3.1.4 presents the resulting database, showcasing the various outputs and summarizing the data obtained for experimental use.

3.1.1 .Simulation Overview

The foundational mechanisms of a Particle-In-Cell (PIC) simulation have been outlined in Chapter 1.2. In this section, we introduce specific notations that enable a formal description of the generated database, providing the precision required for interpreting the simulation results.

Each simulation begins with an initial set of particles and iteratively solves the field and motion equations to update this particle set at each integration step. Two types of inputs guide these simulations:

1. **Control Parameters ($\mathbf{c}$):** The control parameters, denoted by $\mathbf{c}$, are a real vector that describes the configuration of the accelerator. These parameters define the initial distribution properties and the physical state of the accelerator, encompassing elements such as magnetic and electric field strengths, alignment of various components, and material-dependent properties that affect particle dynamics.
2. **Numerical Parameters:** These parameters influence the simulation's precision. A primary example is the number of macro-particles, which determines the discretization granularity within the simulation. While these parameters are essential for accurate simulation and are discussed in Section 3.1.2, they remain constant across experiments. Consequently, they are not explicitly represented in the notation.

Let $\mathcal{D}(t, \mathbf{c})$ represents the particle distribution at time t , determined by control parameters $\mathbf{c}$. The particle distribution function, denoted by $\mathcal{D}(t, \mathbf{c})$ can

²To facilitate the generation of different simulations, we have developed a Python wrapper called `pyastra`, which allows for easy randomization of control parameters and parallel execution on multiple cores. The code is available at <https://gitlab.in2p3.fr/AcceleratorMachineLearning/pyastra.git>.

be interpreted as the collection of particles present at a given time t resulting from the control parameters $\mathbf{c}$. Formally, we define:

$$\mathcal{D}(t, \mathbf{c}) = \{\zeta_i(t, \mathbf{c}), i \in \mathcal{S}(t, \mathbf{c})\}, \quad (3.1)$$

where each $\zeta_i(t, \mathbf{c})$ corresponds to a macro-particle, and $\mathcal{S}(t, \mathbf{c}) \subset \mathbb{N}$ denotes the set of alive particle indices at time t ³.

Each macro-particle ζ_i is represented in an eight-coordinate system, encompassing its position in relativistic four-dimensional space-time ($x_i(t)$, $y_i(t)$, $z_i(t)$, $t_i(t)$), its momentum components ($p_{x,i}(t)$, $p_{y,i}(t)$, $p_{z,i}(t)$), and its charge $q_i(t)$. Here, t represents laboratory clock time⁴, z is the longitudinal beam propagation axis and x and y are the transverse axes.

Due to the iterative nature of the simulator, this evolution is computed at discrete time steps designated as $t^{(i)}$. Let $f^{(i)}$ denote the function describing the particle distribution evolution under the electromagnetic environment at time step $t^{(i)}$. This yields the following relation:

$$\mathcal{D}(t^{(i)}, \mathbf{c}) = f^{(i)}(\mathcal{D}(t^{(i-1)}, \mathbf{c}), \mathbf{c}) \quad (3.2)$$

We extract key beam characteristics from these particle distributions such as barycenter, size, divergence, energy, energy spread, and emittance. These characteristics, which are described in detail in Chapter 1.1, provide critical insights into the beam's properties. In our work, we concentrate on 14 standard observables that are most relevant for understanding the beam behavior in the ThomX's Linac.⁵ Let $u(t, \mathbf{c})$ be the function representing the evolution of these observables for a given control parameter $\mathbf{c}$, and let h be the function extracting these characteristics from the beam

$$u(t, \mathbf{c}) = h(\mathcal{D}(t, \mathbf{c})) \quad (3.3)$$

In particle accelerator simulations, it is often essential to characterize the beam at particular longitudinal positions along the accelerator, denoted by $(z_1, \dots, z_N)$. Each position z_i is reached by the beam at a time $t_i(\mathbf{c})$. These time instants $(t_1(\mathbf{c}), \dots, t_N(\mathbf{c}))$ correspond to discrete timesteps in the simulation: $(t^{(i_1(\mathbf{c}))}, \dots, t^{(i_N(\mathbf{c}))})$, where $i_j(\mathbf{c})$ denotes the index of the timestep at which

³As some particles may be lost during beam propagation, the particle set evolves over time, $\mathcal{S}(t + \Delta t, \mathbf{c}) \subset \mathcal{S}(t, \mathbf{c})$ for all $t \in \mathbb{R}_+$ and $\Delta t \in \mathbb{R}_+$.

⁴Note that the variable t represents the laboratory clock time, which is the global reference time for the entire system. In contrast, the variable $t_i(t)$ represents the individual time coordinate of each macro-particle, which may differ slightly due to relativistic effects and individual particle trajectories within the accelerator.

⁵These observables are x_{avr} , x_{std} , x'_{std} , $\epsilon_{x,norm}$, y_{avr} , y_{std} , y'_{std} , $\epsilon_{y,norm}$, $E_{kin,avr}$, z_{std} , $E_{kin,std}$, $\epsilon_{z,norm}$, Q , $|\mathcal{S}|$. Their formal definitions are provided in Section 1.1.3.

the beam arrives at z_j .⁶

These positions divide the accelerator into sequential segments, $[0, z_1]$, $[z_1, z_2]$, $\dots$, $[z_{N-1}, z_N]$. In light of the fact that certain control parameters exert an influence that is confined to specific localised segments of the accelerator, it is possible to partition the control parameters accordingly. In this context, the term "control parameters" refers to the subset of parameters that affect the beam propagation within a given segment $[z_{i-1}, z_i]$, which is represented by the vector variable $\mathbf{c}_i \in \mathbb{R}^{p_i}$.⁷

To represent the propagation of the particle distribution between these positions, we compose the discrete functions $f^{(i)}$ previously defined. For each segment $[z_{i-1}, z_i]$, we define a composite function f_i , representing the transformation of the particle distribution as it moves from z_{i-1} to z_i :

$$f_i = f^{(i_i(\mathbf{c}))} \circ f^{(i_i(\mathbf{c})-1)} \circ \dots \circ f^{(i_{i-1}(\mathbf{c})+1)}. \quad (3.4)$$

Thus, the distribution at position z_i is given by:

$$\mathcal{D}_i(\mathbf{c}) = \mathcal{D}(t_i(\mathbf{c}), \mathbf{c}) = f_i(\mathcal{D}_{i-1}(\mathbf{c}), \mathbf{c}_i) \quad (3.5)$$

From these distributions, we can extract the observable vectors which provide key characteristics of the beam at each specified position along the accelerator:

$$\mathcal{O}_i(\mathbf{c}) = \mathbf{u}(t_i(\mathbf{c}), \mathbf{c}) = h(\mathcal{D}_i(\mathbf{c})) \quad (3.6)$$

This entire data generation process is summarized in Figure 3.1.

3.1.2 . Macro-Particle Tradeoff

As described in Chapter 1.2, the accuracy of the simulation process heavily depends on the number of macro-particles used to describe a beam. At the same time, the more macro-particles are used in simulation, the more computational time is required for each simulation.

To determine the ideal ratio computing time/number of macro-particles while maximizing the accuracy, we perform various simulations with the same set of parameters by increasing the number of macro-particles. We refer to the error due to the averaging effect of a low number of macro-particles as the *intrinsic statistical error*. We are particularly interested in the intrinsic statistical error of energy spread and emittance due to the number of macroparticles,

⁶As the evolution of the beam in the accelerator depends on the control parameters $\mathbf{c}$, the time $t_i(\mathbf{c})$ at which the beam reaches the position z_i is a function of the control parameters. Consequently, the timestep indices $i_j(\mathbf{c})$ are also functions of these control parameters.

⁷For instance, the parameters controlling the accelerating section cannot affect the beam observed before this section. Conversely, once the beam has fully passed through the section, its control parameters no longer affect subsequent beam observations.

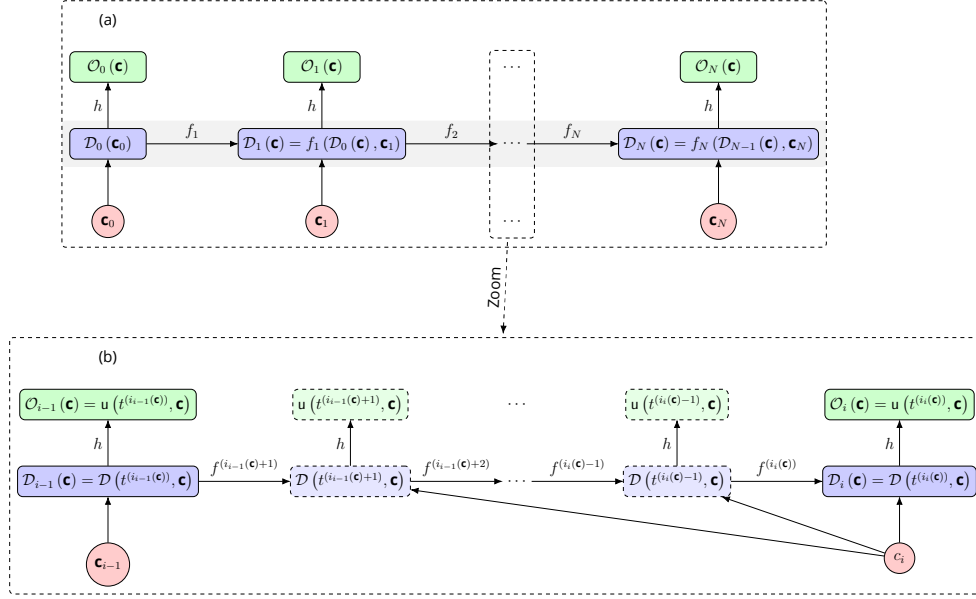


Figure 3.1: Generation of the observables at the location of interest. (a) describes the data generation from the observable positions' point of view. (b) is a zoomed-in version detailing all the steps of the simulation process at all times.

as they are two of the most crucial quantities for the proper operation of the accelerator.

To study a wide range of the number of macro-particles, we conducted 1000 runs with varying numbers of macro-particles, following a log-uniform distribution between 100 and 100000 macro-particles. The results are shown in Figure 3.2. As expected, the spread of the output values is more important for a few macro-particles. A saturation occurs around 10000 macro-particles, a value chosen for the study of this work. However, the spread remains and defines the intrinsic error of the random simulations. For 10000 macro-particles, the intrinsic error of the energy spread is 0.5keV rms, and for the emittance, 0.01 π mm mrad.

3.1.3 .Control Parameter Sampling

To construct our surrogate model, we need to generate a database representing the potential parameters that could be encountered in the real-world experiment. To do so, it is first necessary to distinguish between three types of parameters.

- **Mechanical Parameters.** These parameters are fixed during the construction of the accelerator and their values are known with a certain degree of precision. The database reflects this precision by sampling these parameters from a Gaussian distribution centered around the

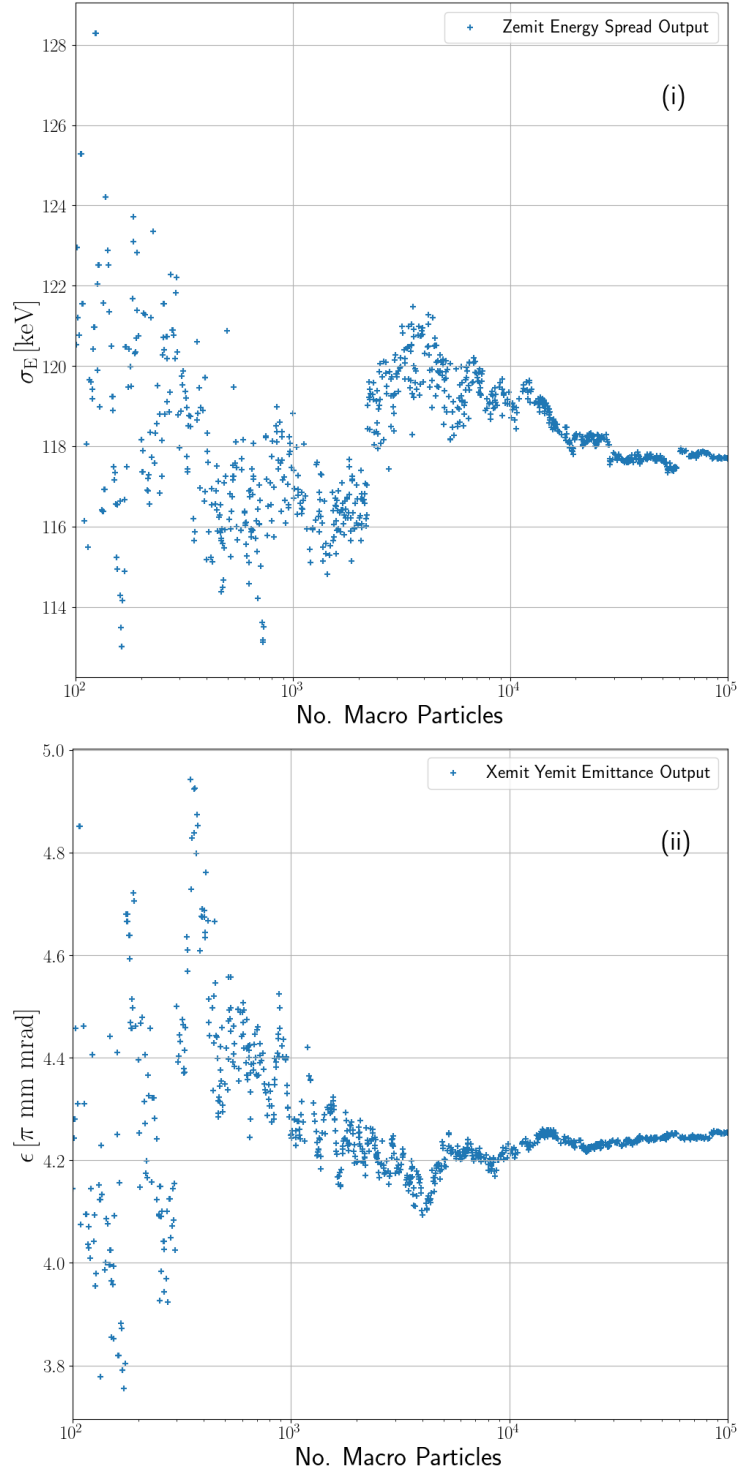


Figure 3.2: (i) Energy spread at the output of the accelerating section for 1000 runs with a randomly generated number of macro-particles following a log-uniform distribution. (ii) Transverse emittance at the output of the accelerating section for 1000 runs with a randomly generated number of macro-particles following a log-uniform distribution.

known value, with the standard deviation representing their uncertainty. The Gaussians are truncated at two standard deviations above and below to avoid unrealistic simulations.

- **Physical Quantities.** These parameters represent physical phenomena occurring inside the accelerator, primarily due to the materials used in its construction or the type of laser employed for photo-emission. We have prior knowledge of these quantities, mostly characterized by Gaussian distributions.
- **Tunable Parameters.** These parameters are subject to control by the operator of the accelerator. We have identified the range within which each tunable parameter can be adjusted. They are uniformly sampled within these intervals in the database.

We denote by $\mathcal{N}(\mu, \sigma, c_{\sigma,l}, c_{\sigma,r})$ the gaussian distribution truncated within $[\mu - c_{\sigma,l}\sigma, \mu + c_{\sigma,r}\sigma]$ and by $\mathcal{U}(a, b)$ the uniform distribution over the interval $[a, b]$. All the following parameters are sampled independently according to the mentioned distribution in Table 3.1.

Category	Variable	Distribution
Laser	x and y position on cathode	$\mathcal{N}(0 \text{ mm}, 0.24 \text{ mm}, 2, 2)$
	x and y size on cathode	$\mathcal{N}(0.52 \text{ mm}, 0.01 \text{ mm}, 2, 2)$
	x and y truncation by iris	$\mathcal{U}(1.0, 5.0)$
	x and y initial emittance	$\mathcal{N}(0.64\pi \text{ mm} \cdot \text{mrad}, 0.02\pi \text{ mm} \cdot \text{mrad}, 2, 2)$
	Temporal dispersion	$\mathcal{N}(0.003 \text{ ns}, 0.0001 \text{ ns}, 2, 2)$
Cathode	x, y initial angle	$\mathcal{N}(0 \text{ rad}, 0.03 \text{ rad}, 5, 5)$
	z initial offset	$\mathcal{N}(0 \text{ mm}, 0.1 \text{ mm}, 2, 2)$
	Initial time offset	$\mathcal{N}(0 \text{ ns}, 0.001 \text{ ns}, 2, 2)$
	Initial charge	$\mathcal{N}(0.92 \text{ nC}, 0.05 \text{ nC}, 2, 2)^8$
	Schottky effect value	$\mathcal{N}(0.0001 \text{ nC} \cdot \text{m/MV}, 0.000025 \text{ nC} \cdot \text{m/MV}, 4, 4)$
	Square root Schottky effect	$\mathcal{N}(0 \text{ nC} \cdot (\text{m/MV})^{1/2}, 0.0005 \text{ nC} \cdot (\text{m/MV})^{1/2}, 0, 2)$
	Emission delay time	$\mathcal{N}(0 \text{ ns}, 0.0001 \text{ ns}, 0, 2)$
Gun	Maximum field	$\mathcal{N}(80.0 \text{ MV/m}, 1.6 \text{ MV/m}, 3, 0.5)$
	Phase	$\mathcal{U}(-40^\circ, 60^\circ)^9$
	x, y translation misalignments	$\mathcal{N}(0 \text{ mm}, 0.1 \text{ mm}, 2, 2)$
	x, y, z rotation misalignments	$\mathcal{N}(0 \text{ mrad}, 0.04 \text{ mrad}, 2, 2)$
Solenoid	Maximum field	$\mathcal{U}(0.25088 \text{ T}, 0.26112 \text{ T})$
	x, y, z translation misalignments	$\mathcal{N}(0 \text{ mm}, 0.1 \text{ mm}, 2, 2)$
	x, y rotation misalignments	$\mathcal{N}(0 \text{ mrad}, 0.4 \text{ mrad}, 5, 5)$
Accelerating section	Maximum field	$\mathcal{N}(14.0 \text{ MV/m}, 0.28 \text{ MV/m}, 3, 0.5)$
	Phase	$\mathcal{U}(-75^\circ, 75^\circ)$
	x, y translation misalignments	$\mathcal{N}(0 \text{ mm}, 0.1 \text{ mm}, 2, 2)$
	x, y, z rotation misalignments	$\mathcal{N}(0 \text{ mrad}, 0.04 \text{ mrad}, 2, 2)$

Table 3.1: Summary of control parameter distributions for each accelerator component. z is the longitudinal direction axis (direction of propagation of the particles in the accelerator) while x and y are the transverse directions.

These 37 parameters encompass a wide array of potential configurations,

capturing diverse experimental setups and conditions specific to the accelerator. We deliberately select a broad set of parameters with extensive sampling ranges, distinguishing our approach from traditional sampling methods commonly used in particle accelerator modeling. This wide-ranging sampling strategy is intended to cover a broader spectrum of configurations in simulations, helping to bridge the inevitable gap between the simulation model and the real accelerator, thereby enhancing the model’s applicability to real-world conditions.

We generate $n = 30000$ sets of control parameters, which are divided as follows: 21000 samples for the training set, 6000 for the validation set, and 3000 for the test set.

3.1.4 .Simulation Results Overview

In this subsection, we summarize the key results from the simulations. After introducing the notation for the resulting database, we first address beam survival to provide an overview of the diversity of the database, including beams that do not reach the end of the Linac. This analysis helps illustrate the range of outcomes and the number of beams that retain sufficient macro-particles. We then present an overview of the resulting observables, which capture the evolution of key beam characteristics throughout the accelerator.

Let $\mathbf{c}^{(i)}$ denote the i -th sample of our database, with $i \in \llbracket 1, n \rrbracket$. We store the particle distributions and trajectories for each sample at specific longitudinal positions along the accelerator. However, due to disk space usage limitations, we cannot store these particle distributions and trajectories for every timestep of the particle accelerator simulator.

The particle distributions $\mathcal{D}(z_{\mathcal{D}}^{(k)}, \mathbf{c}^{(i)})$ are recorded at $N_{\mathcal{D}} = 25$ distinct longitudinal positions, $z_{\mathcal{D}}^{(k)}$, where $k \in \llbracket 1, N_{\mathcal{D}} \rrbracket$. These positions include the six diagnostic stations installed along ThomX’s Linac, in addition to positions spaced uniformly at $\frac{L}{20}$ meters, where L represents the total length of the Linac. Notably, the final diagnostic station coincides with the last position of the uniform partitioning, resulting in 25 distinct storage points.

The trajectories $\mathbf{u}(z_{\mathbf{u}}^{(l)}, \mathbf{c}^{(i)})$ are stored at a higher density, with $N_{\mathbf{u}} = 4004$ longitudinal positions $z_{\mathbf{u}}^{(l)}$ where $l \in \llbracket 1, N_{\mathbf{u}} \rrbracket$, corresponding to uniformly spaced positions at $\frac{L}{4000}$ meters, plus the positions corresponding to the diagnostic stations, yielding a total of 4004 storage points. Along with the trajectories, we also store a flag indicating whether the beam is still alive at each longitudinal position—i.e., whether at least one macro-particle is still present in the beam. This flag is denoted as $\mathbb{1}(z_{\mathbf{u}}^{(l)}, \mathbf{c}^{(i)})$.

Finally, our database is the following set:

$$\left\{ \mathbf{c}^{(i)}, \left(\mathcal{D}_k \left(\mathbf{c}^{(i)} \right), \mathcal{O}_k \left(\mathbf{c}^{(i)} \right) \right)_{k \in \llbracket 1, N_{\mathcal{D}} \rrbracket}, \left(\mathbf{u} \left(z^{(l)}, \mathbf{c}^{(i)} \right), \mathbb{1} \left(z_{\mathbf{u}}^{(l)}, \mathbf{c}^{(i)} \right) \right)_{l \in \llbracket 1, N_{\mathbf{u}} \rrbracket} \right\}_{i \in \llbracket 1, n \rrbracket} \quad (3.7)$$

3.1.4.1 . Beam Survival

Along with the tracking of particles in the simulator, macro-particles may be lost due to various factors, such as collision with the aperture of the accelerator or rejection at the cathode due to the space charge effect. If all particles in a beam are lost, the entire beam is considered lost.

We first examine the proportion of beam survival in Table 3.2. It is observed that half of the beams are rejected at the beginning of the tracking, which is consistent with the high space charge effects at the cathode. Most of the beams that survive the first section are observed to survive until they reach the end of the accelerator.

Generated Beams	30000
Living beams at first screen ($z = 0.46965\text{m.}$)	13421 (44.74%)
Living beams at last screen ($z = 9.393\text{m.}$)	13342 (44.47%)

Table 3.2: Number of beams in the database that are still tracked according to the position in the accelerator. Database contains 30000 beams.

Figure 3.3 displays the number of macro-particles in the remaining beams according to the position in the accelerator. We observe that 30% of the beams surviving the electron gun has experienced a significant loss of particles in the electron gun at the start of the accelerator with less than 8000 particles remaining. This percentage of particle loss increases in the accelerating section (which spans from 1.527m. to 6.325m.). At the end of the tracking, 30% of the beams retain more than 8000 particles.

3.1.4.2 . Resulting observables

Once we understand beam survival, we can analyze the evolution of key beam characteristics, known as observables, to gain further insights into the quality and behavior of the beam as it propagates through the accelerator. The database generation leads to the evolution of 14 observables that the surrogate model should be able to predict accurately. Figure 3.4 shows their evolution through the accelerator.

The comparison between the different observables reveals that each observable operates on a different scale (for example, see the barycenter values (x_{avr} and y_{avr}) compared to the energy dispersion values ($E_{\text{kin, std}}$)). These

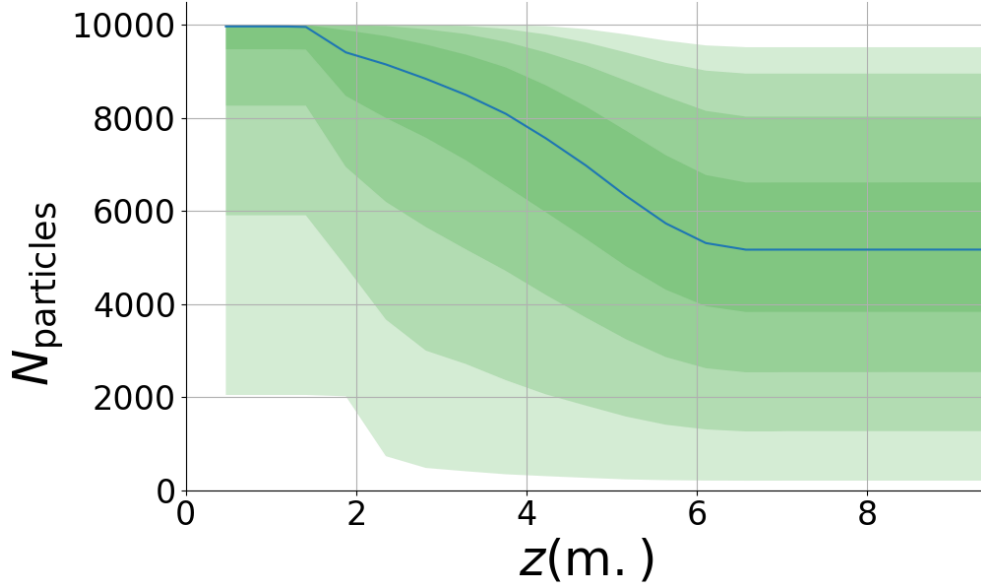


Figure 3.3: Number of macro-particles in beams that are still tracked according to the position in the accelerator. The blue line indicates the median, and the green zones indicate the decile values. The first and last deciles are not shown.

scales also vary between the start and the end of the accelerator (as seen with the longitudinal emittance ($\epsilon_{z,norm}$)). Due to these differences, we have decided to apply a specific scaling for each observable and each section of the accelerator during the training of the surrogate model.

The shape of the observable distribution exhibits significant variation not only across the different observables but also between the start and end of the accelerator. As shown in Figure 3.5, the histograms of the observables' distributions after the first section and at the accelerator's end demonstrate this transformation. Despite the fact that the initial control parameters are sampled from uniform and Gaussian distributions, the evolution of the beam through the accelerator leads to observable distributions that diverge into distinctly different shapes. This indicates that the accelerator dynamics and beam interaction with the environment substantially alter the distribution patterns of key observables.

3.2 .Performance Indicators

To evaluate the performance of a model for predicting the characteristics of a beam, several metrics can be employed to assess its accuracy. Given the diversity of the problems the model aims to address, it is essential to use multiple metrics to evaluate its different aspects. These metrics should capture the model's ability to generalize across different observables and sections of

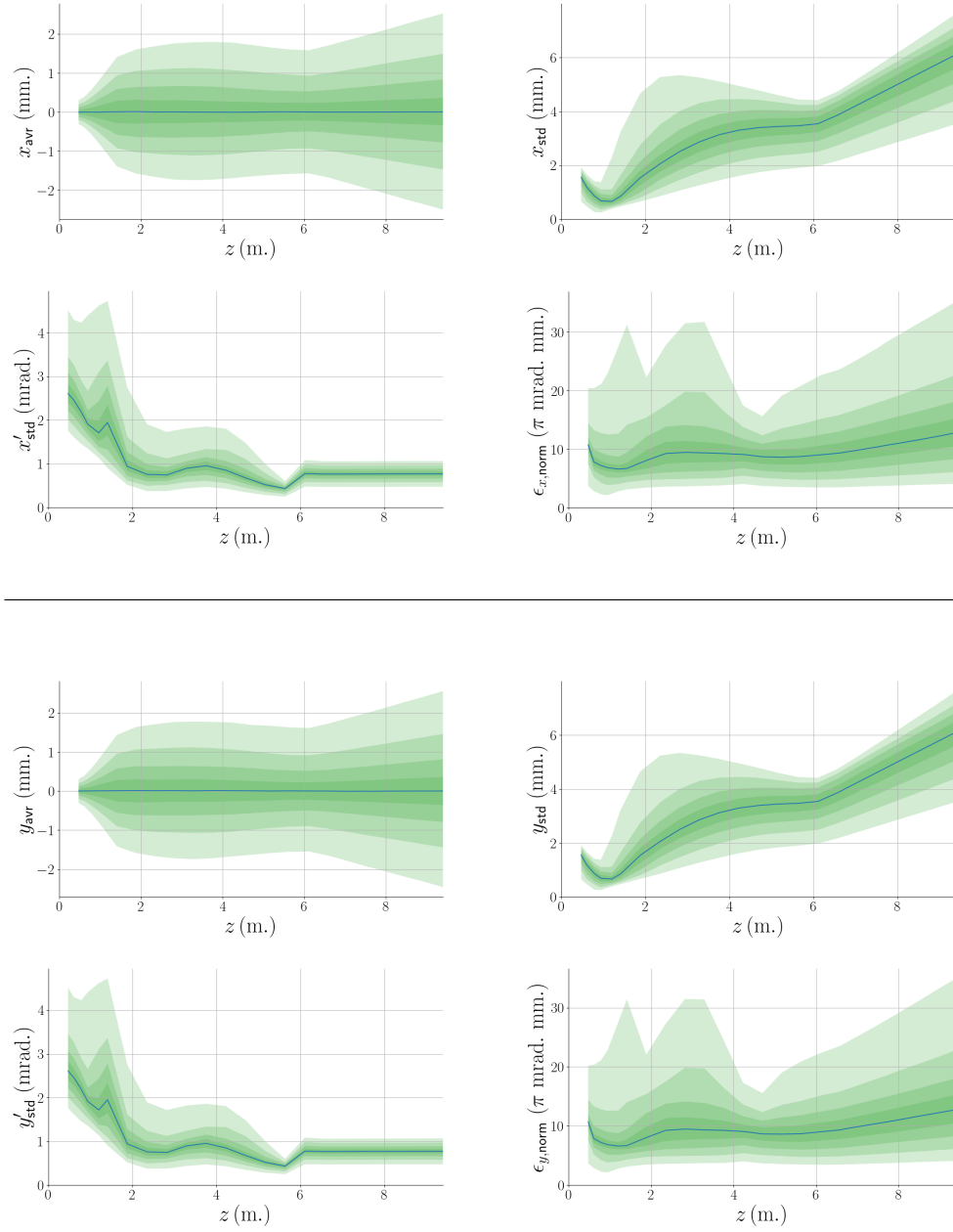


Figure 3.4: Evolution of the 14 observables of the beam through the accelerator. The blue line indicates the median and the green zones indicate the decile values. The first and last deciles are not shown.

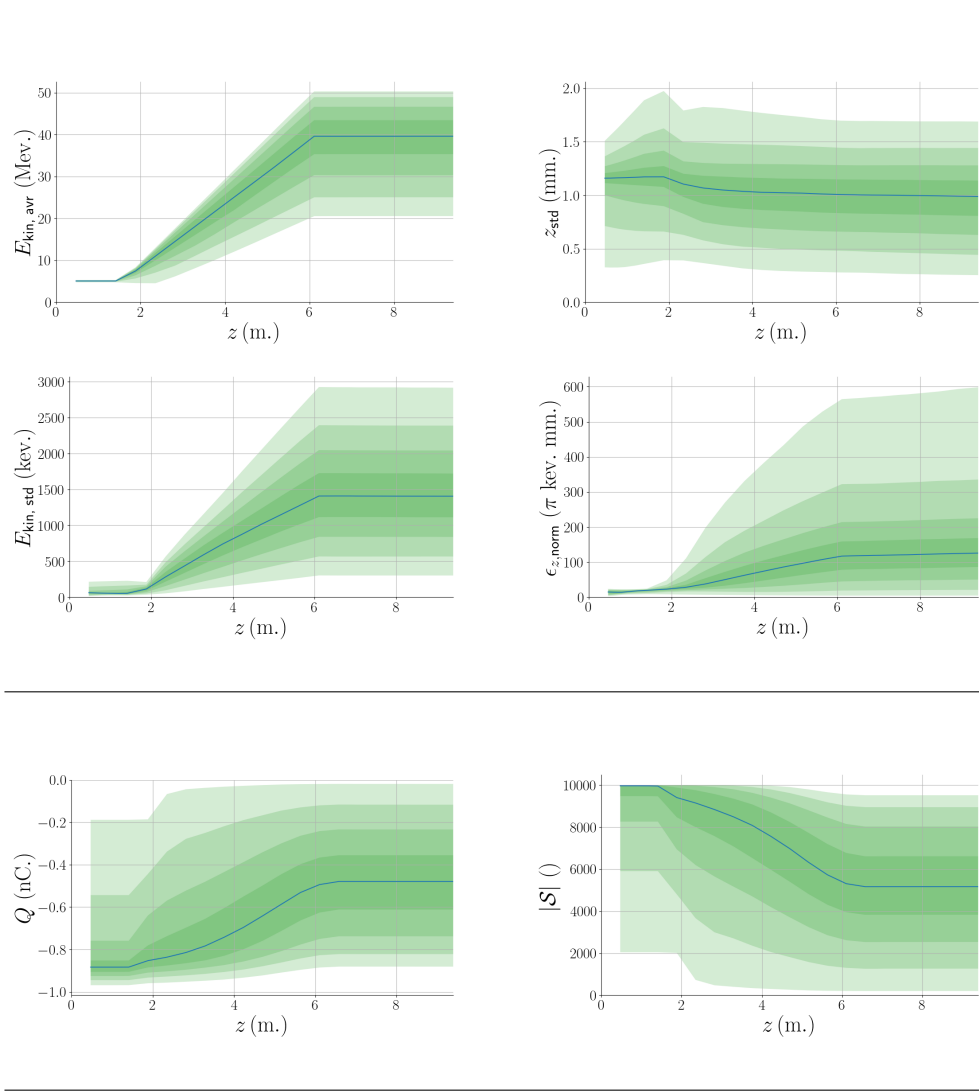


Figure 3.4: Evolution of the 14 observables of the beam through the accelerator. The blue line indicates the median and the green zones indicate the decile values. The first and last deciles are not shown. (Continued)

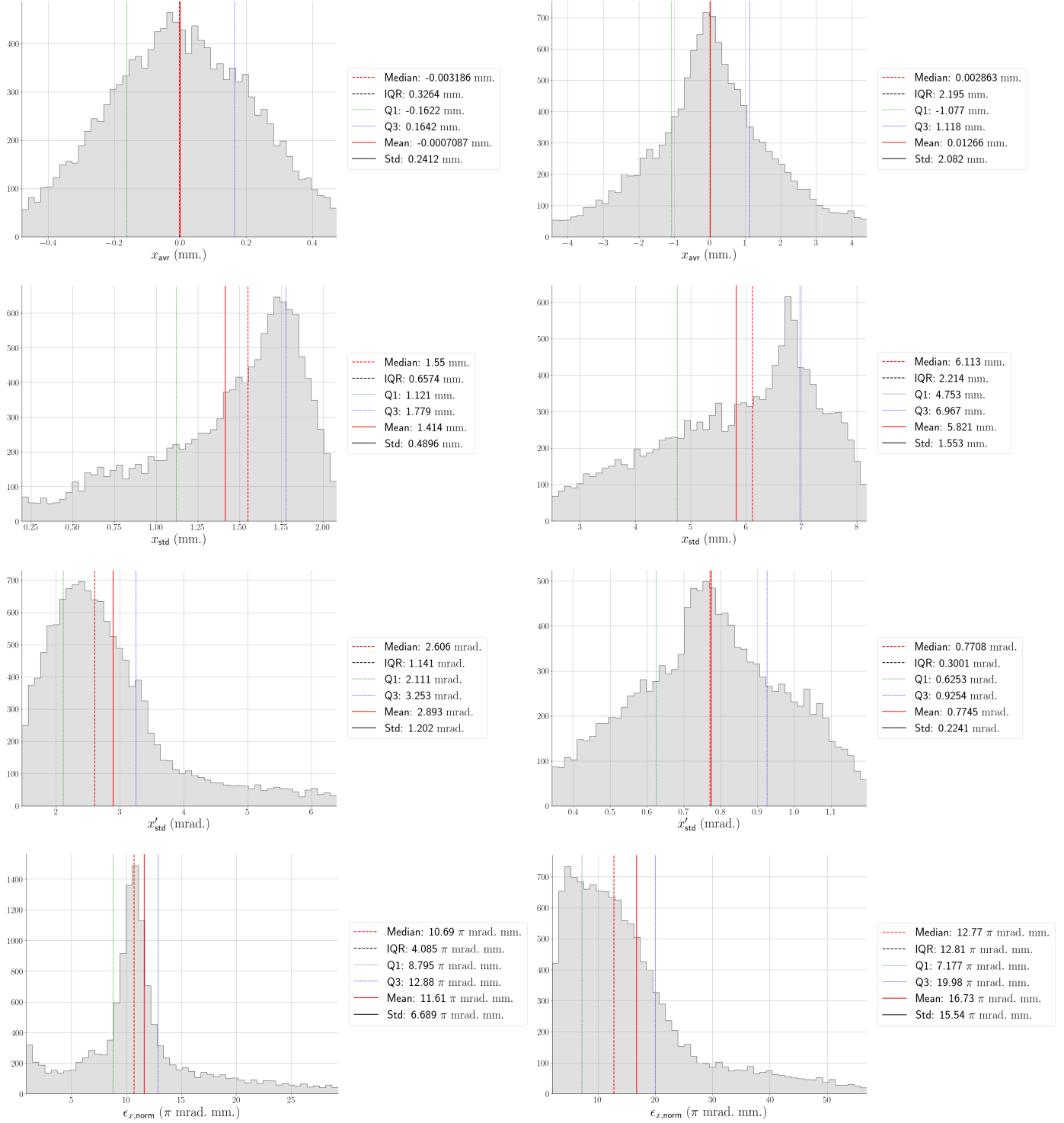


Figure 3.5: Histograms of the observables at the start and end of the accelerator. To enhance readability, only 95% of the data is displayed, with the most extreme 2.5% of values on each side removed from each histogram.

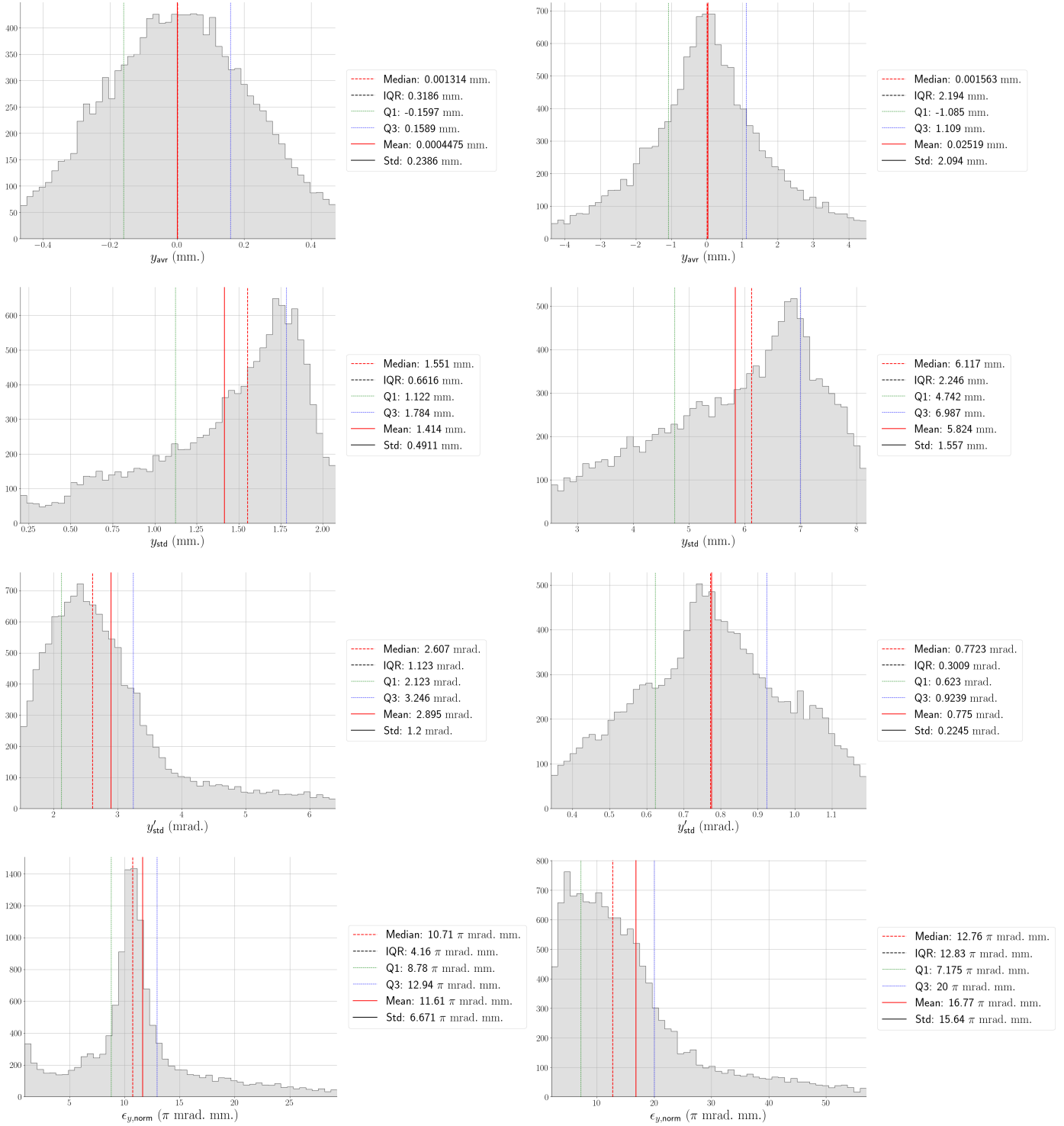


Figure 3.5: Histograms of the observables at the start and end of the accelerator. To enhance readability, only 95% of the data is displayed, with the most extreme 2.5% of values on each side removed from each histogram. (Continued)

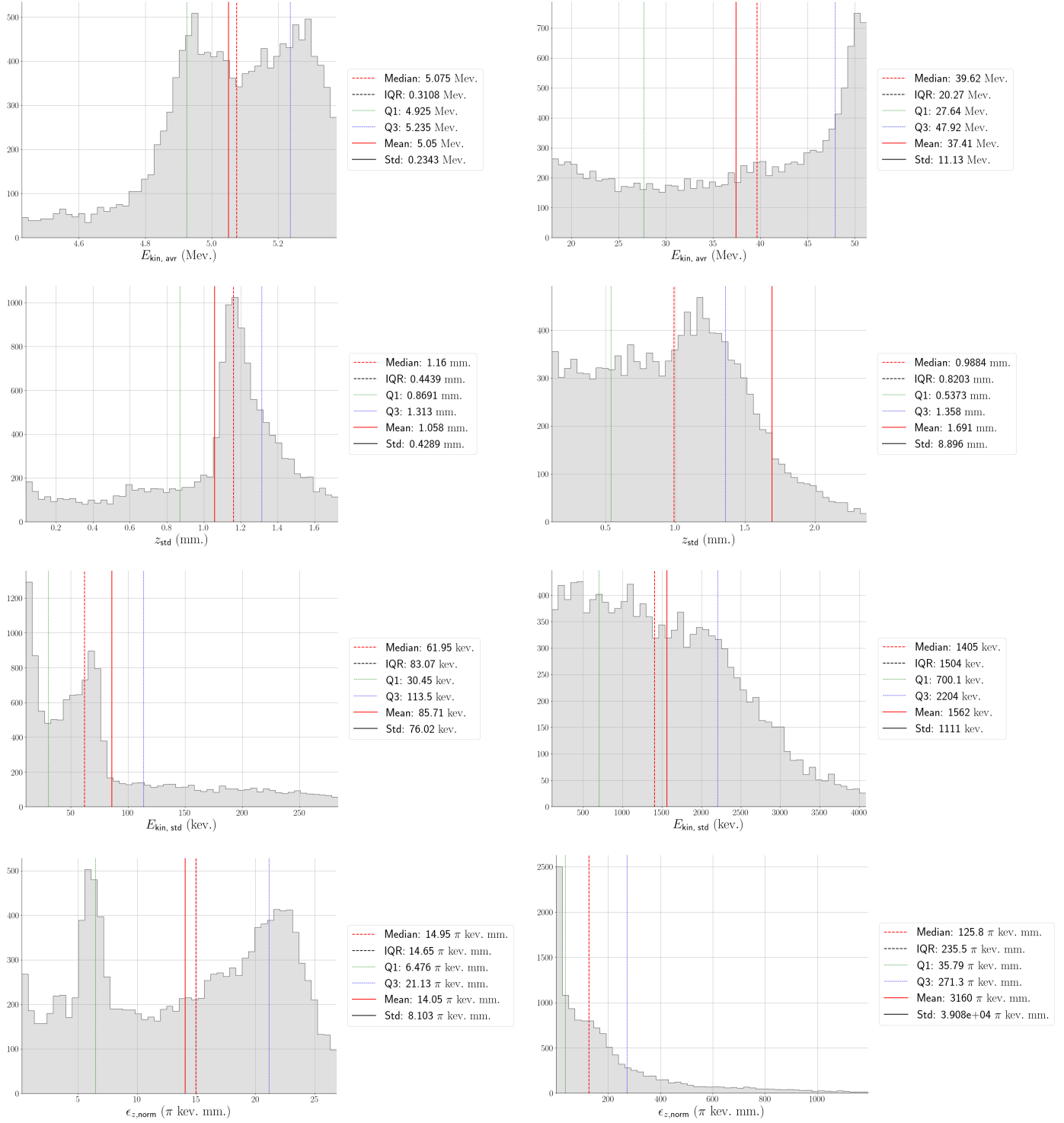


Figure 3.5: Histograms of the observables at the start and end of the accelerator. To enhance readability, only 95% of the data is displayed, with the most extreme 2.5% of values on each side removed from each histogram. (Continued)

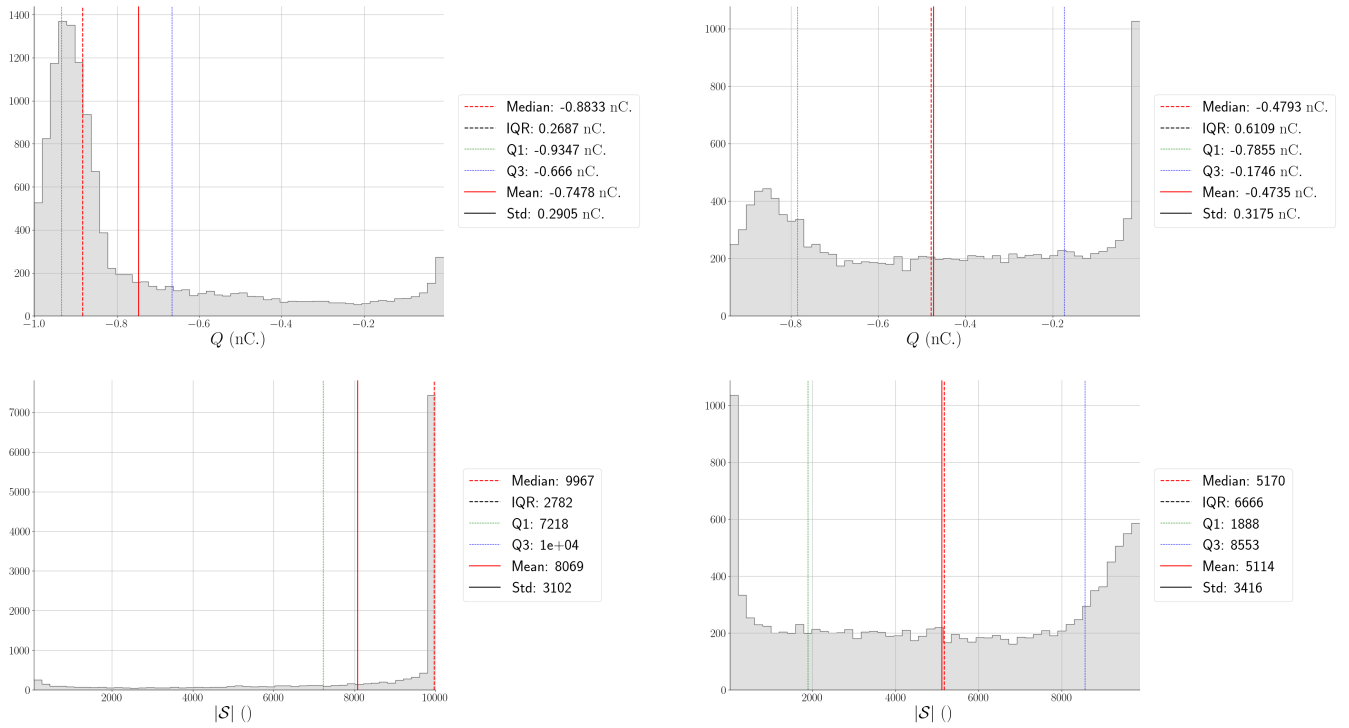


Figure 3.5: Histograms of the observables at the start and end of the accelerator. To enhance readability, only 95% of the data is displayed, with the most extreme 2.5% of values on each side removed from each histogram. (Continued)

the accelerator. Furthermore, we can exhibit various criterion thresholds according to the task for which the model will be used in practice.

Observable Metrics The main purpose of the model is to accurately predict the various observables composing the dataset. This problem is formulated as a typical regression problem, where standard performance metrics such as Mean Absolute Error (MAE) and the R^2 score are used to assess the model's accuracy. It is important to highlight that each model provides predictions for all observables, even in cases where the beam is lost during tracking. To account for this phenomenon, the evaluation metrics are computed considering only the beams that successfully reach the screen, while predictions for lost beams are excluded from the calculation. This approach ensures that the performance assessment focuses on valid outcomes and avoids skewing the results with data from incomplete trajectories.

In practice, we define the R^2 score at each potential position z along the accelerator:

$$R^2(z) = 1 - \frac{\sum_{i=1}^n \mathbb{1}(z, \mathbf{c}^{(i)}) (\hat{u}(z, \mathbf{c}^{(i)}) - u(z, \mathbf{c}^{(i)}))^2}{\sum_{i=1}^n \mathbb{1}(z, \mathbf{c}^{(i)}) (\bar{u}(z) - u(z, \mathbf{c}^{(i)}))^2}, \quad (3.8)$$

where $\hat{u}(z, \mathbf{c}^{(i)})$ is the prediction made by a model for a control parameter $\mathbf{c}^{(i)}$, n is the number of samples in the dataset, $\mathbb{1}(z, \mathbf{c}^{(i)})$ is a flag indicating if the beam is alive at the z position and $\bar{u}(z)$ is the average of the alive trajectories at position z . In practice, $\hat{u}(z, \cdot)$, $u(z, \cdot)$ and $\bar{u}$ being vectors, multiplication and division are implicitly made term by term. The resulting $R^2(z)$ is a vector of size equal to the number of observables.

Similarly, we define:

$$\text{MAE}(z) = \frac{\sum_{i=1}^n \mathbb{1}(z, \mathbf{c}^{(i)}) (\hat{u}(z, \mathbf{c}^{(i)}) - u(z, \mathbf{c}^{(i)}))}{\sum_{i=1}^n \mathbb{1}(z, \mathbf{c}^{(i)})}, \quad (3.9)$$

The MAE is comparatively less influenced by large individual errors than the R^2 score, as it considers the absolute residuals rather than their squared values. However, unlike the R^2 score, the MAE is not scaled and has to be compared to a reference to evaluate its quality. The first reference is the interquartile range (IQR) of the database, which provides a measure of the spread of the data. The values for these IQRs at the diagnostic stations are presented in Table 3.4. Additionally, the MAE can also be compared to the accuracy of the diagnostic stations installed in ThomX. The values of these accuracies can be found in Table 3.3. These comparisons help contextualize

the model's performance and offer insight into its practical reliability relative to the system's inherent measurement precision.

Table 3.3: Accuracy of the diagnostic stations installed on ThomX.

Diagnostic Station	Observable	Accuracy
BPM	x_{avr} (mm.)	0.1
	y_{avr} (mm.)	0.1
ICT	Q (nC.)	0.01
SST	x_{avr} (mm.)	0.1
	y_{avr} (mm.)	0.1
	x_{std} (mm.)	0.1
	y_{std} (mm.)	0.1

Particle Metrics. Similarly to the previously defined metrics, we can define a R^2 score at the particle level for models predicting the complete set of particles denoted as $R_{\mathcal{D}}^2$. In this case, rather than using observables, the metric is based on the particle coordinates. Formally, we have:

$$R_{\mathcal{D}}^2(z) = 1 - \frac{\sum_{i=1}^n \sum_{j \in \mathcal{S}(z, \mathbf{c}^{(i)})} \left(\hat{\zeta}_j(z, \mathbf{c}^{(i)}) - \zeta_j(z, \mathbf{c}^{(i)}) \right)^2}{\sum_{i=1}^n \sum_{j \in \mathcal{S}(z, \mathbf{c}^{(i)})} \left(\bar{\zeta}(z) - \zeta_j(z, \mathbf{c}^{(i)}) \right)^2}, \quad (3.10)$$

where $\hat{\zeta}_j(z, \mathbf{c}^{(i)})$ is the prediction made by a model for the j -th particle for a control parameter $\mathbf{c}^{(i)}$, $\mathcal{S}(z, \mathbf{c}^{(i)})$ is the set of indices of living particles for control parameter $\mathbf{c}^{(i)}$ and $\bar{\zeta}(z)$ is the average of the living particles at position z .

Table 3.4: IQR values of the different observables at the 6 diagnostic stations of ThomX.

		BPM ($z = 0.6055\text{m.}$)	ICT ($z = 0.7845\text{m.}$)	SST ($z = 1.201\text{m.}$)	ICT ($z = 6.868\text{m.}$)	BPM ($z = 7.668\text{m.}$)	SST ($z = 9.393\text{m.}$)
Hor. Trans. Dim.	x_{avr} (mm.)	0.406	0.606	1.191	1.547	1.753	2.195
	x_{std} (mm.)	0.726	0.665	0.502	1.583	1.789	2.214
	x'_{std} (mrad.)	1.096	0.926	1.403	0.297	0.299	0.300
	$\epsilon_{x,\text{norm}}$ (π mm. mrad.)	5.361	5.744	5.311	9.054	10.235	12.807
Vert. Trans. Dim.	y_{avr} (mm.)	0.402	0.596	1.168	1.536	1.742	2.194
	y_{std} (mm.)	0.732	0.667	0.505	1.596	1.812	2.246
	y'_{std} (mrad.)	1.082	0.919	1.387	0.301	0.301	0.301
	$\epsilon_{y,\text{norm}}$ (π mm. mrad.)	5.390	5.748	5.499	9.079	10.294	12.829
Long. Dim.	$E_{\text{kin,avr}}$ (MeV.)	0.311	0.311	0.309	20.274	20.270	20.272
	z_{std} (mm.)	0.480	0.525	0.625	0.792	0.802	0.820
	$E_{\text{kin,std}}$ (keV.)	85.056	85.114	92.995	1506.590	1506.166	1504.195
	$\epsilon_{z,\text{norm}}$ (π keV. mm.)	12.805	10.777	8.719	227.759	229.864	235.497
Other	Q (nC.)	0.269	0.269	0.269	0.611	0.611	0.611
	$ S $	2782	2782	2782	6666.75	6666	6665.5

3.3 .Conclusion

In this chapter, we outlined the experimental framework for modeling ThomX's Linac using surrogate models, including the generated database and evaluation metrics.

The database, generated by through extensive ASTRA simulations, captures diverse accelerator conditions by randomizing control parameters and applying systematic sampling. This approach enhances real-world applicability while balancing simulation accuracy and computational feasibility. Making the dataset publicly available supports future research and benchmarking in particle accelerator modeling.

We defined performance metrics for both beam observables and particle distributions. Metrics such as Mean Absolute Error and R^2 scores assess model accuracy across different longitudinal positions along the accelerator, contextualized by diagnostic precision and data spread. Particle-level metrics further assess the ability of the model to reproduce macroparticle distributions, which are critical for understanding beam dynamics.

The results in this chapter provide the foundation for optimizing and evaluating surrogate models. In the next chapters, we will explore surrogate model architectures and training, using the generated database and metrics to demonstrate the ability of surrogate models to replicate Linac behavior.

4 -Particle Cloud Modeling: Explicit Sequential Supervised Learning

Contents

4.1 Particle Cloud Modeling	62
4.1.1 Modeling a Particle Cloud	62
4.1.2 Computing a Loss for a Particle Cloud Prediction	66
4.2 Beam Propagation Modeling in a Sequence of Accelerator Segments	68
4.2.1 Modeling	68
4.2.2 A Multi-Objective Problem	69
4.2.3 A Multi-Task Learning Solution	70
4.3 Validation	74
4.3.1 Goals of Experiments	74
4.3.2 First Segment Modeling	75
4.3.3 Multi-segment modeling	85

This chapter presents our method, denoted as `LINACNET`, for modeling the propagation of a particle cloud through a linear accelerator. Unlike existing surrogate models developed for particle accelerators, `LINACNET` is distinct in its focus on modeling the entire particle beam—predicting the coordinates of each macroparticle within the simulation—rather than solely estimating beam observables. This approach has been adopted for two primary reasons.

First, a complete prediction of the beam facilitates interoperability of `LINACNET` with other surrogate models or particle accelerator simulators. Traditional surrogate models typically output a limited set of observables, which are insufficient for subsequent simulations that require a complete particle beam. This interoperability is particularly crucial for large accelerators composed of multiple subsystems, where the output beam of one segment serves as the input for the next.

Second, predicting the complete particle beam enhances the reliability and interpretability of the model. By providing physicists with increased granularity and detail, `LINACNET` allows for a more comprehensive analysis of the model results, thereby supporting a deeper understanding of the underlying particle dynamics.

In Section 4.1, we present the methodology by which `LINACNET` models the dynamics of a particle cloud within a segment of the accelerator. Our approach

draws inspiration from 3D computer vision frameworks, such as PointNet [11], which are particularly effective for capturing the intricate interactions within particle clouds.

Another key aspect of LINACNET is its capacity to generate predictions at multiple locations along the accelerator. Section 4.2 details how LINACNET accomplishes this by stacking multiple models, each corresponding to a distinct subpart of the accelerator. We illustrate how these models are interconnected to ensure accurate end-to-end predictions, while preserving the physical validity of each submodel in capturing the phenomena specific to its respective segment.

Finally, Section 4.3 provides an experimental validation of LINACNET. We demonstrate its capability in modeling the entire particle cloud, as well as discuss its limitations when applied to the complete accelerator system.

4.1 .Particle Cloud Modeling

This section presents the methodology of LINACNET for modeling a particle cloud, which represents a particle beam in its entirety rather than restricting the analysis to a limited set of observables. This modeling approach offers both significant advantages and challenges. Predicting the complete set of particle coordinates provides a comprehensive and detailed representation of the beam, enabling a more nuanced analysis than focusing solely on beam observables. However, this approach also necessitates addressing the complexities associated with modeling a high-dimensional set of particle coordinates. This section explores the methods for addressing this complexity.

4.1.1 .Modeling a Particle Cloud

A macro-particle ζ in a particle accelerator simulator is represented in eight-dimensional (8D) coordinates: three positional coordinates (x, y, z) three momentum coordinates (p_x, p_y, p_z) , intrinsic time t , and charge q . These dimensions capture the fundamental physical properties and dynamics of particles within the accelerator. In addition to these coordinates, we introduce a scalar value representing the probability that a macro-particle is not lost during beam tracking. This probability, denoted as ϕ , signifies the probability of a particle surviving through each segment.

A beam, therefore, is represented as a set of macro-particles $\mathcal{D} = \{\zeta_j, j \in \mathcal{S}\}$, where $\mathcal{D}$ is the particle cloud and $\mathcal{S}$ is the set of surviving particle indices. We denote by N_{part} the total number of initial macro-particles generated at the beginning of the simulation. Drawing inspiration from 3D computer vision, where the goal is to model 3D scenes composed of sets of points, we adapt these models to handle 8D particles. In 3D computer vision, techniques such as PointNet [11] are commonly used to process and analyze sets of points

efficiently. Similarly, we employ these techniques to manage the complexity of our particle cloud.

4.1.1.1 .Reinterpreting PointNet for Particle Cloud Modeling

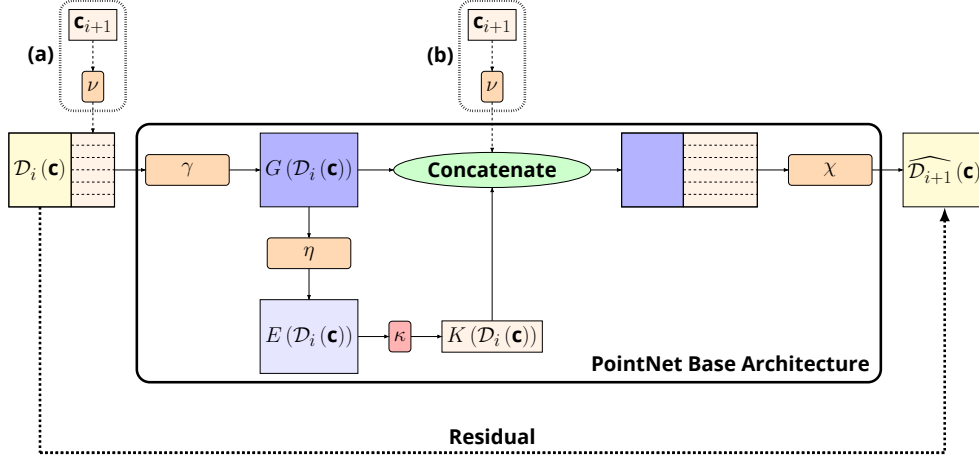


Figure 4.1: LINACNET Architecture. It processes the set of particles $\mathcal{D}_i(\mathbf{c})$ entering the i -th segment using the PointNet [11] architecture. It adapts it to incorporate control parameters. In LINACNET (a), the local control parameters $\mathbf{c}_{i+1}$ are provided as input of the particles processed by PointNet. In LINACNET (b), the local control parameters $\mathbf{c}_{i+1}$ are concatenated to the intermediate step of PointNet extracting the local features and the global descriptor of the beam.

Inspired by PointNet [11], we propose LINACNET for modeling particle clouds. Figure 4.1 illustrates the architecture of LINACNET, showing the adaptation of PointNet to process the particle cloud and incorporate control parameters.

The particle cloud entering the i -th segment of the accelerator is the set of particles:

$$\mathcal{D}_i(\mathbf{c}) = \{\zeta_{i,j}(\mathbf{c}), j \in \mathcal{S}_i(\mathbf{c})\}, \quad (4.1)$$

where $\mathcal{S}_i(\mathbf{c})$ represents the set of indices of living particle in the beam at z_i and $\zeta_{i,j}(\mathbf{c})$ is the vector of coordinates of the j -th particle.

LINACNET first processes each particle $\zeta_{i,j}$ individually using a multilayer perceptron (MLP) γ to extract features $\gamma(\zeta_{i,j}(\mathbf{c}))$, capturing the local properties of particles. The resulting set $G(\mathcal{D}_i(\mathbf{c}))$ is therefore an embedding of each individual particle composing the beam.

$$G(\mathcal{D}_i(\mathbf{c})) = \{\gamma(\zeta_{i,j}), j \in \mathcal{S}_i(\mathbf{c})\} \quad (4.2)$$

These features are then embedded in another space using another MLP η to extract new set of features $E(\mathcal{D}_i(\mathbf{c})) = \{\eta(\gamma(\zeta_{i,j}(\mathbf{c}))), j \in \mathcal{S}_i(\mathbf{c})\}$ that are aggregated using a symmetric function κ , such as max pooling, mean, variance,

or quantiles, to generate a global descriptor of the particle cloud. This global descriptor is the vector

$$K(\mathcal{D}_i(\mathbf{c})) = \kappa(\{\eta(\gamma(\zeta_{i,j}(\mathbf{c}))), j \in \mathcal{S}_i(\mathbf{c})\}) \quad (4.3)$$

This global descriptor is concatenated with the features of each particle to model the interactions between each particle and the rest of the beam. Finally, the coordinates of each particle are predicted using a final MLP χ , which takes as input both the global descriptor and the local features of the particle, as well as a residual connection to the input particle.

$$\widehat{\zeta}_{i+1,j}(\mathbf{c}) = \chi(\gamma(\zeta_{i,j}(\mathbf{c})), K(\mathcal{D}_i(\mathbf{c}))) + \zeta_{i,j}(\mathbf{c}) \quad (4.4)$$

This architecture allows us to capture the particle cloud's local and global properties efficiently.

4.1.1.2 .Adjusting for Control Parameters

Contrary to traditional vision models that only take a point cloud as input, here LINACNET should also incorporate the information of the control parameter $\mathbf{c}_i$ describing the state of the accelerator influencing the propagation of the beam between z_i and z_{i+1} . We explore two alternative architectures for integrating control parameters with LINACNET :

1. LINACNET .(a): Control parameters $\mathbf{c}_i$ are part of the input, directly influencing the feature extraction process:

$$G(\mathcal{D}_i(\mathbf{c})) = \{\gamma(\zeta_{i,j}(\mathbf{c})), \nu(\mathbf{c}_i)), j \in \mathcal{S}_i(\mathbf{c})\} \quad (4.5)$$

2. LINACNET .(b): Control parameters are concatenated to the output of the feature extraction process:

$$\widehat{\zeta}_{i,j}(\mathbf{c}) = \chi(\gamma(\zeta_{i,j}(\mathbf{c})), K(\mathcal{D}_i(\mathbf{c})), \nu(\mathbf{c}_i)) + \zeta_{i,j}(\mathbf{c}) \quad (4.6)$$

4.1.1.3 .Survival Probability

To model whether a particle reaches the end of a segment, we predict a probability $\phi_{i,j}(\mathbf{c})$ that particle $\zeta_{i,j}(\mathbf{c})$ survives. This probability is updated at the end of each segment.

We explore three alternatives to model this probability:

1. RAW: Initially, we propose predicting this probability as part of the regression process described above:

$$\phi_{i+1,j}(\mathbf{c}) = \chi_\phi(\gamma(\zeta_{i,j}(\mathbf{c})), K(\mathcal{D}_i(\mathbf{c}))), \quad (4.7)$$

where χ_ϕ is another head to the χ regression network described above.

2. MIN: For consistency, a particle deemed lost in a segment should not reappear in subsequent segments. Therefore, we compare this approach to a *minimum constraint* alternative that enforces the survival probability to be the minimum between the input and output probabilities:

$$\phi_{i+1,j}(\mathbf{c}) = \min(\chi_\phi(\gamma(\zeta_{i,j}(\mathbf{c})), K(\mathcal{D}_i(\mathbf{c}))), \phi_{i,j}(\mathbf{c})), \quad (4.8)$$

3. CONDITIONAL: Additionally, we propose modeling this probability using conditional probability. Instead of predicting the raw probability of the particle being alive at the end of the segment, we predict the probability of the particle being alive at the end of the segment, given it was alive at the entrance. Using the law of total probability:

$$\phi_{i+1,j}(\mathbf{c}) = \phi_{i,j}(\mathbf{c}) \cdot \chi_\phi(\gamma(\zeta_{i,j}(\mathbf{c})), K(\mathcal{D}_i(\mathbf{c}))), \quad (4.9)$$

4.1.1.4 .Aggregation

In PointNet [11], the aggregation function κ typically uses max, which learns the extremities of a distribution. However, we test various aggregation functions for particle clouds. These include mean, variance, and quantiles, which can better capture the statistical properties and distributions within the particle cloud.

We also consider their weighted counterparts, accounting for the importance of each particle in the particle-beam interactions. In a particle accelerator simulator, the weight of a macro-particle in the beam is its charge q . Therefore, we weight all aggregation operators with the charge of the macro-particles. Additionally, considering each particle's probability of being alive $\phi_{i,j}$, the final weight is the product of the charge and survival probability. Finally, the weight applied to the particle $\zeta_{i,j}(\mathbf{c})$ is:

$$w_{i,j}(\mathbf{c}) = q_{i,j}(\mathbf{c}) \phi_{i,j}(\mathbf{c}) \quad (4.10)$$

For completeness, the definitions of these weighted aggregation functions are as follows:

- Weighted mean aggregation:

$$\text{Weighted Mean} = \frac{\sum_{j=1}^{N_{\text{part}}} w_{i,j} x_{i,j}}{\sum_{j=1}^{N_{\text{part}}} w_{i,j}}$$

- Weighted variance aggregation:

$$\text{Weighted Variance} = \frac{\sum_{j=1}^{N_{\text{part}}} w_{i,j} \left(x_{i,j} - \frac{\sum_{k=1}^{N_{\text{part}}} w_{i,k} x_{i,k}}{\sum_{k=1}^{N_{\text{part}}} w_{i,k}} \right)^2}{\sum_{j=1}^{N_{\text{part}}} w_{i,j}}$$

- Weighted quantile: For the weighted quantile version, we compute the empirical weighted cumulative distribution function (CDF) and determine the minimal inverse image of the desired quantile. This process involves:
 1. Computing the Weighted CDF: Calculate the cumulative sum of the weights $w_{i,j}$ sorted by the values $x_{i,j}$, forming a step function where the height of each step is proportional to the weight of each particle.
 2. Quantile Estimation: Identify the value $x_{i,q}$ such that the cumulative weight at $x_{i,q}$ reaches the desired quantile level. This value corresponds to the position in the distribution where a certain percentage of the total weight lies below it.

One downside of computing the quantile this way is that the gradient of this estimator only affects the point corresponding to the given quantile. This can limit the effectiveness of gradient-based optimization methods, as only a single point in the distribution is adjusted in each iteration.

While more sophisticated estimators, like the Harrell-Davis estimator [29], can provide more accurate quantile estimates by considering all points in the distribution and ensuring the gradient affects every point, they are computationally intensive. Thus, they are not considered in this study.

4.1.2 . Computing a Loss for a Particle Cloud Prediction

To formalize the learning process of LINACNET , we introduce the function f_{i+1} , which represents the transformation of the particle cloud $\mathcal{D}_i$ from one position z_i to the next, z_{i+1} . This transformation depends on control parameters $\mathbf{c}_i$ and is governed by the neural network weights θ_{i+1} . The predicted particle cloud at the next position is given by:

$$\hat{\mathcal{D}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1}) = f_{i+1}(\mathcal{D}_i(\mathbf{c}), \mathbf{c}_{i+1}; \theta_{i+1}) \quad (4.11)$$

where $\hat{\mathcal{D}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1})$ denotes the predicted particle cloud at z_{i+1} , given the ground truth particle cloud $\mathcal{D}_i$ at z_i .

We also denote $\hat{\zeta}_{i \rightarrow i+1,j}(\mathbf{c}; \theta_{i+1})$ as the predicted coordinate of the j -th particle at z_{i+1} , based on the ground truth at z_i . Similarly, $\hat{\phi}_{i \rightarrow i+1,j}(\mathbf{c}; \theta_{i+1})$ represents the predicted survival probability of particle $\zeta_{i+1,j}$.

Beam observables predictions, derived from the particle distribution predictions, are defined as:

$$\hat{\mathcal{O}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1}) = h\left(\hat{\mathcal{D}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1})\right), \quad (4.12)$$

where h extracts beam characteristics from the particle cloud, as defined in Chapter 3.

Lastly, $\hat{\mathbb{1}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1})$ represents the predicted indicator of the beam being alive at z_{i+1} knowing its state at z_i :

$$\hat{\mathbb{1}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1}) = \max_j \left(\hat{\phi}_{i \rightarrow i+1,j}(\mathbf{c}; \theta_{i+1}) \right) \quad (4.13)$$

4.1.2.1 . Distribution Losses

Using these definitions, we express the loss for predicting the particle cloud at position z_{i+1} as follows. Given that LINACNET provides a coordinate prediction for each potential particle in the beam, this problem can be viewed as a standard regression task, which justifies the use of the mean squared error (MSE) as the loss function. The MSE loss for the particle positions is:

$$\mathcal{L}_{\mathcal{D}, \text{MSE}, i \rightarrow i+1}(\theta_{i+1}) = \frac{1}{|\mathcal{S}_{i+1}(\mathbf{c})|} \sum_{j \in \mathcal{S}_{i+1}(\mathbf{c})} \left| \hat{\zeta}_{i \rightarrow i+1,j}(\mathbf{c}; \theta_{i+1}) - \zeta_{i+1,j}(\mathbf{c}) \right|^2 \quad (4.14)$$

The survival probability prediction for each particle is treated as a binary classification task. Using Binary Cross Entropy (BCE), we define the survival loss as:

$$\mathcal{L}_{\mathcal{D}, \text{BCE}, i \rightarrow i+1}(\theta_{i+1}) = \frac{1}{|\mathcal{S}_{i+1}(\mathbf{c})|} \sum_{j \in \mathcal{S}_{i+1}(\mathbf{c})} -\phi_{i+1,j}(\mathbf{c}) \log\left(\hat{\phi}_{i \rightarrow i+1,j}(\mathbf{c}; \theta_{i+1})\right) - (1 - \phi_{i+1,j}(\mathbf{c})) \log\left(1 - \hat{\phi}_{i \rightarrow i+1,j}(\mathbf{c}; \theta_{i+1})\right) \quad (4.15)$$

The complete distribution loss is then:

$$\mathcal{L}_{\mathcal{D}, i \rightarrow i+1}(\theta_{i+1}) = \mathcal{L}_{\mathcal{D}, \text{MSE}, i \rightarrow i+1}(\theta_{i+1}) + \mathcal{L}_{\mathcal{D}, \text{BCE}, i \rightarrow i+1}(\theta_{i+1}) \quad (4.16)$$

4.1.2.2 . Observable Losses

To emphasize values relevant to physicists, we also define losses for beam observables.

The MSE loss for observable predictions is:

$$\mathcal{L}_{\mathcal{O}, \text{MSE}, i \rightarrow i+1}(\theta_{i+1}) = \mathbb{1}_{i+1} \left| \hat{\mathcal{O}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1}) - \mathcal{O}_{i+1}(\mathbf{c}) \right|^2 \quad (4.17)$$

The survival probability of the beam is also treated using BCE:

$$\mathcal{L}_{\mathcal{O}, \text{BCE}, i \rightarrow i+1}(\theta_{i+1}) = -\mathbb{1}_{i+1}(\mathbf{c}) \log(\hat{\mathbb{1}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1})) - (1 - \mathbb{1}_{i+1}(\mathbf{c})) \log(1 - \hat{\mathbb{1}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1})) \quad (4.18)$$

where N_{part} is the number of initial macro-particles.

The complete observable loss is then:

$$\mathcal{L}_{\mathcal{O}, i \rightarrow i+1}(\theta_{i+1}) = \mathcal{L}_{\mathcal{O}, \text{MSE}, i \rightarrow i+1}(\theta_{i+1}) + \mathcal{L}_{\mathcal{O}, \text{BCE}, i \rightarrow i+1}(\theta_{i+1}) \quad (4.19)$$

4.1.2.3 . Total Loss

The overall loss guiding LINACNET 's training combines the distribution and observable losses with respective weights $w_{\mathcal{D}}$ and $w_{\mathcal{O}}$:

$$\mathcal{L}_{i \rightarrow i+1}(\theta_{i+1}) = w_{\mathcal{D}} \mathcal{L}_{\mathcal{D}, i \rightarrow i+1}(\theta_{i+1}) + w_{\mathcal{O}} \mathcal{L}_{\mathcal{O}, i \rightarrow i+1}(\theta_{i+1}) \quad (4.20)$$

This formulation is intended to support LINACNET in learning both particle-level dynamics and specific beam characteristics effectively.

4.2 . Beam Propagation Modeling in a Sequence of Accelerator Segments

In this section, we explore how LINACNET connect multiple models corresponding to multiple segments of the accelerator to perform a modeling of the whole accelerator.

4.2.1 . Modeling

Let us recall the elements of the database introduced in Chapter 3, focusing only on the outputs saved along with the particle distribution:

$$\left\{ \mathbf{c}^{(i)}, \left(\mathcal{D}_k(\mathbf{c}^{(i)}), \mathcal{O}_k(\mathbf{c}^{(i)}), \mathbb{1}\left(z_{\mathcal{D}}^{(k)}, \mathbf{c}^{(i)}\right) \right)_{k \in \llbracket 1, N_{\mathcal{D}} \rrbracket} \right\}_{i \in \llbracket 1, n \rrbracket} \quad (4.21)$$

To construct a comprehensive model of the particle beam's evolution through the entire accelerator, LINACNET sequentially stacks modules, each corresponding to a segment of the accelerator. This design allows LINACNET to capture the cumulative effects of particle dynamics as the beam propagates through multiple segments, each characterized by unique control parameters $\mathbf{c}_i$

Recall that the function f_{i+1} models the transformation of the particle cloud $\mathcal{D}_i$ from one position z_i to the next, z_{i+1} , parameterized by θ_{i+1} and dependent on the control parameters $\mathbf{c}_i$:

$$\hat{\mathcal{D}}_{i \rightarrow i+1}(\mathbf{c}; \theta_{i+1}) = f_{i+1}(\mathcal{D}_i(\mathbf{c}), \mathbf{c}_i; \theta_{i+1}). \quad (4.22)$$

Each of these functions captures the specific transformation and dynamics within a single accelerator segment.

We introduce $\widehat{\mathcal{D}}_{k \rightarrow i}(\mathbf{c}; \Theta_{k \rightarrow i})$ to represent the predicted transformation of the particle cloud from an initial position z_k to a target position z_i , given a sequence of control parameters $(\mathbf{c}_{k+1}, \dots, \mathbf{c}_i)$ that define the state of each segment within this range. Specifically, $\widehat{\mathcal{D}}_{k \rightarrow i}(\mathbf{c}; \Theta_{k \rightarrow i})$ denotes the particle cloud at position z_i , as predicted by successively applying the segment-specific transformations from z_k to z_i :

$$\widehat{\mathcal{D}}_{k \rightarrow i}(\mathbf{c}; \Theta_{k \rightarrow i}) = f_i(f_{i-1}(\dots f_{k+1}(\mathcal{D}_k(\mathbf{c}), \mathbf{c}_{k+1}; \theta_{k+1}) \dots, \mathbf{c}_{i-1}; \theta_{i-1}), \mathbf{c}_i; \theta_i), \quad (4.23)$$

where $\Theta_{k \rightarrow i} = (\theta_{k+1}, \dots, \theta_i)$ denotes the set of parameters governing each transformation within this segment range.

To encapsulate this cumulative transformation from z_k to z_i , we define the composed function $f_{k \rightarrow i}$, representing the series of segment-specific functions applied sequentially over the range. Thus, we can rewrite $\widehat{\mathcal{D}}_{k \rightarrow i}(\mathbf{c}; \Theta_{k \rightarrow i})$ as:

$$\widehat{\mathcal{D}}_{k \rightarrow i}(\mathbf{c}; \Theta_{k \rightarrow i}) = f_{k \rightarrow i}(\mathcal{D}_k(\mathbf{c}), \mathbf{c}_{k \rightarrow i}, \Theta_{k \rightarrow i}) \quad (4.24)$$

where $\mathbf{c}_{k \rightarrow i} = (\mathbf{c}_{k+1}, \dots, \mathbf{c}_i)$ denotes the set of control parameters representing the accelerator state within this segment range.

As a specific case, $\widehat{\mathcal{D}}_{0 \rightarrow i}(\mathbf{c}; \Theta_{0 \rightarrow i})$ represents the particle cloud transformation from the initial state $\mathcal{D}_0$ up to segment i , capturing the cumulative effects of each transformation from the entrance of the accelerator.

This sequential composition of transformations in LINACNET preserves the physical fidelity of each segment-specific model, accurately capturing the dynamics unique to each segment.

4.2.2 .A Multi-Objective Problem

To ensure that LINACNET effectively models particle dynamics across multiple segments of the accelerator, we define a cumulative loss function, $\mathcal{L}_{k \rightarrow i}$, which evaluates the accuracy of the predicted particle cloud transformation from an initial position z_k to a target position z_i . This loss function is applied to the cumulative prediction generated by $f_{k \rightarrow i}$, representing the transformation over the entire segment range.

As established in the previous section, each transformation function f_{i+1} produces a segment-specific loss composed of *distribution losses* $\mathcal{L}_{\mathcal{D}, i \rightarrow i+1}(\theta_{i+1})$ and *observable losses* $\mathcal{L}_{\mathcal{O}, j \rightarrow j+1}(\theta_{i+1})$, which evaluate particle positions, survival probabilities, and key beam observables for the segment $z_i \rightarrow z_{i+1}$.

For a cumulative transformation over multiple segments, we introduce the loss $\mathcal{L}_{k \rightarrow i}(\Theta_{k \rightarrow i})$ as the evaluation of the prediction $\widehat{\mathcal{D}}_{k \rightarrow i}(\Theta_{k \rightarrow i})$ made by the composed function $f_{k \rightarrow i}$. This cumulative loss is defined as:

$$\mathcal{L}_{k \rightarrow i}(\Theta_{k \rightarrow i}) = w_{\mathcal{D}} \mathcal{L}_{\mathcal{D}, k \rightarrow i}(\Theta_{k \rightarrow i}) + w_{\mathcal{O}} \mathcal{L}_{\mathcal{O}, k \rightarrow i}(\Theta_{k \rightarrow i}), \quad (4.25)$$

where:

- $\mathcal{L}_{\mathcal{D},k \rightarrow i}$ is the distribution loss comparing $\widehat{\mathcal{D}}_{k \rightarrow i}$ to the true particle cloud $\mathcal{D}_i$
- $\mathcal{L}_{\mathcal{O},k \rightarrow i}$ is the observable loss comparing predicted beam observables $\widehat{\mathcal{O}}_{k \rightarrow i}$ derived from $\widehat{\mathcal{D}}_{k \rightarrow i}$ to the true observables $\mathcal{O}_i$

This cumulative loss $\mathcal{L}_{k \rightarrow i}$ LINACNET to evaluate and optimize the accuracy of its predictions over extended portions of the accelerator, capturing both the overall beam trajectory and the specific dynamics within each segment.

The cumulative loss $\mathcal{L}_{0 \rightarrow i}$ is especially crucial for evaluating the model's real predictive capability because, in practice, only the initial particle cloud at z_0 is known. By measuring LINACNET's ability to predict the particle cloud at z_i solely from initial conditions, $\mathcal{L}_{0 \rightarrow i}$ reflects the model's accuracy in scenarios resembling real experimental setups, where intermediate particle states are not available. The sum of these cumulative losses across all segments forms the *end-to-end loss* $\mathcal{L}_{\text{end-to-end}}$, which captures LINACNET's capability to model the particle beam's evolution from start to end.

$$\mathcal{L}_{\text{end-to-end}}(\Theta_{0 \rightarrow N}) = \sum_{i=1}^N \mathcal{L}_{0 \rightarrow i}(\Theta_{0 \rightarrow i}) \quad (4.26)$$

In parallel, the segment-specific losses $\mathcal{L}_{i \rightarrow i+1}$ ensure that each transition aligns with the physical dynamics expected within individual segments. By focusing on local accuracy, $\mathcal{L}_{i \rightarrow i+1}$ verifies that each component respects the underlying physics, enhancing the model's trustworthiness. The combined sum of these losses forms what we refer to as the *independent loss*, which ensures each segment is consistent and reliable in isolation.

$$\mathcal{L}_{\text{independent}}(\Theta_{0 \rightarrow N}) = \sum_{i=1}^N \mathcal{L}_{i-1 \rightarrow i}(\Theta_{i-1 \rightarrow i}) \quad (4.27)$$

Together, the independent loss and the end-to-end loss evaluate LINACNET in producing robust predictions that are both globally accurate and locally consistent with physics, thereby increasing the model's reliability and interpretability across the full accelerator.

4.2.3 .A Multi-Task Learning Solution

Gradient-based methods for multi-objective optimization typically involve calculating the gradient of each objective function with respect to the model parameters and then combining these gradients in a way that guides the search towards Pareto optimal solutions. The challenge lies in how these gradients are aggregated, as directly optimizing for one objective can inadvertently lead to suboptimal outcomes in others.

In the following, we introduce three alternative methods for combining and optimizing the multi-objective losses in LINACNET . Each approach explores a

unique strategy to balance the independent and end-to-end objectives, providing different perspectives on how the model's predictive accuracy and physical consistency can be prioritized.

4.2.3.1 .Scalarization Method

One common technique for combining gradients in multi-objective optimization is the Scalarization Method. This approach involves assigning a weight to each objective and summing their gradients to form a single composite gradient. Mathematically, the weighted loss $\mathcal{L}_w(\Theta)$ can be expressed as:

$$\mathcal{L}_w(\Theta_{0 \rightarrow N}) = \sum_{0 \leq k < i \leq N} w_{k \rightarrow i} \mathcal{L}_{k \rightarrow i}(\Theta_{k \rightarrow i}), \quad (4.28)$$

where $w_{k \rightarrow i}$ are the weights assigned to each objective, reflecting their relative importance.

Three specific cases of the scalarization approach illustrate how different weightings impact the optimization strategy. These cases focus on the *end-to-end loss* $\mathcal{L}_{\text{end-to-end}}$, the *independent loss* $\mathcal{L}_{\text{independent}}$, and a weighted combination of the two, where $\mathcal{L}_{\text{independent}}$ and $\mathcal{L}_{\text{end-to-end}}$ are balanced by a scalar weight.

1. *End-to-End Loss* $\mathcal{L}_{\text{end-to-end}}$: This case focuses solely on minimizing the cumulative losses $\mathcal{L}_{0 \rightarrow i}$ across all segments, where:

$$\mathcal{L}_{\text{end-to-end}} = \sum_{i=1}^N \mathcal{L}_{0 \rightarrow i}. \quad (4.29)$$

By optimizing $\mathcal{L}_{\text{end-to-end}}$, the model aims to predict the final particle cloud as accurately as possible from the initial conditions alone, reflecting the real-world context where only initial information is available.

2. *Independent Loss* $\mathcal{L}_{\text{independent}}$: This case focuses on minimizing the segment-specific losses $\mathcal{L}_{i \rightarrow i+1}$ across each transition:

$$\mathcal{L}_{\text{independent}} = \sum_{i=1}^N \mathcal{L}_{i-1 \rightarrow i}. \quad (4.30)$$

Here, the model is trained to ensure that each segment transition aligns closely with the physical dynamics expected within that specific segment, fostering local consistency. Optimizing $\mathcal{L}_{\text{independent}}$ enhances the accuracy and trustworthiness of individual components, ensuring that each segment behaves in a way that respects the underlying physics, even in isolation.

An important observation within this optimization framework is that if all the loss functions for adjacent segment sequences $(\mathcal{L}_{i-1 \rightarrow i})_{i \in \llbracket 1; N \rrbracket}$ are

equal to zero, then all other composite loss functions $(\mathcal{L}_{k \rightarrow i})_{0 \leq k < i \leq N}$ are also equal to zero.

3. *Weighted Combination*: This approach combines both the end-to-end and independent losses, using a scalar weight $w_{\text{independent}}$ to balance the two:

$$\mathcal{L}_w = w_{\text{independent}} \mathcal{L}_{\text{independent}} + (1 - w_{\text{independent}}) \mathcal{L}_{\text{end-to-end}}. \quad (4.31)$$

By adjusting $w_{\text{independent}}$, this method allows the model to find an optimal balance between local segment accuracy and global consistency.

Each of these cases offers a distinct approach to optimizing LINACNET in multi-objective contexts, with varying emphases on local and global accuracy.

4.2.3.2 .Hybrid Constrained Optimization

The Hybrid Constrained Optimization method, inspired by the Multiple Gradient Descent Algorithm (MGDA) [63], aims to balance local segment accuracy with overall beam observability. In this approach, the primary objective focuses on minimizing the independent segment losses, $\mathcal{L}_{i-1 \rightarrow i}$, ensuring that each segment transition remains physically consistent. However, constraints derived from the gradients of the end-to-end losses, $\mathcal{L}_{0 \rightarrow i}$, are imposed to maintain alignment with the cumulative, global objective.

For each segment i of the accelerator model, this approach constructs an optimal direction for updating the model parameters, balancing both segment-specific and cumulative accuracy. This direction is derived as a linear combination of two gradients:

1. The gradient of the cumulative observation loss from the accelerator's start up to segment i , *focusing only on parameters of segment i* , denoted as $\nabla_{\theta_i} \mathcal{L}_{0 \rightarrow i}$.
2. The gradient of the loss specific to the transition from segment $i - 1$ to i , $\nabla \mathcal{L}_{i-1 \rightarrow i}$, which emphasizes the fidelity of segment-to-segment dynamics.

The coefficients for combining these gradients are chosen based on a set of considerations:

1. **Alignment with the Global Objective**: The scalar product of the final descent direction with the gradient of the end-to-end loss must yield a non-negative value. This ensures that the chosen direction aligns with or improves the observation loss, avoiding any counterproductive moves.
2. **Maximization of Segment Consistency**: Among all possible combinations that satisfy the above conditions, we prefer the one that maximizes

the coefficient for the segment-specific loss. This preference roots in our aim to closely adhere to the physical dynamics between segments, ensuring that any optimization respects the inherent physics of segment transitions.

The formal optimization for segment i then becomes:

$$\nabla \mathcal{L}(\theta_i) = \alpha^* \nabla_{\theta_i} \mathcal{L}_{0 \rightarrow i}(\Theta_{0 \rightarrow i}) + \beta^* \nabla \mathcal{L}_{i-1 \rightarrow i}(\Theta_{i-1 \rightarrow i}) \quad (4.32)$$

where α^* and β^* are defined as

$$\begin{aligned} \alpha^*, \beta^* &= \arg \max_{\alpha, \beta} \beta \\ \text{subject to:} & \quad (4.33) \\ & \begin{cases} \alpha + \beta = 1 \\ \langle \alpha \nabla_{\theta_i} \mathcal{L}_{0 \rightarrow i}(\Theta_{0 \rightarrow i}) + \beta \nabla \mathcal{L}_{i-1 \rightarrow i}(\Theta_{i-1 \rightarrow i}), \nabla_{\theta_i} \mathcal{L}_{0 \rightarrow i}(\Theta_{0 \rightarrow i}) \rangle \geq 0 \end{cases} \end{aligned}$$

These equations have a formal solution:

$$\begin{aligned} \beta^* &= \begin{cases} 1 & \text{if } \langle \nabla \mathcal{L}_{i-1 \rightarrow i}(\Theta_{i-1 \rightarrow i}), \nabla_{\theta_i} \mathcal{L}_{0 \rightarrow i}(\Theta_{0 \rightarrow i}) \rangle \geq 0 \\ 1 - \frac{\langle \nabla \mathcal{L}_{i-1 \rightarrow i}(\Theta_{i-1 \rightarrow i}), \nabla_{\theta_i} \mathcal{L}_{0 \rightarrow i}(\Theta_{0 \rightarrow i}) \rangle}{\langle \nabla \mathcal{L}_{i-1 \rightarrow i}(\Theta_{i-1 \rightarrow i}), \nabla_{\theta_i} \mathcal{L}_{0 \rightarrow i}(\Theta_{0 \rightarrow i}) \rangle - \|\nabla \mathcal{L}_{i-1 \rightarrow i}(\Theta_{i-1 \rightarrow i})\|^2} & \text{otherwise} \end{cases} \\ \alpha^* &= 1 - \beta^* \end{aligned} \quad (4.34)$$

This hybrid method ensures that while the model's overall predictive accuracy at designated observation points is optimized, the optimization process remains deeply rooted in and respectful of the segment-specific physical processes. It represents a balance between achieving high-fidelity end-to-end accuracy and maintaining the integrity of the accelerator's segmental dynamics.

4.2.3.3 .Tchebycheff Approach

In the Tchebycheff approach, the objective function is defined as the maximum of the individual objectives. For our loss functions $\mathcal{L}_{k \rightarrow i}$, the Tchebycheff scalarized objective $\mathcal{L}_{\text{Tchebycheff}}$ can be mathematically formulated as:

$$L_{\text{Tchebycheff}}(\Theta_{0 \rightarrow N}) = \max_{0 \leq k < i \leq N} \{w_{k \rightarrow i} \mathcal{L}_{k \rightarrow i}(\Theta_{k \rightarrow i})\}, \quad (4.35)$$

Optimizing the Tchebycheff scalarized objective involves finding the set of parameters Θ that minimizes the maximum weighted deviation from the reference points across all objectives. This approach emphasizes the worst-performing objective in each iteration, guiding the optimization process towards regions of the parameter space where the trade-offs between objectives are balanced.

Building upon the Tchebycheff approach, the Augmented Tchebycheff method introduces an additional term to the scalarized objective function to

ensure a more balanced consideration of all objectives, not just the worst-performing one. This augmentation helps in guiding the optimization process more smoothly towards the Pareto front, particularly in regions where the objectives are closely competing.

For our composite loss functions $\mathcal{L}_{k \rightarrow i}$, the Augmented Tchebycheff scalarized objective $\mathcal{L}_{\text{AugTchebycheff}}$ is enhanced by a summation of all the weighted objectives, and can be mathematically formulated as:

$$\mathcal{L}_{\text{AugTchebycheff}}(\Theta_{0 \rightarrow N}) = \max_{0 \leq k < i \leq N} \{w_{k \rightarrow i} \mathcal{L}_{k \rightarrow i}(\Theta_{k \rightarrow i})\} + \rho \sum_{0 \leq k < i \leq N} w_{k \rightarrow i} \mathcal{L}_{k \rightarrow i}(\Theta_{k \rightarrow i}), \quad (4.36)$$

where ρ is a small positive value that ensures all objectives are accounted for in the optimization process, thus preventing the scenario where a single objective dominates the optimization direction at the expense of others.

4.3 .Validation

This section presents the experiments conducted to assess the modeling of the ThomX Linac using LINACNET . After presenting the goal of experiments in Section 4.3.1, Section 4.3.2 assesses LINACNET 's ability to model a particle cloud through experiments performed on the first segment of the Linac. Additionally, this section investigates the influence of various LINACNET hyperparameters, leading to the selection of an optimal set, which is used throughout the remaining work. Section 4.3.3 reports the performance of LINACNET on the entire Linac, analyzing the effects of different solutions to the multi-objective optimization problem described in Section 4.2.2.

4.3.1 .Goals of Experiments

This section outlines the primary goals of the experiments, which are designed to validate the effectiveness of LINACNET and assess the importance of its various hyperparameters.

4.3.1.1 .Capability to Model a Particle Cloud

The first goal of LINACNET is to accurately model the propagation of a particle beam within a single segment of the accelerator. The model's performance is primarily evaluated by its ability to predict a set of observables at the end of the section, using the $R^2_{0 \rightarrow 1}$ and $\text{MAE}_{0 \rightarrow 1}$ performance metrics, as detailed in Chapter 3.2. In addition to predicting the observables, LINACNET must also accurately model the entire particle set. This is necessary for two reasons: (1) to ensure the model can continue making predictions in subsequent sections of the accelerator, and (2) to increase confidence in the model's predictions by enriching the data with more detailed beam characteristics. The accuracy of these particle-level predictions is assessed using the $R^2_{\mathcal{D}, 0 \rightarrow 1}$ metric.

Given the large number of hyperparameters involved, the initial section of the accelerator is used as a testbed for optimizing these hyperparameters, which is then applied to train the model across the entire accelerator. These hyperparameters include the network size, architecture type (see Section 4.1.1.2, the aggregation function (see Section 4.1.1.4), the segment loss function (see Section 4.1.2), as well as various training parameters such as learning rates, weight decay, and gradient clipping.

4.3.1.2 .Capability to Model Sequential Segments of a Particle Accelerator

The second goal is to achieve high accuracy in modeling all segments of the accelerator. The key indicators for evaluating the model's performance in this context are the $\{R_{0 \rightarrow i}^2\}_{0 < i \leq N}$ metrics, which assess the model's ability to reproduce all the observables based solely on the initial set of control parameters.

Another important set of metrics includes $\{R_{i-1 \rightarrow i}^2\}_{0 < i \leq N}$. These metrics help to confirm whether the model accurately captures the underlying physical behavior of the accelerator. A model that performs well on the $(0 \rightarrow i)_{0 < i \leq N}$ metrics but poorly on the $(i-1 \rightarrow i)_{0 < i \leq N}$ metrics would raise concerns about its reliability, as it may diverge from the true physical processes in specific segments of the accelerator.

Finally, the metrics $\{R_{\mathcal{D}, 0 \rightarrow i}^2\}_{0 < i \leq N}$ and $\{R_{\mathcal{D}, i-1 \rightarrow i}^2\}_{0 < i \leq N}$ are examined to ensure that the model not only predicts the predefined observables but also provides rich insights into the whole 6D particle distribution.

To achieve these objectives, the training strategies outlined in Section 4.2.3, as well as the survival probability modeling described in Section 4.1.1.3, are studied.

4.3.2 .First Segment Modeling

4.3.2.1 .LINACNET Particle Cloud Modeling Hyperparameters Optimization

The first experiment presented in this section involves a comprehensive hyperparameter optimization of a single LINACNET segment, conducted on the first segment of the accelerator. The goal of this optimization is to confirm that LINACNET can model the propagation of a particle cloud and establish a suitable baseline of hyperparameters for the remainder of the study.

The hyperparameters considered include: (i) the size (depth and width) of the four multi-layer perceptron that compose LINACNET, (ii) the integration of control parameters within the architecture (see Section 4.1.1.2), (iii) the choice of the aggregation function (see Section 4.1.1.4), (iv) the distribution of loss between local and global losses, and (v) various training parameters including the learning rate, the weight decay, and the gradient clipping.

Models are evaluated based on two metrics (see Section 3.2): (a) the ability to accurately predict the beam observables, measured using the R-squared metric $R_{0 \rightarrow 1}^2$, and (b) the ability to model the entire particle cloud, assessed through the R-squared of the particle coordinates $R_{\mathcal{D}, 0 \rightarrow 1}^2$.

In practice, the optimization is carried out using Optuna's [2] MultiObjective Tree-structured Parzen Estimator (MOTPE) [46] sampler. Each model is allocated a maximum runtime of 10 hours and is stopped after 20 epochs without an improvement on the validation loss. The hyperparameters tuned are presented in Table 4.2.

Fig. 4.2 shows the result of this optimization process on 1210 trials. The set of points forming the pareto front are presented in Table 4.3.

Firstly, we observe a strong positive correlation between the two metrics, with data points aligned along a linear trend. Notably, the Pareto-optimal set is limited to just two points, which are closely matched in terms of their objective values. This result is consistent with the fact that both metrics are statistical functions derived from the particle cloud. As a result, high values for the particle cloud metrics are naturally associated with high values for the observables.

Secondly, the trials on the pareto front achieve high values for both R-squared metrics, exceeding 0.99. Table 4.1 presents the Mean Absolute Error (MAE) on the 14 observables. These errors are below the precision measure of all diagnostic stations installed on ThomX (see Table 3.3). This result demonstrates that LINACNET can accurately replicate the behavior of a particle cloud within one segment of the accelerator.

Finally, Fig. 4.3 illustrates various hyperparameter values obtained during the optimization process. From these figures, we can draw the following hypothesis:

1. The aggregation function is not predominant in the overall performance of the network. This is illustrated by each aggregation function being represented by a trial in the top right quadrant of the objective space.
2. Contrary to the original PointNet [11], the size of the global network, before aggregation, does not need to be large for an effective training.
3. Best results are obtained with a low value of gradient clipping, suggesting a high sensitivity of the network in the parameter space.
4. LINACNET trained on the first segment of the Linac of ThomX does not need strong regularization. This is consistent with the data generation process by a simulator, preventing imperfect data in the database.
5. There is a clear conflict between the local and the global loss. A strong global loss leads to a better modeling of the beam observables, while a strong local loss leads to a better modeling of the particle cloud.

These hypotheses result from the hyperparameter optimization process. This process can bias the set of trials. These hypotheses need to be confirmed by a scan of each variable. Considering the size of the hyperparameter space, we cannot perform a multi-dimensional scan of each variable. We choose to select one trial and to perform a scan on each variable independently with all other hyperparameters set to the value of this trial. Considering the result of the hyperparameter optimization, trial 501 serves as a baseline.

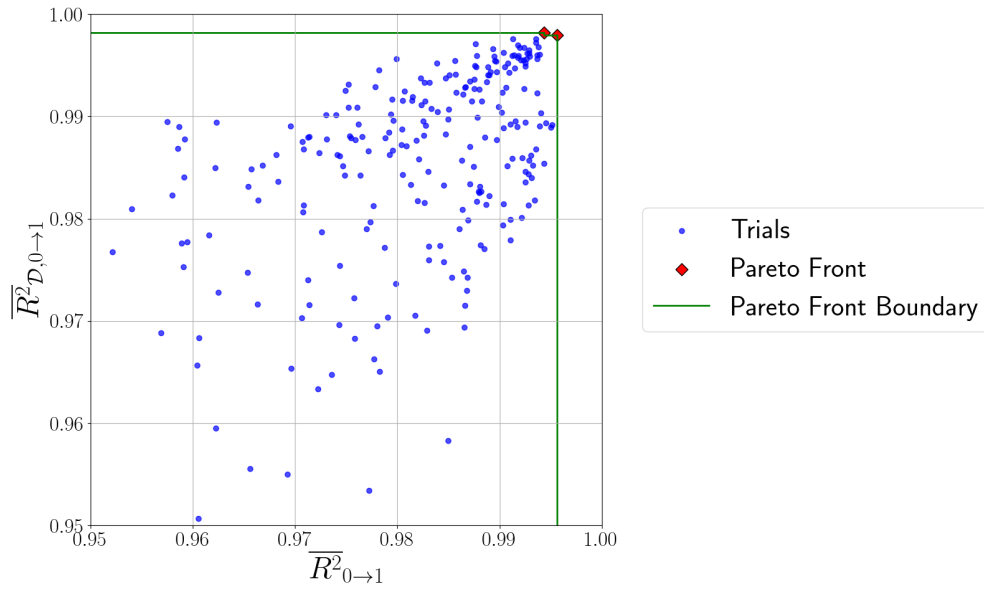


Figure 4.2: Visualization of the Pareto front of the optimization of one segment of LINACNET . Each blue point represents the results of one trials, while the red diamonds highlight the Pareto-efficient points. The green staircase line illustrates the boundary between the dominated and non-dominated regions.

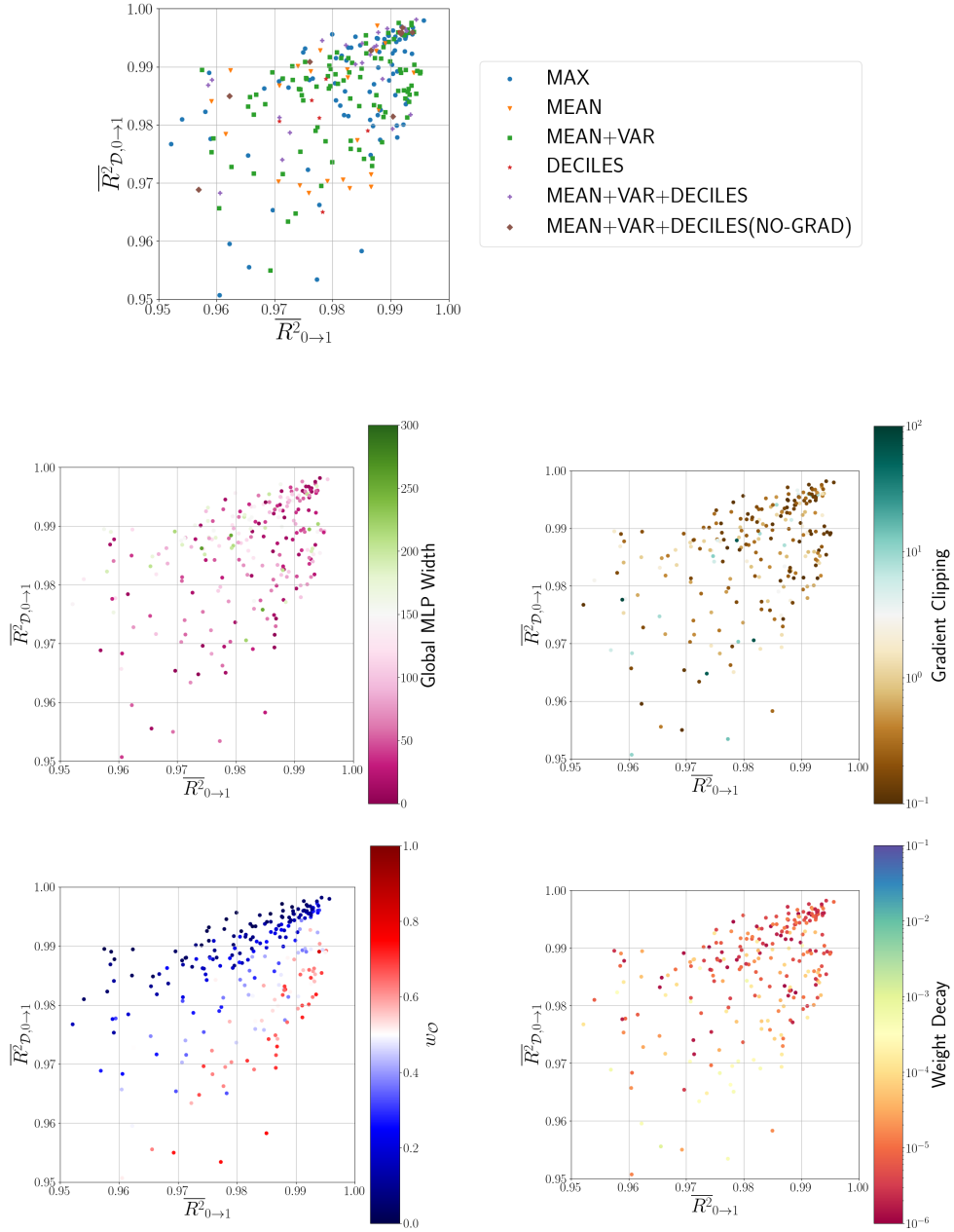


Figure 4.3: Values of various hyperparameters of LINACNET used during the optimization.

Table 4.1: Mean Absolute Error of the trials on the Pareto Front. When applicable, MAE are below the precision measure of all diagnostic stations installed on ThomX (see Table 3.3).

		Trial 501	Trial 596
Hor. Trans. Dim.	x_{avr} (mm.)	0.0105	0.0125
	x_{std} (mm.)	0.0133	0.0117
	x'_{std} (mrad.)	0.0390	0.0384
	$\epsilon_{x,norm}$ (π mm. mrad.)	0.312	0.281
Vert. Trans. Dim.	y_{avr} (mm.)	0.0103	0.0118
	y_{std} (mm.)	0.0141	0.0118
	y'_{std} (mrad.)	0.0389	0.0431
	$\epsilon_{y,norm}$ (π mm. mrad.)	0.302	0.336
Long. Dim.	$E_{kin,avr}$ (MeV.)	0.00337	0.00514
	z_{std} (mm.)	0.0123	0.0115
	$E_{kin,std}$ (keV.)	2.020	1.986
	$\epsilon_{z,norm}$ (π keV. mm.)	0.515	0.497
Other	Q (nC.)	0.00834	0.00929
	$ \mathcal{S} $	88.500	99.099

Table 4.2: Prior distribution of the different hyperparameters of LINACNET used during the optimization. $\mathcal{U}$ denotes a uniform distribution and $\mathcal{LU}$ a log-uniform distribution.

Hyperparameter	Prior Distribution
$w_{\mathcal{O}}$	$\mathcal{U}(0.0, 1.0)$
Architecture Type	(a.), (b.)
Aggregation ¹	MAX, MEAN, MEAN+VAR, DECILES, MEAN+VAR+DECILES, MEAN+VAR+DECILES(NO-GRAD)
Context MLP ν : Width, Depth	$\mathcal{U}(\{1, \dots, 300\}), \mathcal{U}(\{1, \dots, 5\})$
Local MLP γ : Width, Depth	$\mathcal{U}(\{1, \dots, 300\}), \mathcal{U}(\{1, \dots, 5\})$
Global MLP η : Width, Depth	$\mathcal{U}(\{0, \dots, 300\}), \mathcal{U}(\{1, \dots, 5\})$
Regression MLP χ : Width, Depth	$\mathcal{U}(\{1, \dots, 300\}), \mathcal{U}(\{1, \dots, 5\})$
lr	$\mathcal{LU}(0.0001, 0.1)$
weight decay	$\mathcal{LU}(0.000001, 0.1)$
gradient clipping	$\mathcal{LU}(0.1, 100.0)$

Table 4.3: Set of Pareto-efficient runs for the optimization of one segment of LINACNET

Trial ID	$w_{\mathcal{O}}$	Architecture Type	Context MLP	Local MLP	Global MLP	Aggregation	Regression MLP	lr	weight decay	gradient clipping	$R_{0 \rightarrow 1}^2$	$R_{\mathcal{D},0 \rightarrow 1}^2$
501	0.0646	(b.)	(39,)	(259, 259)	(105,)	MAX	(254, 254, 254)	0.002	9.022e-06	0.128	0.9957	0.9980
596	0.0609	(b.)	(107,)	(222, 222)	(6,)	MEAN+VAR+DECILES	(252, 252, 252)	0.004	3.015e-06	0.119	0.9943	0.9982

4.3.2.2 .LINACNET Particle Cloud Modeling Hyperparameters Scan

In this subsection, we explore the impact of some hyperparameters on the overall performance of LINACNET for a single accelerator segment. We conduct scans of relevant hyperparameters, starting from a baseline set established through optimization in the previous section. These hyperparameters include the aggregation function, architecture type, size of the global MLP, global loss weight, gradient clipping value, learning rate and weight decay. The results of these scans are presented in Fig 4.4. Each experiment is repeated using ten different seeds.

- **Aggregation Function.** All considered aggregation functions considered achieve similar good results on the final performance of LINACNET. The MAX aggregation function demonstrates slightly better performance than MEAN, a result that aligns with previous findings from PointNet [11] and the common use of Max Pooling in Computer Vision [9]. Introducing variance into the MEAN function offers no noticeable performance gain. While the DECILES function shows marginally improved results compared to MEAN, its performance remains similar to that of MAX. Moreover, DECILES increase the output vector size by a factor of 10, making MAX the more efficient choice. Finally, adding MEAN+VAR to DECILES does not result in any significant performance enhancement. In summary, the MAX function emerges as the most suitable choice for the remainder of the study.
- **Architecture Type.** Incorporating the control vector in the middle or at the beginning of the architecture has no impact on the final performance. The final network is sufficient to combine the information of the control to the one of the entering beam distribution.
- **Global MLP η Width.** When the width of the global MLP is reduced to 0, LINACNET effectively reduces to a siamese network. Experimental results show that a global network width greater than 0 leads to improved performance, demonstrating the superiority of LINACNET over the siamese architecture in terms of overall performance and underlying the necessity to model particle interactions. Unlike the findings of PointNet [11] in the context of 3D vision, which uses a width of 1024 for the global network, we find that even a small width (greater than or equal to 5) is sufficient to achieve strong performance, with no significant improvement observed up to a width of 300.
- **Loss weight.** When the loss component assigned to observables is reduced to 0, performance on these metrics worsens compared to cases

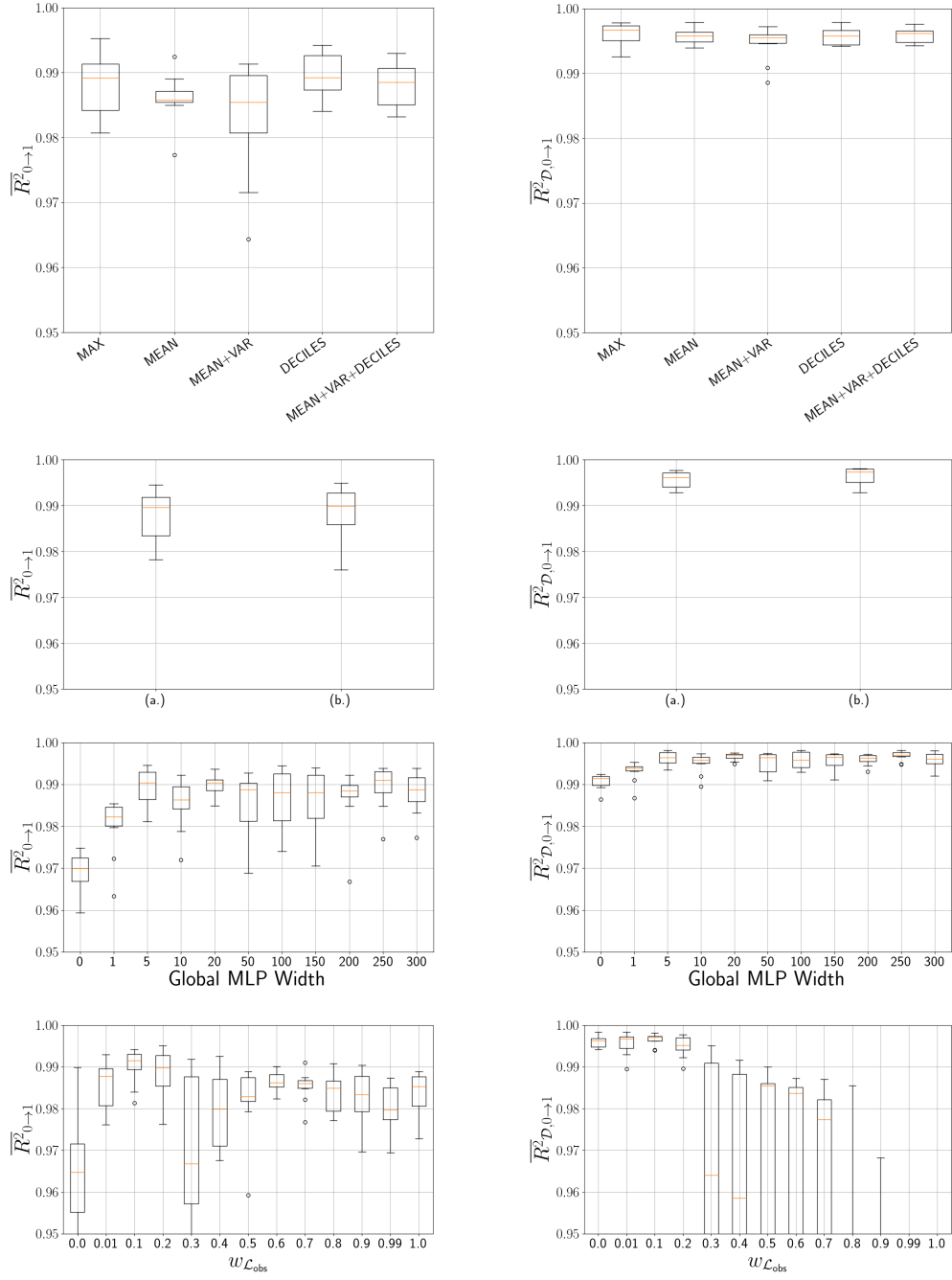


Figure 4.4: Scan of various hyperparameters of LINACNET using trial 501 as a reference. Trials are repeated over 10 different seeds. Each row of figure represents one hyperparameter. The left column indicates performance on the beam observables. The right column indicates performance on the particle cloud. Boxplot represents the median, first and third quartile with whiskers extending to the farthest data point within 1.5 the IQR from the box.

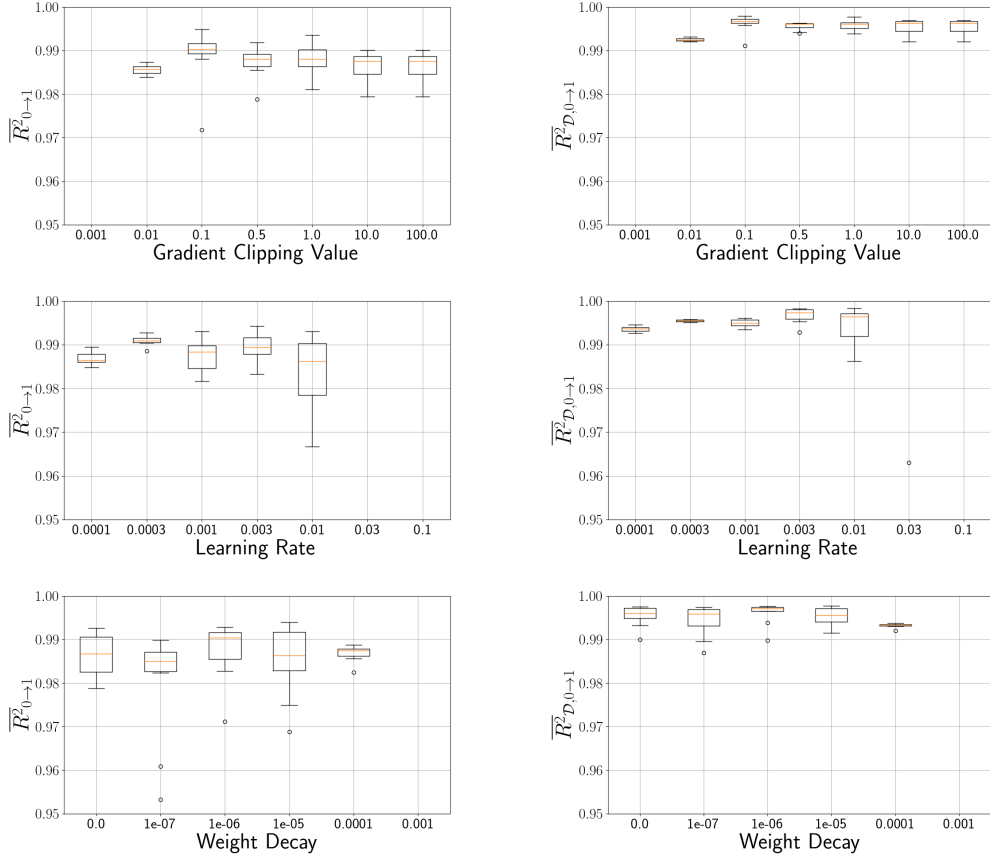


Figure 4.4: Scan of various hyperparameters of LINACNET using trial 501 as a reference. Trials are repeated over 10 different seeds. Each row of figure represents one hyperparameter. The left column indicates performance on the beam observables. The right column indicates performance on the particle cloud. Boxplot represents the median, first and third quartile with whiskers extending to the farthest data point within 1.5 the IQR from the box.

where this weight is greater than 0. Even a minimal allocation to the observable loss ($w_O = 0.01$) significantly boosts LINACNET's ability to predict observables. However, when this weight becomes too large ($w_O > 0.2$), LINACNET's prediction performance on the particle cloud deteriorates significantly. In this scenario, LINACNET still accurately predicts observables ($\overline{R^2}_{0 \rightarrow 1} > 0.95$), despite poor particle cloud performance. Here, LINACNET relies on the particle cloud as a learning scaffold for observables, but this scaffold becomes unreliable, leading to a loss of robustness and interpretability in the network's predictions.

- **Training Parameters.** Scans of Gradient Clipping values, learning rate and weight decay indicate that the baseline cannot be significantly improved by modifying these parameters.

These various scans confirm the validity of the hyperparameters identified through the optimization process. They also highlight the critical importance of the loss weighting between the local and global components, as well as the superiority of LINACNET compared to a full Siamese architecture. The influence of other parameters appears to be comparatively limited.

4.3.3 .Multi-segment modeling

In the previous section, LINACNET demonstrated its ability to model the beam propagation within a single segment of the Linac. This section focuses on LINACNET's capacity to model the entire sequence of segments. The first part explores the optimal approach for managing particle survival status updates (see Section 4.1.1.3), establishing a baseline model for addressing the multi-task learning challenge posed by the sequential structure of the segments (see Section 4.2.3).

4.3.3.1 .Particle Status Update

In this section, we investigate the three strategies presented in Section 4.1.1.3, namely the RAW strategy, the MIN strategy and the CONDITIONAL strategy .

Training is conducted on the full Linac, which consists of $N = 25$ segments. Since these strategies primarily impact how accurately the beam is modeled across multiple segments, training is performed using the end-to-end losses ($\mathcal{L}_{\text{end-to-end}}$), which govern the propagation over the entire Linac. The same hyperparameters determined in the previous section are used here.

As in the previous section, training is terminated if there is no improvement in validation loss after 20 epochs. However, in this case, a maximum training time of 100 hours is imposed (compared to 10 hours in the previous section). Given the extended training time, we reduce the number of trials to three different seeds, as opposed to 10 in the previous section.

Figure 4.5 presents the outcomes of the study on observable and distribution metrics for the three particle survival modeling strategies, revealing no

significant performance discrepancies among them. The median end-to-end prediction quality for observables is reasonably high, achieving $R_{0 \rightarrow N}^2 \geq 0.6$, while the median distribution prediction reaches a higher level of accuracy, with $R_{\mathcal{D}, 0 \rightarrow i}^2 \geq 0.8$. This suggests that the models effectively capture the particle distribution dynamics across the Linac.

However, independent predictions for observables ($\hat{\mathcal{O}}_{i \rightarrow i+1}$) sometimes exhibit poor accuracy, an outcome that initially appears counterintuitive. Although segment-wise tasks (i.e., $i \rightarrow i+1$) might seem simpler than the end-to-end task $0 \rightarrow i$, the observed discrepancy is explained by the use of end-to-end losses during training. This approach may lead the model to adopt simplified representations that compromise physical consistency at the segment level.

In contrast, the independent predictions for distributions demonstrate robust performance, with $R_{\mathcal{D}, i \rightarrow i+1}^2 \geq 0.8$ and values approaching 1 after 7 meters. This stability indicates that the model can reliably capture distribution dynamics within individual segments, even when training is based solely on end-to-end losses. The findings underscore the effectiveness of the strategies in modeling distribution accuracy, while also suggesting that segment-level observable consistency may require additional adjustments to maintain alignment with physical principles throughout the Linac.

4.3.3.2 .Multi-Task Learning Study

In this section, we assess the multi-task learning strategies detailed in Section 4.2.3 to evaluate LINACNET's capacity to accurately model the sequential structure of the accelerator. Given the computational limitations (high GPU memory usage) associated with calculating all individual losses $\mathcal{L}_{k \rightarrow i}$, we restrict our study to strategies combining independent losses $\mathcal{L}_{i \rightarrow i+1}$ and end-to-end losses $\mathcal{L}_{0 \rightarrow i}$.

This analysis considers several scalarization approaches, varying the weight $w_{\text{independent}}$ from 0 to 1, in addition to hybrid, Tchebycheff, and MGDA (Multiple Gradient Descent Algorithm) methods, applied solely to the independent and end-to-end losses. As in the previous section, we maintain the same experimental setup.

We begin by examining the effect of $w_{\text{independent}}$ in the scalarization method. Results, presented in Figure 4.6, reveal two key findings. First, a lower $w_{\text{independent}}$ value consistently enhances end-to-end performance on observable metrics, underscoring the advantage of prioritizing the overall sequence over independent segment-level optimization. Secondly, in the case where training is entirely independent ($w_{\text{independent}} = 1.0$), segment-wise metrics ($i \rightarrow i+1$) significantly improve, achieving $R_{\mathcal{D}, i \rightarrow i+1}^2 > 0.99$. However, this gain in segment-level accuracy does not translate to the end-to-end task; while performance slightly improves within the first 2 meters of the accelerator, it declines abruptly in later sections. This behavior suggests that independent segment errors,

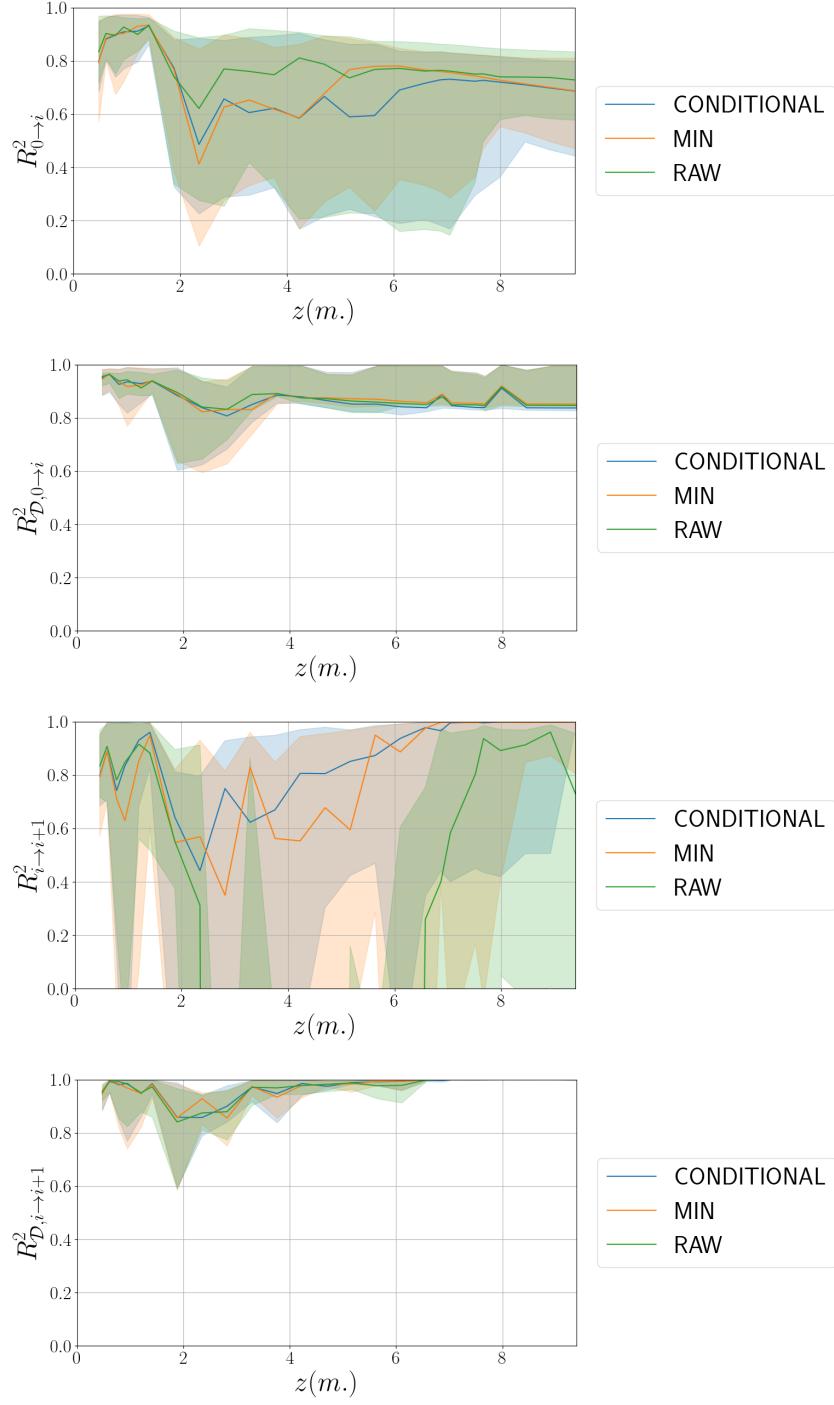


Figure 4.5: Performance indicators for the difference particle survival modeling strategies. Line indicate the median of the various R^2 vectors, while shaded areas indicate Interquartile Range (IQR). Plots are reported for observation-centric metric R^2 as well as distribution-centric metric R^2_D for independent segment ($i \rightarrow i + 1$) as well as end-to-end evaluation ($0 \rightarrow i$).

while minimized locally, accumulate across segments, leading to an amplification effect that increases overall error. Conversely, even a small end-to-end weighting ($tw_{\text{end-to-end}} = 1 - w_{\text{independent}}$) provides a form of regularization that seems to align segment errors, thus reducing error propagation throughout the model sequence.

Figure 4.7 compares the results of the Tchebycheff-based, MGDA-based, independent, and end-to-end strategies. As shown, no single approach surpasses the end-to-end strategy on end-to-end metrics, nor does any strategy outperform the independent strategy on segment-level metrics. These results align with recent findings indicating that adaptive multi-objective optimization techniques do not outperform standard scalarization methods, highlighting a trade-off in multi-task learning: end-to-end strategies best capture global behaviors, while independent strategies excel at local accuracy.

4.3.3.3 .Conclusion

In summary, this section demonstrates that LINACNET successfully models beam propagation across multiple segments of the accelerator, extending its capacity beyond single-segment modeling. The strategies applied achieve a reasonably accurate representation of observable evolution throughout the Linac. However, a conflict remains between optimizing performance within each individual segment and maintaining accuracy across the entire sequence. While independent segment modeling enhances local precision, aligning the individual errors of each segment proves essential to prevent error amplification in subsequent segments. Despite various approaches explored, a fully satisfactory balance between segment-level and end-to-end accuracy has yet to be achieved, highlighting the challenge of aligning local and global modeling objectives in multi-segment beam propagation.

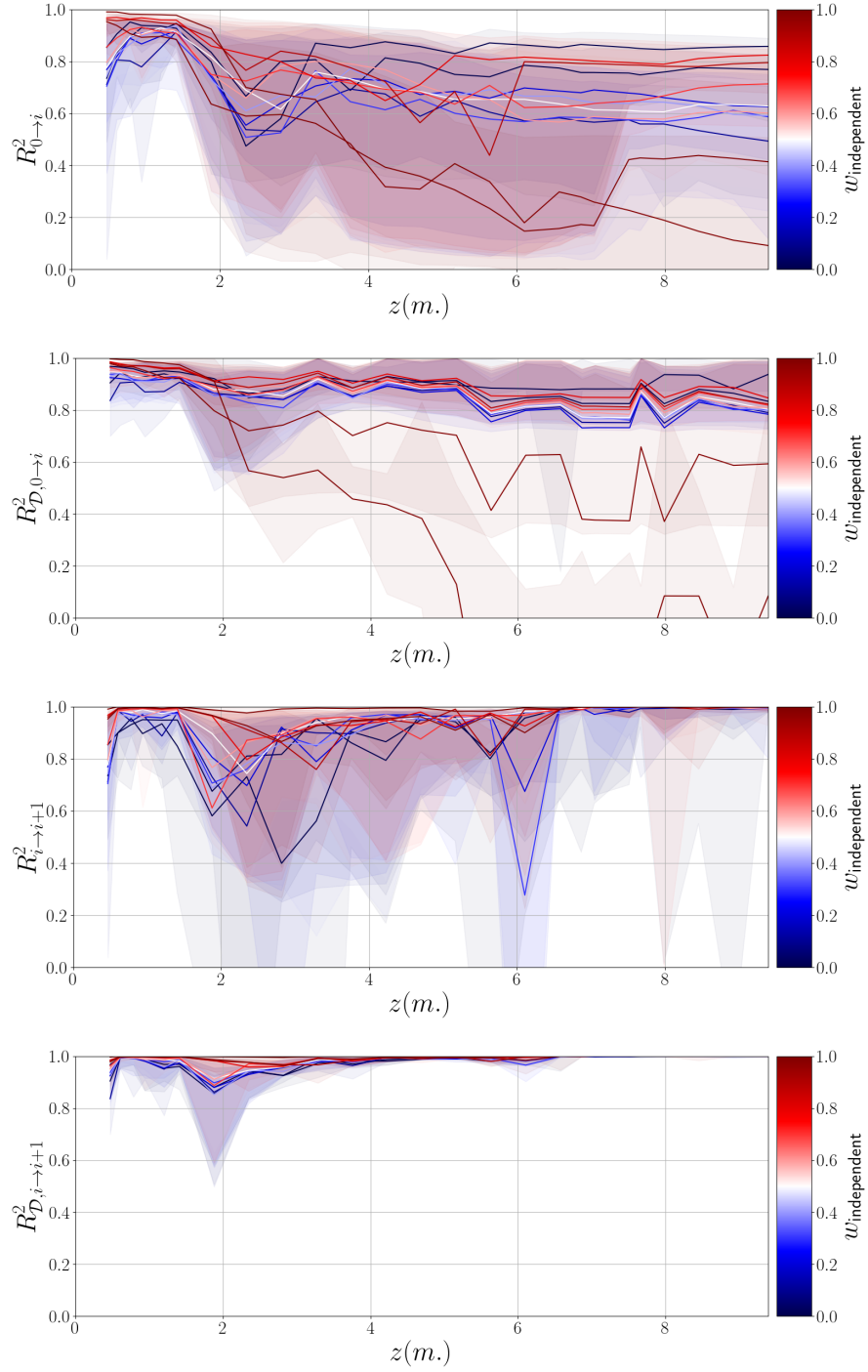


Figure 4.6: Effect of varying $w_{\text{independent}}$ in the scalarization method on end-to-end and independent metrics. The plots illustrate median R^2 values for both observable metrics (R^2) and distribution metrics (R^2_D) as $w_{\text{independent}}$ is adjusted from 0 to 1. Shaded areas represent the Interquartile Range (IQR) for each metric.

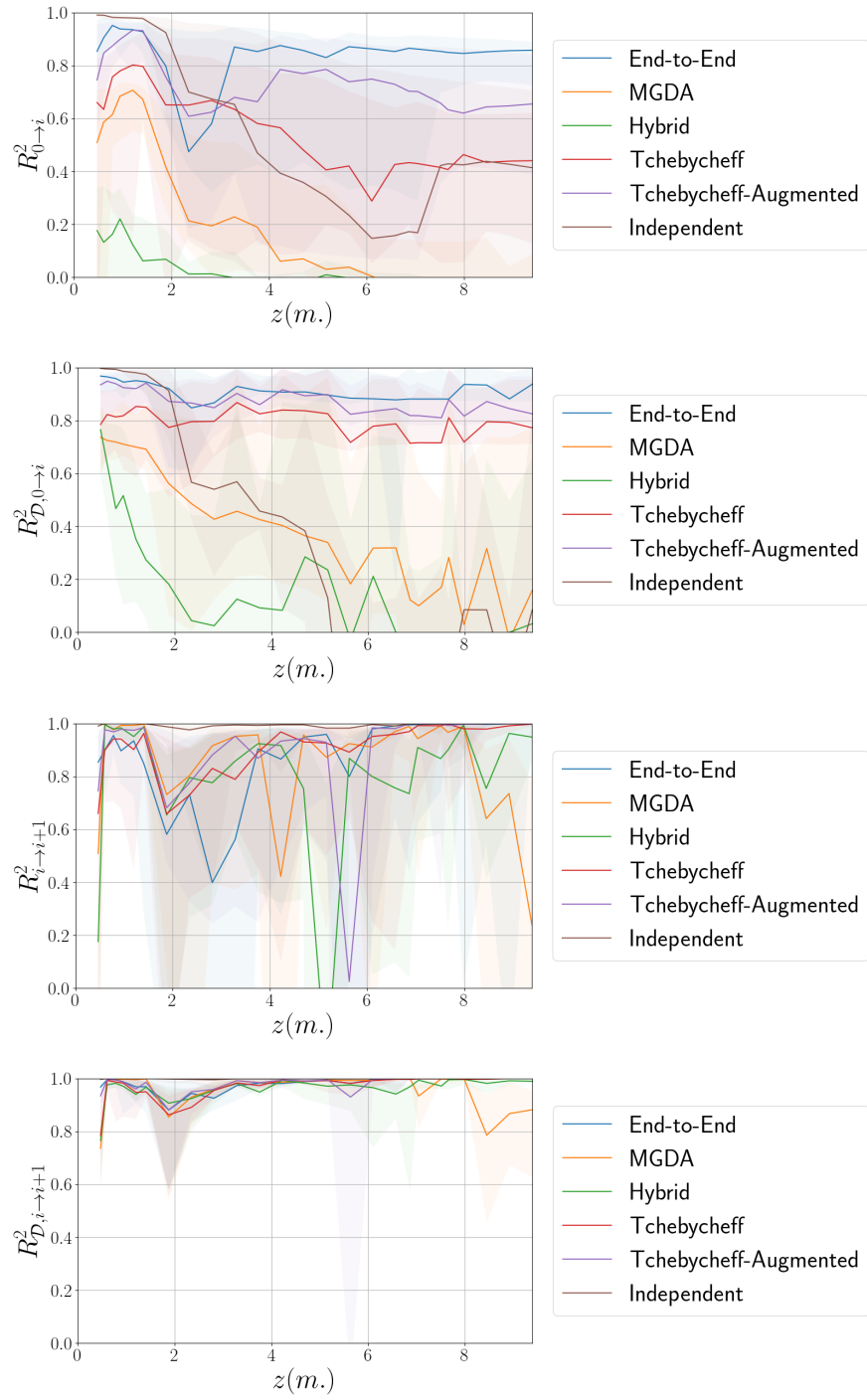


Figure 4.7: Comparison of Tchebycheff, MGDA, independent, and end-to-end strategies in multi-task learning. Plots report median R^2 values for observable (R^2) and distribution metrics (R^2_D) across different strategies, with IQRs shown as shaded areas.

5 -Dynamical System Modeling: an Implicit ODE Approach

Contents

5.1 Machine Learning for Dynamical System Modeling	92
5.2 INODE Method	93
5.2.1 From NODE to INODE : Learning an ODE in an Implicit Space	93
5.2.2 Adjusting for Control Parameters	94
5.2.3 Evaluation Process with INODE	95
5.2.4 Summary of the INODE Method	95
5.3 INODE Theoretical Foundation	96
5.3.1 Implicit Trajectories	98
5.3.2 Universal Approximation Theorem	100
5.4 INODE Experiments	103
5.5 Conclusion and Future Works	106

The approach presented in the preceding chapters focuses explicitly on predicting the transport of a particle distribution from one point in the accelerator to another. This chapter, however, shifts perspective: rather than modeling the direct transport of the entire distribution, we aim to learn the continuous function governing the evolution of the beam observables over the length of the accelerator. We will refer to this continuous function as a *trajectory*. To manage computational complexity, we only model the observables evolution rather than the entire particle cloud evolution, as the latter would be computationally prohibitive.

This chapter is exploratory, providing an initial evaluation of the proposed method. Our theoretical analysis and preliminary experiments shed light on the general validity of the approach, while further validation is required to better understand the practical merits and limitations thereof.

As discussed in Chapter 1, the evolution of the beam observables results from a set of underlying differential equations, which define a nonlinear dynamical system. Solving these equations directly is the primary function of particle accelerator simulators, which aim to model the accelerator's dynamics with high accuracy. However, accurately solving such dynamical systems can be computationally expensive.

Most generally, differential dynamical systems arise in many fields, such as climate modeling, population dynamics, and robotics, where solving the governing equations is often computationally intensive. To address these computational challenges, machine learning has been increasingly applied to approximate the solutions of these differential systems more efficiently. Various methods, such as Neural ODEs, Physics-Informed Neural Networks (PINNs), and Recurrent Neural Networks (RNNs), have been developed to tackle these problems, as briefly introduced in Section 5.1.

In current machine learning methods for modeling dynamical systems, two major issues arise. First, while differential equations provide a powerful representation for describing these systems, solving these equations during model training is computationally demanding. Second, the trajectories considered may contain discontinuities, which pose significant challenges for methods based on ordinary differential equations (ODEs), whose solutions are inherently continuous. This chapter introduces a new method, Implicit Neural ODE (INODE), which is described in Section 5.2, to address these challenges while retaining the powerful representation offered by ODEs. The problem is reframed as a simple pretext regression task that handles discontinuous trajectories while externalizing the ODE solving process by preprocessing the trajectories into an implicit space.

Section 5.3 presents the theoretical framework underlying INODE, providing a more generalized perspective on the approach introduced in Section 5.2. Specifically, it identifies sufficient conditions for preprocessing the trajectories, ensuring that the proposed method can be applied to trajectories with local discontinuities. Additionally, it proves that under these conditions, INODE is a universal approximation of the said trajectories. Furthermore, the error of the eventual trajectory model is upper bounded by a function of the error on the pretext task.

Section 5.4 presents some preliminary results by applying INODE to the Linac Database. These results provide an initial evaluation of the method's ability to model accelerator dynamics, serving as a basis for understanding the potential and limitations of the approach. Finally, Section 5.5 concludes the chapter by discussing the limitations of the method and suggesting future directions for further research and development.

5.1 .Machine Learning for Dynamical System Modeling

In Recurrent Neural Networks (RNNs), the system's evolution from one time step to the next is described by the following equation:

$$\mathbf{u}(t_{i+1}) = \mathbf{u}(t_i) + f_{\theta}(\mathbf{u}(t_i)) \quad (5.1)$$

Here, f is a function parameterized by θ , representing the dynamics gov-

erning the system's behavior.

The concept of Neural ODE [12] can be interpreted as a continuous generalization of the classical recurrent neural network by adding additional layers and progressively decreasing the step size. NODE's novelty lies in considering the function u as the solution of an ODE:

$$\frac{du}{dt}(t) = f(u(t), t) \quad (5.2)$$

A neural network f_θ is employed to approximate the unknown f function. The estimate function $\hat{u}$ is the solution to the initial value problem defined by f_θ , and $u(t_0)$.

$$\frac{d\hat{u}}{dt}(t) = f_\theta(\hat{u}(t), t) \quad (5.3)$$

The state estimate at any time can thus be predicted by solving the ODE:

$$\hat{u}(t) = \text{ODESolve}(f_\theta, u(t_0), t_0, t) \quad (5.4)$$

$\text{ODESolve}(f_\theta, u(t_0), t_0, t)$ serves as a black-box that solves Eq. (5.3), starting with the value $u(t_0)$ at time t_0 .

To calculate the gradient of the loss function $L(\hat{u})$, the adjoint method is utilized [12]. This method computes the gradient of the loss by solving another ODE backward in time, enabling a more efficient memory consumption than directly backpropagating the loss through the ODE solver.

Despite the efficient memory consumption of such a method, the full differential equation needs to be solved at each training step, which becomes particularly challenging when dealing with complex underlying neural networks or long sequences.

5.2 .INODE Method

In this section, we introduce the fundamental principles of the Implicit Neural ODE (INODE) method and illustrate its implementation using a specific preprocessing step: integration. The core idea behind this approach is that integration acts as a smoothing operator. While the original trajectory of a system might exhibit discontinuities, its integral counterpart is inherently continuous. Instead of directly modeling the trajectory through an ODE, the INODE method aims to learn the ODE governing its integral representation.

5.2.1 .From NODE to INODE : Learning an ODE in an Implicit Space

As said, NODE aims at learning the derivative of trajectory u . In contrast, rather than directly modeling the original trajectory u , the INODE method focuses on an *implicit trajectory* v , obtained through a preprocessing step.

Specifically, in this section, v is defined as the integral of u , yielding a smoother trajectory.¹

$$v(t) = \int_{t_0}^t u(\tau) d\tau \quad (5.5)$$

Instead of learning an ODE for u , we focus on learning the dynamics governing v . In this setting, the differential relation is given by:

$$\frac{dv}{dt}(t) = u(t) \quad (5.6)$$

This formulation implies that the ODE for v is simpler to learn, as we have access to both the original trajectory u from the database and its implicit form v obtained via the preprocessing step. Consequently, learning f_θ becomes a straightforward regression problem where:

$$u(t) = f_\theta(v(t), t) \quad (5.7)$$

The regression problem defined by Equation (5.7) serves as a *pretext task* [17]. We start by learning a function f_θ that maps the implicit trajectory to the original trajectory. Once this function is learned, it yields a solution to the prediction task of the trajectory u , as described in Section 5.2.3.

5.2.2 .Adjusting for Control Parameters

Let us recall that our model aims to create a surrogate model capable of predicting trajectories of observables for various sets of control parameters, denoted by $\mathbf{c}$. Therefore, our objective extends beyond learning a single trajectory; we aim to generalize over a range of trajectories based on different values of $\mathbf{c}$. Initially, we simplified the presentation by omitting $\mathbf{c}$ in the equations, but in practice, the functions u , v , and f_θ must all depend on both t and $\mathbf{c}$. Consequently, Equation (5.7) should be revised to:

$$u(t, \mathbf{c}) = f_\theta(v(t, \mathbf{c}), t, \mathbf{c}) \quad (5.8)$$

This formulation reflects that our model aims to capture the generative process that maps a set of control parameters $\mathbf{c}$ to a trajectory $u(\cdot, \mathbf{c})$.

Traditionally, modeling dynamical systems relies on the temporal variable t , representing time. However, in the context of particle accelerators, the focus shifts to the evolution of observables along the longitudinal axis z of the accelerator. Despite this distinction, we will adhere to the conventional use of

¹Traditionally, modeling dynamical systems relies on the temporal variable t , representing time. However, in the context of particle accelerators, the focus shifts to the evolution of observables along the longitudinal axis z of the accelerator. Despite this distinction, we will adhere to the conventional use of the temporal variable t in this section for consistency with existing literature. In our specific case, t should be interpreted as the longitudinal position z along the accelerator.

the temporal variable t in this section for consistency with existing literature. In our specific case, t should be interpreted as the longitudinal position z along the accelerator.²

5.2.3 .Evaluation Process with INode

Once the function f_θ is learned, the goal is to predict the trajectory u given a specific set of control parameters $\mathbf{c}$. Since the learning process relied on the pretext task, which uses $v(t, \mathbf{c})$, the evaluation process must reconstruct the original trajectory from this learned representation. This process proceeds in two main steps:

1. **Predicting the Implicit Trajectory:** We first predict the implicit trajectory v by solving the ODE defined by the learned function f_θ :

$$\frac{\partial v}{\partial t}(t) = f_\theta(v(t, \mathbf{c}), t, \mathbf{c}) \quad (5.9)$$

Note that, unlike traditional NODE methods, this ODE is not solved during the training phase but is deferred to the evaluation phase. Solving this ODE yields an estimate $\hat{v}$ for the implicit trajectory.

$$\hat{v}(t, \mathbf{c}) = \text{ODESolve}(f_\theta, v(t_0, \mathbf{c}), t_0, t) \quad (5.10)$$

2. **Mapping the Implicit Trajectory to the original Trajectory:** The second step involves re-using the previously learned function f_θ to predict the trajectory u . This prediction is performed at any desired point along the trajectory:

$$\hat{u}(t, \mathbf{c}) = f_\theta(\hat{v}(t, \mathbf{c}), t, \mathbf{c}) \quad (5.11)$$

5.2.4 .Summary of the INode Method

We have formulated a pretext task as a regression problem, mapping an implicit trajectory (the integral) to the original trajectory. Once the regression task is completed, the learned function is applied in two distinct stages to solve the generative process that links control parameters to trajectories: (i) Solving the ODE defined by the learned function to generate the implicit trajectory. (ii) Using the same function to map this implicit trajectory back to the original trajectory.

This division between learning and evaluation allows the INode method to manage the complexity inherent to ODE solvers while capitalizing on the strengths of differential system modeling.

²While it is theoretically possible to exclude the explicit temporal variable t from the input to the neural network in Eq. (5.8)—resulting in an autonomous system valid outside the training interval—this would require the underlying dynamics to be time-independent. In particle accelerator simulations, where the beam behavior varies along the longitudinal axis z , such an assumption is not valid. Therefore, we explicitly maintain the time dependence.

Considering that the integral operator is only an example of potential preprocessing, we introduce more general notations based on some implicit function g whose details are given in Section 5.3.

1. **Preprocessing Step ($\mathcal{F}_g$):** INODE first computes the implicit trajectories v given the original trajectory u : $v = \mathcal{F}_g(u)$. In this section, $\mathcal{F}_g$ is the integral operator.
2. **Learning Step (f_θ):** Use the implicit trajectory v to predict the trajectory u : $\hat{u}(t, \mathbf{c} | v(t, \mathbf{c}))$ and adjust weights θ to minimize the prediction error.
3. **Evaluation Step (G_g):**
 - (a) Predict the implicit trajectory by solving the ODE defined by the neural network: $\hat{v}(\cdot, \mathbf{c}) = \text{ODESolve}(f_\theta, v(t_0, \mathbf{c}), t_0, \cdot)$. In the rest of this chapter, this step is more generally formulated as: $\hat{v}(\cdot, \mathbf{c}) = G_g(f_\theta, \mathbf{c})$.
 - (b) Predict the trajectory: $\hat{u}(t, \mathbf{c}) = f_\theta(\hat{v}(t, \mathbf{c}), t, \mathbf{c})$.

The abovementioned process is visualized in Figure 5.1.

5.3 .INODE Theoretical Foundation

In this section, we develop the theoretical framework supporting the method introduced in Section 5.2. It will be shown that the implicit trajectories at the core of the INODE method, can be defined in a much more general way than the integral representation used to exemplify the approach in Section 5.2. Moreover, we establish a set of sufficient conditions on the preprocessing step that guarantee that a good approximation of the pretext task leads to a good approximation of the overall mapping from control parameters to trajectories.

The proofs of the main theorems and results presented in this section are detailed in Appendix B. Here, we present the first principles establishing the soundness of the INODE approach.

We assume that the trajectories considered are essentially bounded (meaning there exists a bound on their values that holds almost everywhere, except possibly on a set of points with measure zero), measurable functions. This assumption allows for local discontinuities in the trajectories while ensuring they are continuous almost everywhere.

Definition 1 (Observable Trajectory). *Let $u : \begin{cases} [t_0, T] \times \mathbb{R}^p & \rightarrow \mathbb{R}^n \\ t, \mathbf{c} & \rightarrow u(t, \mathbf{c}) \end{cases}$. Function u is an Observable Trajectory if u is L^∞ , that is measurable and essentially bounded.*

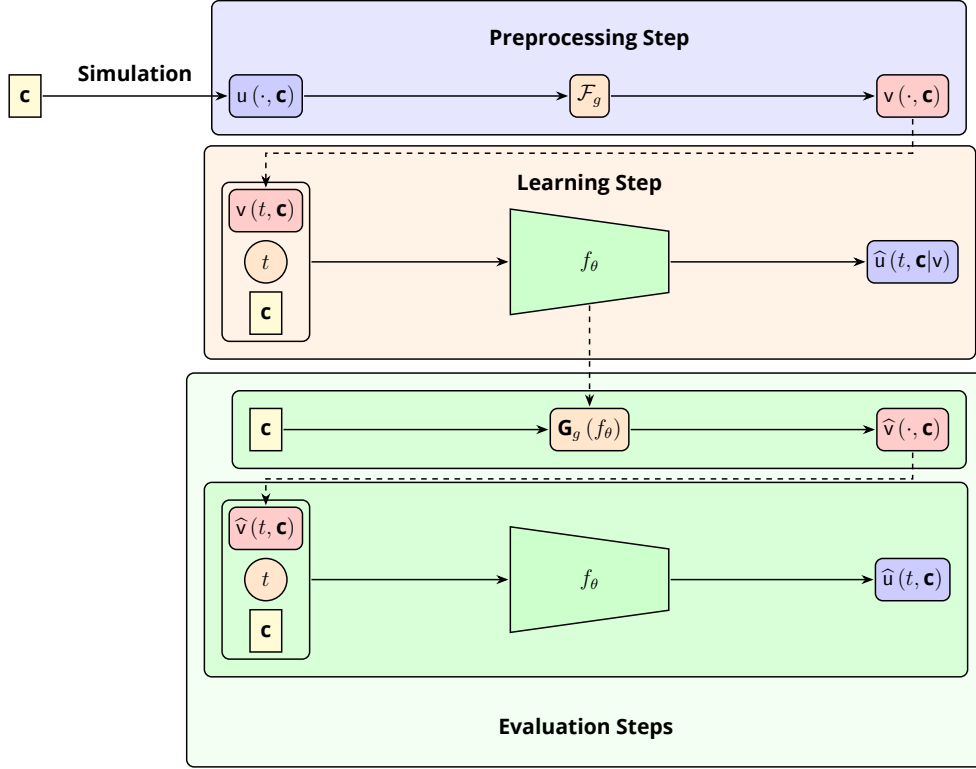


Figure 5.1: **Overview of the INode Method.** The figure illustrates the three main stages of the Implicit Neural ODE (INode) approach for modeling the evolution of beam observables in a particle accelerator. Supposing that we have a database of trajectories $u(t, \mathbf{c})$ resulting from a simulation process taking $\mathbf{c}$ as input, the first stage, the **Preprocessing Step**, takes as input the trajectory $u(\cdot, \mathbf{c})$ resulting from a simulation process and applies a transformation $\mathcal{F}_g$ (like the integral operator) to obtain the implicit trajectory $v(\cdot, \mathbf{c})$. In the **Learning Step**, the neural network f_θ is trained to map the implicit trajectory $v(t, \mathbf{c})$ to the original trajectory $u(t, \mathbf{c})$. The final stage, the **Evaluation Step**, uses the learned function f_θ to (a) predict the implicit trajectory $\hat{v}(t, \mathbf{c})$ by solving the ODE for v , and (b) map $\hat{v}$ to the original trajectory $\hat{u}$. The figure highlights the separation of the ODE solving process from training, which is deferred to the evaluation phase.

5.3.1 .Implicit Trajectories

In the previous sections, we defined the implicit trajectory as the integral of the original observable trajectory. However, this definition can be extended to a broader class of implicit representations. In this section, we outline the conditions necessary for preprocessing observable trajectories to obtain valid implicit trajectories to which the methods described earlier can be applied.

We begin by defining an implicit function g that plays a central role in deriving the implicit trajectory. For the preprocessing step to yield a valid implicit trajectory, this implicit function must possess sufficient smoothness. Specifically, g should be continuous and satisfy Lipschitz continuity conditions in both the observable and implicit spaces. Once these conditions are met, the implicit trajectory can be derived as the solution of the following ordinary differential equation (ODE):

$$\frac{\partial v}{\partial t}(t, \mathbf{c}) = g(u(t, \mathbf{c}), v(t, \mathbf{c}), t), \quad (5.12)$$

We see already that the integral operator presented in the previous section is a specific case where $g(u(t, \mathbf{c}), v(t, \mathbf{c}), t) = u(t, \mathbf{c})$.

To ensure the existence and uniqueness of an implicit trajectory, we impose the following smoothness properties on the implicit function g :

Definition 2 (Implicit Function). Let $g : \begin{cases} \mathbb{R}^n \times \mathbb{R}^m \times [t_0, T] & \rightarrow \mathbb{R}^m \\ \mathbf{x}_u, \mathbf{x}_v, t & \rightarrow g(\mathbf{x}_u, \mathbf{x}_v, t) \end{cases}$. Function g is an implicit function if it satisfied the three properties:

- **Lipschitz Continuity in State Space:** g is uniformly Lipschitz continuous on $\mathbf{x}_u$. That is, for each $\mathbf{x}_v, t \in \mathbb{R}^m \times [t_0, T]$, $\mathbf{x}_u \rightarrow g(\mathbf{x}_u, \mathbf{x}_v, t)$ is Lipschitz continuous and there exists $k_{\mathbf{x}_u}$ such that for all $\mathbf{x}_v \in \mathbb{R}^m$, for all $t \in [t_0, T]$, for all $\mathbf{x}_{u1}, \mathbf{x}_{u2} \in \mathbb{R}^n$, $\|g(\mathbf{x}_{u2}, \mathbf{x}_v, t) - g(\mathbf{x}_{u1}, \mathbf{x}_v, t)\| \leq k_{\mathbf{x}_u} \|\mathbf{x}_{u2} - \mathbf{x}_{u1}\|$.
- **Lipschitz Continuity in Implicit Space:** g is uniformly Lipschitz continuous on $\mathbf{x}_v$. That is, for each $\mathbf{x}_u, t \in \mathbb{R}^n \times [t_0, T]$, $\mathbf{x}_v \rightarrow g(\mathbf{x}_u, \mathbf{x}_v, t)$ is Lipschitz continuous and there exists $k_{\mathbf{x}_v}$ such that for all $\mathbf{x}_u \in \mathbb{R}^n$, for all $t \in [t_0, T]$, for all $\mathbf{x}_{v1}, \mathbf{x}_{v2} \in \mathbb{R}^m$, $\|g(\mathbf{x}_u, \mathbf{x}_{v2}, t) - g(\mathbf{x}_u, \mathbf{x}_{v1}, t)\| \leq k_{\mathbf{x}_v} \|\mathbf{x}_{v2} - \mathbf{x}_{v1}\|$.
- **Continuity over time:** g is continuous on t . That is, for each $\mathbf{x}_u, \mathbf{x}_v \in \mathbb{R}^n \times \mathbb{R}^m$, $t \rightarrow g(\mathbf{x}_u, \mathbf{x}_v, t)$ is continuous.

Existence of the Implicit Trajectory

Given these conditions, we can establish the existence of the implicit trajectory as the solution to the ODE involving g . Specifically:

Theorem 1 (Implicit Trajectory Existence). Let $u : [t_0, T] \times \mathbb{R}^p \rightarrow \mathbb{R}^n$ be a observable trajectory, let g be a implicit function. For any initial condition $v_0(\mathbf{c})$, there exists a unique absolutely continuous function $v : [t_0, T] \times \mathbb{R}^p$, such that for all $\mathbf{c} \in \mathbb{R}^p$, for all $t \in [t_0, T]$, $v(t, \mathbf{c}) = v_0(\mathbf{c}) + \int_{t_0}^t g(u(\tau, \mathbf{c}), v(\tau, \mathbf{c}), \tau) d\tau$.

The core of the proof, detailed in Appendix B, lies in verifying the conditions of Carathéodory's theorem, a generalization of Cauchy-Lipschitz. The conditions imposed on g have been designed to meet these properties.

Definition 3 (Implicit Trajectory). *The unique solution guaranteed by Theorem 1 above is denoted by $\mathcal{F}_{g,v_0}(u)$.*

Examples of Implicit Trajectories

The implicit trajectory described above allows various choices of the implicit function g .

- **Integral Operator** One simple example is the integral operator, for which

$$g(\mathbf{x}_u, \mathbf{x}_v, t) = \mathbf{x}_u \quad (5.13)$$

and $v_0(\mathbf{c}) = 0$. This example yields the integral of the observable trajectory, as discussed earlier. Indeed, the solution to the ODE (5.12) is:

$$v(t, \mathbf{c}) = \int_{t_0}^t u(\tau, \mathbf{c}) d\tau \quad (5.14)$$

- **Exponential Smoothing** Another example is exponential smoothing, where the implicit function is defined as:

$$g(\mathbf{x}_u, \mathbf{x}_v, t) = \lambda(\mathbf{x}_u - \mathbf{x}_v), \quad (5.15)$$

with $\lambda > 0$, which define the implicit trajectory:

$$v(t, \mathbf{c}) = v_0(\mathbf{c}) e^{-\lambda t} + \int_{t_0}^t \lambda e^{-\lambda(t-\tau)} u(\tau, \mathbf{c}) d\tau \quad (5.16)$$

This results in an implicit trajectory that smooths the original trajectory, where the parameter λ controls the degree of smoothing. The advantage of such a preprocessing step is that it simplifies the regression task: the neural network in the pretext task only needs to learn the deviations from this smoother trajectory rather than the full complexity of the original trajectory.

To match the definition of exponential smoothing, we set the initial condition $v_0(\mathbf{c}) = u(t_0, \mathbf{c})$, assuming the initial value of the trajectory is known, which is a valid assumption in the context of particle accelerator simulations.

- **Multiple Exponential Smoothing** A more complex example is multiple exponential smoothing, where we define

$$g(\mathbf{x}_u, \mathbf{x}_{v1}, \mathbf{x}_{v2}, t) = (\lambda_1(\mathbf{x}_u - \mathbf{x}_{v1}), \lambda_2(\mathbf{x}_u - \mathbf{x}_{v2})), \quad (5.17)$$

yielding an implicit trajectory that captures trends at multiple scales.

- **Holt Operator** Finally, the Holt operator, or double exponential smoothing, can be used to handle trajectories with trends. In this case, the implicit space is split into two terms, $\mathbf{x}_v = (\mathbf{x}_s, \mathbf{x}_b)$, representing the smoothed trajectory and the trend, respectively. The implicit function is then defined as:

$$g(\mathbf{x}_u, \mathbf{x}_s, \mathbf{x}_b, t) = \left(\frac{\alpha(1-\beta)}{(1-\alpha)\Delta t} (\mathbf{x}_u - \mathbf{x}_s) + \frac{1}{\Delta t} \mathbf{x}_b, \frac{\alpha\beta}{(1-\alpha)\Delta t} (\mathbf{x}_u - \mathbf{x}_s) \right), \quad (5.18)$$

where α, β , and Δt are smoothing factors and time scaling parameters. The initial condition is set to $v_0(\mathbf{c}) = (u(t_0, \mathbf{c}), 0)$.

These examples illustrate the flexibility of the implicit trajectory concept, allowing us to design operators that simplify the learning task by retaining key information from the original trajectory.

5.3.2 . Universal Approximation Theorem

In this subsection, we establish the theoretical foundation demonstrating that the INODE method effectively approximates any observable trajectory with arbitrary precision.

First, we demonstrate that the INODE method is well-defined by proving the existence of a unique solution for the implicit trajectory. Then, we show that the pretext task falls under the assumptions of the classic Universal Approximation Theorem. Finally, we prove that the INODE method is a universal approximate for any observable trajectory and establish a direct relationship between the accuracy of the pretext task and the final INODE solution.

We recall that the proofs of the main theorems and results presented in this section are detailed in Appendix B.

Existence of the INODE Solution

Following the notation of [37], we define $\mathcal{NN}_{m,n,k}^\rho$ as the set of multi-layer perceptrons with m neurons in the input layer, n neurons in the output layer, and an arbitrary number of hidden layers, each containing at most k neurons, with ρ as the activation function.

Theorem 2 (INODE solution existence). *Let g be an implicit function and f_θ be a multi-layer perceptron. For any initial condition $v_0(\mathbf{c})$, there exists a unique function v such that for all $t \in [t_0, T]$, for all $\mathbf{c} \in \mathbb{R}^p$,*

$$\begin{cases} \frac{\partial v}{\partial t}(t, \mathbf{c}) = g(f_\theta(v(t, \mathbf{c}), t, \mathbf{c}), v(t, \mathbf{c}), t) \\ v(t_0, \mathbf{c}) = v_0(\mathbf{c}) \end{cases}$$

Proof. Let $\mathbf{c} \in \mathbb{R}^p$

Here we can apply the Picard-Lindelöf Theorem (see Appendix B) by verifying the continuity of the right-term of the ODE.

1. *Continuity.*

$\forall \mathbf{x}_v \in \mathbb{R}^m, t \rightarrow g(f_\theta(\mathbf{x}_v, t, \mathbf{c}), \mathbf{x}_v, t)$ is continuous as a composition of continuous functions.

2. *Lipschitz Continuity.*

Let $(t, \mathbf{x}_{v,1}, \mathbf{x}_{v,2}) \in [t_0, T] \times \mathbb{R}^m \times \mathbb{R}^m$.

$$\begin{aligned}
& \|g(f_\theta(\mathbf{x}_{v,2}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,1}, t)\| \\
&= \|g(f_\theta(\mathbf{x}_{v,2}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) + g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,1}, t)\| \\
&\leq \|g(f_\theta(\mathbf{x}_{v,2}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,2}, t)\| + \|g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,1}, t)\| \\
&\leq k_{\mathbf{x}_u} \|f_\theta(\mathbf{x}_{v,2}, t, \mathbf{c}) - f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c})\| + k_{\mathbf{x}_v} \|\mathbf{x}_{v,2} - \mathbf{x}_{v,1}\| \\
&\leq k_{\mathbf{x}_u} k_{f_\theta} \|\mathbf{x}_{v,2} - \mathbf{x}_{v,1}\| + k_{\mathbf{x}_v} \|\mathbf{x}_{v,2} - \mathbf{x}_{v,1}\| \\
&= (k_{\mathbf{x}_u} k_{f_\theta} + k_{\mathbf{x}_v}) \|\mathbf{x}_{v,2} - \mathbf{x}_{v,1}\|
\end{aligned}$$

We can apply the Picard-Lindelöf Theorem to conclude. $\square$

This theorem ensures that the INODE method defines a well-posed estimator for any given observable trajectory.

Definition 4. *The unique solution guaranteed by Theorem 2 is denoted by $\mathbf{G}_{g, \mathbf{v}_0}(f_\theta, \mathbf{c})$.*

Universal Approximation of the Pretext Task

Next, we examine the approximation capabilities of the pretext task.

Theorem 3 (Universal Approximation of the Pretext Task). *Let u be an observable trajectory and g be a implicit function. Let v be the implicit trajectory of u guaranteed by Theorem 1: $v = \mathcal{F}_{g, \mathbf{v}_0}(u)$.*

For all $\varepsilon > 0$, for all $k > 0$, there exists $f_\theta \in \mathcal{NN}_{m+p+1, n, \max(m+n+2, n)}^{\text{ReLU}}$ such for all $t, \mathbf{c} \in [t_0, T] \times \mathbb{R}^p$, $\|f_\theta(v(t, \mathbf{c}), t, \mathbf{c}) - u(t, \mathbf{c})\| < \varepsilon$, and f_θ is a k -uniformly Lipschitz continuous over the implicit space.

The proof is based on the application of the universal approximation theorem of [48]. See Appendix B for a detailed proof.

This result establishes that a multi-layer perceptron can approximate the pretext task with any arbitrary precision. The universal approximation property is a critical element, as it guarantees that the INODE method can accurately model the relationship between the implicit and observable trajectories.

Furthermore, this theorem also states that we can find a neural network with an arbitrary value of the Lipschitz constant over the implicit space. This result is important for the link between the accuracy of the pretext task and the accuracy of the INODE method, as detailed in the following paragraphs.

Universal Approximation by INODE

Finally, we extend the universal approximation property to the entire INODE method.

Theorem 4 (INODE Universal Approximation). *Let u be an observable trajectory and g be an implicit function. Let v be the implicit trajectory of u corresponding to the solution guaranteed by Theorem 1: $v = \mathcal{F}_{g,v_0}(u)$.*

For all $\varepsilon > 0$, there exists $\varepsilon_1 > 0$ such that all $f_\theta \in \mathcal{NN}_{m+p+1,n,\max(m+p+2,n)}^{\text{ReLU}}$ verifying: for all $t, \mathbf{c} \in [t_0, T] \times \mathbb{R}^p$, $\|f_\theta(v(t, \mathbf{c}), t, \mathbf{c}) - u(t, \mathbf{c})\| < \varepsilon_1$, also verifies: $\|f_\theta(\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - u(t, \mathbf{c})\| < \varepsilon$.

Furthermore, if we denote by $k_{\mathbf{x}_u}$ the Lipschitz constant of the implicit function g over the Observable Space, and by k the Lipschitz constant of the neural network over the implicit trajectory variable, we also have:

$$\|f_\theta(v(t, \mathbf{c}), t, \mathbf{c}) - u(t, \mathbf{c})\| < \varepsilon_1 \Rightarrow \|f_\theta(\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - u(t, \mathbf{c})\| < (1 + k k_{\mathbf{x}_u}(T - t_0)) \varepsilon_1$$

Proof. Let $\varepsilon_1 > 0$, $k > 0$. Theorem 3 states that there exists at least a neural network f_θ satisfying the above properties. Let f_θ be a neural network satisfying the above properties.

$$\begin{aligned} \|g(f_\theta(v(t, \mathbf{c}), t, \mathbf{c}), v(t, \mathbf{c}), t) - g(u(t, \mathbf{c}), v(t, \mathbf{c}), t)\| &\leq k_{\mathbf{x}_u} \|f_\theta(v(t, \mathbf{c}), t, \mathbf{c}) - u(t, \mathbf{c})\| \\ &\leq k_{\mathbf{x}_u} \varepsilon_1 \end{aligned}$$

Integrating the term $g(f_\theta(v(t, \mathbf{c}), t, \mathbf{c}), v(t, \mathbf{c}), t) - g(u(t, \mathbf{c}), v(t, \mathbf{c}), t)$, the previous inequality can be rewritten as follows:

$$\begin{aligned} \|\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}) - v(t, \mathbf{c})\| &= \left\| \int_{t_0}^t g(f_\theta(v(\tau, \mathbf{c}), \tau, \mathbf{c}), v(\tau, \mathbf{c}), \tau) - g(u(\tau, \mathbf{c}), v(\tau, \mathbf{c}), \tau) d\tau \right\| \\ &\leq \int_{t_0}^t \|g(f_\theta(v(\tau, \mathbf{c}), \tau, \mathbf{c}), v(\tau, \mathbf{c}), \tau) - g(u(\tau, \mathbf{c}), v(\tau, \mathbf{c}), \tau)\| d\tau \\ &\leq (t - t_0) k_{\mathbf{x}_u} \varepsilon_1 \\ &\leq (T - t_0) k_{\mathbf{x}_u} \varepsilon_1 \end{aligned}$$

Finally, using f_θ for the final estimation, we have

$$\begin{aligned} &\|f_\theta(\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - u(t, \mathbf{c})\| \\ &= \|f_\theta(\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - f_\theta(v(t, \mathbf{c}), t, \mathbf{c}) + f_\theta(v(t, \mathbf{c}), t, \mathbf{c}) - u(t, \mathbf{c})\| \\ &\leq \|f_\theta(\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - f_\theta(v(t, \mathbf{c}), t, \mathbf{c})\| + \|f_\theta(v(t, \mathbf{c}), t, \mathbf{c}) - u(t, \mathbf{c})\| \\ &\leq k \|\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}) - v(t, \mathbf{c})\| + \varepsilon_1 \\ &\leq (k(T - t_0) k_{\mathbf{x}_u} + 1) \varepsilon_1 \end{aligned}$$

Choosing $\varepsilon_1 = \frac{\varepsilon}{1 + k(T - t_0) k_{\mathbf{x}_u}}$, we conclude the proof. $\square$

This theorem proves two important results. First, it establishes that the INODE method can approximate any observable trajectory to a desired precision, provided the pretext task is solved with sufficient accuracy. Second, it relates the accuracy of the pretext task to the accuracy of the final INODE solution, showing that the error in the INODE solution grows proportionally with the elapsed time, the Lipschitz constant of the neural network, and the Lipschitz constant of the implicit function. Specifically:

- The longer the trajectory, the greater the potential loss of precision in the INODE solution.
- Controlling the Lipschitz constant of the neural network, for example, through techniques like spectral normalization, can improve the overall solution accuracy.
- The implicit function should be chosen carefully to minimize its Lipschitz constant and thus improve the approximation quality.

5.4 .INODE Experiments

In this section, we present preliminary results that evaluate the effectiveness of the INODE method in modeling the dynamics of a particle accelerator. For the sake of simplicity, we exclude from the analysis any particle beams that do not reach the end of the Linac. The experiments are conducted in two stages: first, we focus on the first 1.4 meters of the Linac; next, we extend the analysis to the entire length of the Linac, which spans 9.393 meters.

The version of the INODE method examined in this section incorporates a neural network with two hidden layers, each containing 256 units. The preprocessing step involves stacking two exponential smoothing operations with distinct smoothing parameters ($\lambda_1 = 0.001$ and $\lambda_2 = 1.0$).

Performances are assessed using the R^2 score, defined at every step across the database.

Initial Evaluation on the First Section of the Linac

We begin by evaluating INODE's ability to predict the evolution of beam observables over the first 1.4 meters of the Linac. The focus here is on assessing whether the method can capture the underlying dynamical system governing the accelerator's behavior, especially in the presence of discontinuities in the data. As shown in Figure 5.2, INODE shows decent performance to model the observable trajectories for this initial section, especially for $z \geq 0.3\text{m.}$. The extra difficulties at the beginning of the accelerator are blamed on the fact that the first 25 cm enable the beam to reach a fraction of the light speed.

The variability in performance highlights the heterogeneity of the physical phenomenon, suggesting a model structure that could provide greater flexibil-

ity—such as a U-Net architecture or attention mechanisms, which allow for variable weights depending on surrounding context.

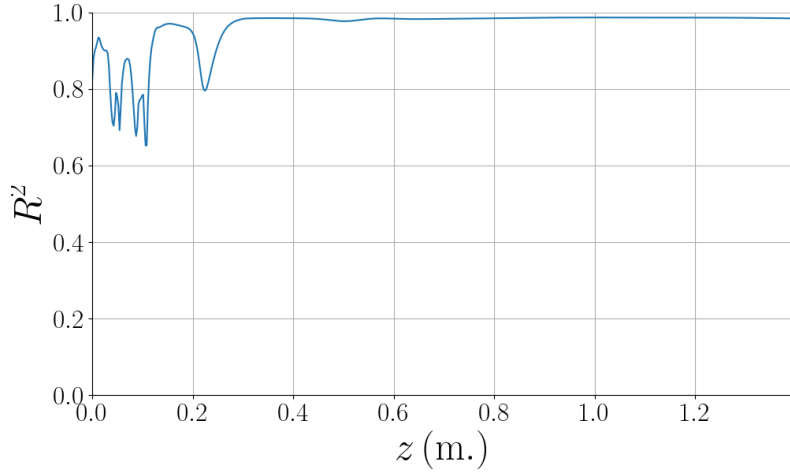


Figure 5.2: Performance of INODE in the prediction of the observables in the 1.4 first meters of the Linac.

Comparison with MLP and NODE Modeling

To benchmark INODE's performance, we compare it with a simpler approach where a multilayer perceptron (MLP) directly predicts the observable trajectory $\mathbf{u}(t, \mathbf{c})$ given the time t and the control parameters $\mathbf{c}$. We use the same underlying architecture as the one used for INODE. Surprisingly, as shown in Figure 5.3 despite the theoretical advantages of INODE in handling complex dynamical systems, the MLP achieves comparable results on this dataset.

We also compare INODE's performance to a NODE-like approach where the neural network aims at learning the derivative of the trajectory. The INODE's performance is also comparable with the NODE-like approach. However, the NODE-like approach is more computationally intensive with a total training time exceeding 10 hours compared to the 20 minutes required for the INODE version.

This suggests that the improvements that INODE can make in the pretext task do not necessarily translate into a significant performance gain in the final evaluation compared to the MLP.

Evaluation on the Entire Linac

Next, we extend the evaluation to the entire 9.393 meters of the Linac. As shown in Figure 5.4, in this case, INODE struggles to maintain accurate predictions for the full length of the Linac. While it handles the early sections well, its performance deteriorates as the beam progresses further through the

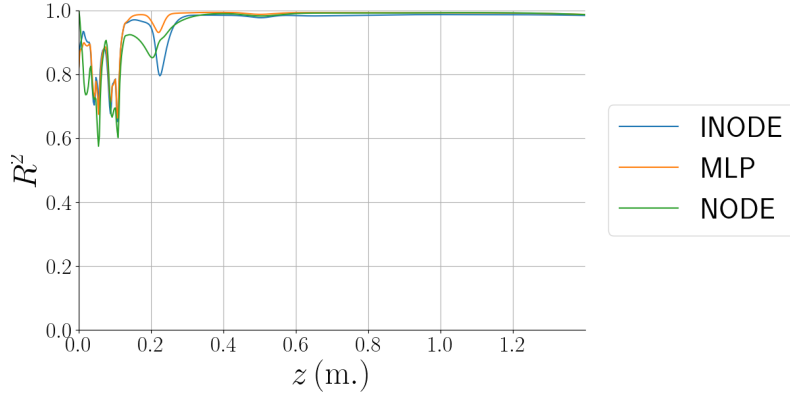


Figure 5.3: Comparison of INODE with an MLP and a NODE-like approach in the prediction of the observables in the 1.4 first meters of the Linac.

accelerator. The complexity of the system increases over longer trajectories, and INODE cannot generalize effectively to the end of the Linac. This may be due to the cumulative effect of prediction errors over time and the growing influence of system dynamics that were not adequately captured by the implicit pretext task.

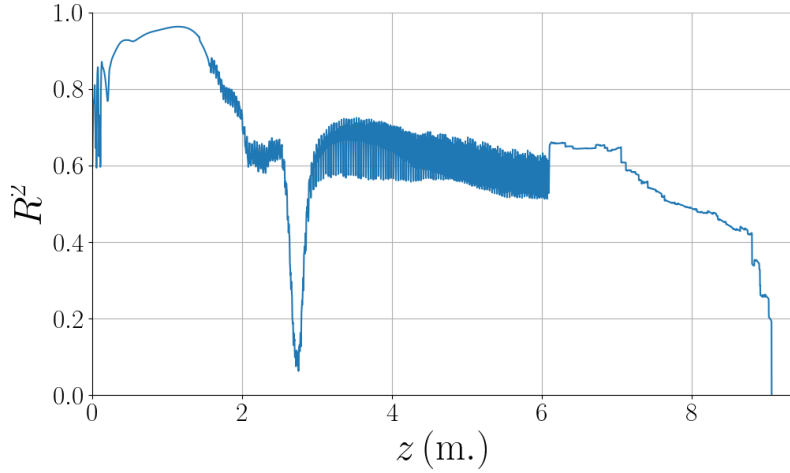


Figure 5.4: Performance of INODE in the prediction of the observables of the Linac.

Summary of Results

In summary, the preliminary results show that while INODE can model the evolution of beam observables over short sections of the Linac, it faces challenges in extending these predictions over the entire length of the accelerator. Additionally, the comparison with a direct MLP approach reveals that, despite its sophisticated framework, INODE behaves on par with the MLP and NODE

approach in this case.

On the positive side, the soundness and generality of the approach is rigorously established under mild condition. On the practical side, some empirical limitations are identified and discussed with respect to the complexity of the underlying physical phenomenon.

5.5 .Conclusion and Future Works

The INODE approach introduced in this chapter presents an original framework for modeling ODEs that can accommodate discontinuities in target trajectories. A theoretical analysis of this framework is provided to establish the method's validity.

Experimental validation demonstrates that the approach performs comparably to both MLP and NODE-like methods. An analysis of the approach's limitations suggests that these may stem from the inherent heterogeneity of the underlying physical phenomenon.

Feasibility analysis: the Lyapunov exponents

One of the key challenges encountered in applying INODE is its difficulty maintaining accuracy over longer trajectories, particularly as predictions extend to the end of the linac. This raises an important question: Is the underlying dynamical system even learnable with the current approach? One way to assess this is through the use of Lyapunov exponents.

Lyapunov exponents provide a way to measure the sensitivity of a system to initial conditions. In the context of INODE , these exponents could help determine whether the system's complexity makes it feasible to learn the dynamics accurately. A large positive Lyapunov exponent would indicate chaotic behavior, suggesting that minor errors in prediction could exponentially amplify over time, making long-term prediction inherently challenging. Using Lyapunov exponents as a diagnostic tool could help judge whether the problem is feasible for the INODE method to learn or whether alternative modeling approaches may be necessary.

Extending the INODE Method to Delayed Differential Equations.

One promising extension is to incorporate delayed differential equations (DDEs). Many real-world systems, including particle accelerators, exhibit delayed interactions or feedback mechanisms that standard ODEs cannot capture. By extending the current INODE formulation to handle DDE, we can use multiple past values of the implicit trajectory in the pretext task, enriching the implicit space without increasing the Lipschitz constant of the preprocessing step.

Exploring Alternative Operators

Another direction for improvement involves experimenting with alternative operators in the preprocessing step. The goal is to find an informative operator to significantly improve the performance of the pretext task with a low Lipschitz constant to prevent these improvements from being lost during the evaluation phase.

Conclusion

The INODE method provides a novel way to model particle accelerators by leveraging implicit trajectories, but its limitations, particularly in long-term prediction, highlight the need for further refinement. Using Lyapunov exponents to assess system learnability, extending the framework to handle delayed dynamics, and exploring alternative operators are all promising future directions.

6 -Conclusion

Contents

6.1 Context and Challenges of Particle Accelerator Simulation	110
6.2 Contributions	110
6.2.1 LINACNET	110
6.2.2 INODE	111
6.3 Limitations and Future Work	111
6.3.1 Applicability of the Surrogate Models	111
6.3.2 Particle Cloud Representation	111
6.3.3 Computation Intensive Device Requirements	112
6.3.4 Looking for more adapted operators for INODE	112
6.3.5 Experimental Validation of INODE	112
6.3.6 Merging INODE and LINACNET	112

The work presented in this manuscript addresses critical challenges in particle accelerator simulation by developing sophisticated surrogate models. Particle accelerators are complex systems that rely heavily on detailed simulations to guide their design, optimization, and operation. However, the computational cost of running such simulations is significant, limiting their real-time application and necessitating exploring alternative modeling techniques.

In this context, surrogate models offer a promising solution, enabling faster and more efficient approximations of particle accelerator behavior. This conclusion summarizes the main contributions of this work, mainly focusing on two key models—LINACNET and INODE—that were developed to tackle specific challenges in particle accelerator simulation.

The chapter begins by revisiting the context and challenges that motivated this research (see Section 6.1), followed by a discussion of the significant contributions made (see Section 6.2). The limitations of the current models are then considered, alongside potential directions for future research aimed at improving both the accuracy and practicality of surrogate models in particle accelerator applications (see Section 6.3).

6.1 .Context and Challenges of Particle Accelerator Simulation

As outlined in Chapter 1, particle accelerator simulations are critical throughout the lifecycle of an accelerator, providing detailed insights into beam dynamics and overall system performance. These simulations often rely on advanced techniques like the Particle-in-Cell (PIC) method to model complex interactions between particles and electromagnetic fields, particularly when considering space charge effects. Although these simulations are essential, they are also resource-intensive, demanding significant amounts of time and computational power. This requirement poses challenges for real-time applications and extensive optimization efforts, highlighting the importance of developing efficient alternatives. Surrogate models emerge as a viable solution, offering a way to approximate these complex simulations without compromising on accuracy.

This computational bottleneck opens up the need for surrogate models, which strive to replicate the results of detailed simulations at a fraction of the cost. Surrogate models provide a promising alternative by learning to approximate the behavior of particle accelerator simulators based on past simulations, delivering rapid predictions that can significantly speed up the design and optimization processes. Their ability to generalize across a broad range of parameter space enables the exploration of different accelerator configurations in a considerably reduced timeframe.

The foundation of these simulations lies in the propagation of a particle cloud through the various components of the accelerator. This highly detailed beam representation allows for extracting key observables, such as beam size, energy spread, and emittance, which are critical for assessing the performance of the accelerator. For a surrogate model to be effective, it must be able to predict the detailed behavior of the particle cloud and the corresponding observables accurately. These challenges raise fundamental questions about the feasibility of surrogate modeling: Is it possible to accurately predict the entire evolution of a particle cloud, and can the underlying equations that govern its propagation be learned effectively?

6.2 .Contributions

This thesis presents two major contributions to the field of surrogate modeling for particle accelerator simulations: `LINACNET` and `INODE`. Both models were developed to address the core challenges identified in particle accelerator simulations, particularly the need to model the detailed behavior of particle clouds and the underlying equations governing beam propagation.

6.2.1 .LINACNET

`LINACNET`, inspired by the PointNet architecture, is specifically designed to model the particle beam as a dynamic particle cloud. By leveraging the detailed structure of the particle cloud, `LINACNET` can predict the evolution of

the entire beam, offering a comprehensive and rich set of predictions. One of the key strengths of LINACNET lies in its ability to produce internally consistent predictions. Observables such as beam size and emittance are directly derived from the evolution of the particle cloud, ensuring that all predicted properties are coherent.

Furthermore, one of the key strengths of LINACNET is its ability to not only predict the beam's evolution from its initial state but also provide consistent and accurate predictions from any intermediate state of the beam. This ensures that LINACNET maintains a high level of consistency across the entire simulation, making it highly reliable for modeling beam dynamics throughout different stages of its propagation within the accelerator.

6.2.2 .INODE

The second significant contribution of this thesis is INODE, which introduces a novel approach to learning dynamic systems. A major advantage of INODE is that it externalizes the ODE solver from the training process, allowing the model to focus on learning the underlying dynamics more efficiently. In addition, INODE handles systems with discontinuous trajectories by transforming these trajectories into a smoother, continuous form, guaranteeing the feasibility of the training process.

6.3 .Limitations and Future Work

While this thesis highlights the potential of surrogate models such as LINACNET and INODE for particle accelerator simulations, it also identifies several limitations that present opportunities for future research and development.

6.3.1 .Applicability of the Surrogate Models

Although the surrogate models developed in this work have demonstrated promising results based on specific performance indicators, their application in real-world scenarios remains to be tested. These models still require validation in contexts such as real-time optimization of an operational accelerator, uncertainty quantification of control parameters, or other practical tasks that particle accelerator operators frequently encounter. Ensuring these models can perform reliably in operational environments with diverse beam configurations and varying conditions will be an essential next step in their development.

6.3.2 .Particle Cloud Representation

LINACNET employs PointNet as the foundational architecture for representing the particle cloud. However, recent advances in particle cloud representation may offer more efficient and powerful alternatives. These newer architectures could improve the model's accuracy and efficiency, reducing the computational resources needed and potentially offering better scalability.

Future research could explore integrating these state-of-the-art methods to enhance further LINACNET's ability to model complex beam dynamics.

6.3.3 .Computation Intensive Device Requirements

A significant limitation of LINACNET lies in its reliance on high-end hardware for training, specifically the most recent GPUs (such as the A100 with 80GB of memory). This requirement challenges wider adoption, as not all research institutions or facilities have access to such resources. To address this issue, future research could explore pre-learning a representation of the particle cloud using an autoencoder. Compressing the particle cloud into a lower-dimensional space may make it possible to work with a more manageable representation, reducing the computational burden without sacrificing accuracy.

6.3.4 .Looking for more adapted operators for INode

The INode performance is limited by the Lipschitz constant of its underlying operator that still needs to be informative to bring value to the whole method. Future research could explore other operators that would have a better tradeoff between informativeness and its Lipschitz constant.

6.3.5 .Experimental Validation of INode

Despite its theoretical foundation, INode still needs to demonstrate significant improvements over traditional methods for learning the underlying equations governing beam propagation. The current work relies on transformations like integral and exponential smoothing to stabilize learning, but these approaches have yet to lead to superior performance. Future research should explore alternative transformations that strike a better balance between capturing critical information and ensuring stability in the learning process.

6.3.6 .Merging INode and LINACNET

The potential for combining LINACNET and INode represents an exciting avenue for future research. While LINACNET provides a detailed representation of the particle cloud, INode models the evolution of the important observables along the accelerator. One possible direction for future work is to merge these two approaches, enabling the learning of the continuous evolution of the particle cloud. Another promising avenue is developing a joint learning framework where both models are trained together, with enforced consistency.

A -Sampling Strategies in Surrogate Modeling

Effective surrogate modeling relies on the preliminary selection of data points for evaluating the high-fidelity simulator. The chosen sampling method significantly influences the accuracy and efficiency of the surrogate model.

A.1 .Random Sampling

Random sampling involves selecting data points independently across the input space according to a predefined distribution. While straightforward and easy to implement, this method can lead to uneven coverage, resulting in clusters of points and gaps. These irregularities reduce the surrogate model's accuracy because clustered points provide redundant information, while gaps leave regions of the input space poorly represented [23].

The challenge of achieving uniform coverage with random sampling becomes more pronounced in high-dimensional spaces. This issue is quantified by the concept of discrepancy, which measures the deviation of samples from a perfectly uniform distribution.

Mathematically, let $P = \{x_1, x_2, \dots, x_n\}$ be a set of n points in the d -dimensional unit hypercube $[0, 1]^d$. The discrepancy D is defined as:

$$D = \sup_{B \in \mathcal{J}} \left| \frac{1}{n} \sum_{i=1}^n \mathbb{1}_B(x_i) - \lambda_d(B) \right|,$$

where $\mathcal{J}$ is the set of all axis-aligned subintervals B of $[0, 1]^d$, $\mathbb{1}_B$ is the indicator function of B , and $\lambda_d(B)$ is the Lebesgue measure (volume) of B . A high discrepancy indicates poor uniformity, which can occur even with a large number of samples, particularly in high-dimensional spaces.

The star discrepancy D_n^* is a specific form of discrepancy where the supremum is taken over all subintervals B of the form $[0, \mathbf{t}) = \prod_{i=1}^d [0, t_i)$ with $\mathbf{t} = (t_1, t_2, \dots, t_d) \in [0, 1]^d$. It is defined as:

$$D_n^* = \sup_{\mathbf{t} \in [0, 1]^d} \left| \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{[0, \mathbf{t})}(x_i) - \prod_{i=1}^d t_i \right|.$$

The general discrepancy D_n can be bounded by the star discrepancy D_n^* through the inequality:

$$D_n \leq D_n^* \leq 2^d D_n.$$

For random sampling, the expected star discrepancy $\mathbb{E}[D_n^*]$ is given by [39]:

$$\mathbb{E}[D_n^*] = O\left(\sqrt{\frac{\log \log n}{n}}\right).$$

A.2 .Quasi-Monte Carlo Methods

Quasi-Monte Carlo methods use deterministic sequences to achieve more uniform input space coverage than random sampling. These methods generate low-discrepancy sequences, where discrepancy measures the deviation from a perfectly uniform distribution.

Latin Hypercube Sampling (LHS) divides each input dimension into k equal intervals and ensures that each interval is sampled exactly once. This stratification ensures more uniform coverage compared to random sampling [44].

Other low-discrepancy sequences include Sobol sequences [67] or Hammersley sequences [27].

The discrepancy of these low-discrepancy sequences can be estimated as:

$$D_n = O\left(\frac{(\log n)^d}{n}\right).$$

This expression shows a significant improvement over random sampling in the convergence rate. However, due to the exponential dependence on the dimension d , this advantage only becomes apparent for very large values of n in moderate to high-dimensional problems.

A.3 .Active Learning

Active learning is an iterative process to select the most informative data points for evaluation, thereby improving model accuracy with fewer training samples. This approach is particularly valuable when evaluating the simulator, which is costly. The primary goal of active learning is to maximize the information gained from each evaluated sample.

Formally, let D be the dataset and $\mathcal{M}$ be the model. At each iteration, the goal is to select a new point x that maximizes the information gain. This can be formalized as:

$$x^* = \arg \max_{x \in \mathcal{X}} I(x; \mathcal{M}),$$

where $I(x; \mathcal{M})$ denotes the information gain from adding x to the dataset for model $\mathcal{M}$.

Active learning uses different strategies to select the most informative points. Here, we list the most commonly used strategies.

- **Uncertainty Sampling:** Uncertainty sampling selects data points where the model is most uncertain about the output [64]. Mathematically, if $\sigma(x)$ denotes the prediction uncertainty at point x , uncertainty sampling aims to select points where $\sigma(x)$ is maximized:

$$x^* = \arg \max_x \sigma(x).$$

- **Query-by-Committee (QBC):** QBC [65] involves training a committee of models on the current dataset. The points where these models disagree the most are selected for evaluation. Let $\{M_1, M_2, \dots, M_k\}$ be a committee of models, and x be a data point. The disagreement among the models can be quantified using a measure such as variance or entropy. The point x with the highest disagreement is selected for evaluation:

$$x^* = \arg \max_x \text{Var}(\{M_i(x)\}_{i=1}^k).$$

Alternatively, entropy can be used as a measure of disagreement:

$$x^* = \arg \max_x \sum_{i=1}^k p_i(x) \log p_i(x),$$

where $p_i(x)$ is the probability assigned by model i to the prediction at x .

- **Expected Error Reduction** Expected error reduction [56] focuses on selecting points that are anticipated to reduce the overall prediction error of the model the most. For each potential new point x , the expected reduction in error can be estimated, and the point with the maximum expected error reduction is selected:

$$x^* = \arg \max_x \mathbb{E}[\Delta \text{Error}(x)],$$

where $\Delta \text{Error}(x)$ represents the decrease in prediction error resulting from adding x to the training set.

While active learning offers significant advantages in terms of data efficiency, it also has several limitations. Here are a few of them.

- **Computational Cost:** Each iteration can be computationally expensive, requiring retraining the model and recalculating the acquisition function for each new data point. This can be particularly burdensome with complex models.
- **Dependence on Initial Model:** The effectiveness of active learning heavily depends on the initial model and acquisition function. The learning process may only converge to an optimal solution if the initial model is good or the acquisition function is well-chosen.
- **Exploration-Exploitation Trade-off:** Another challenge is balancing exploration (selecting points in less known regions) and exploitation (selecting points in regions likely to improve model accuracy). Poor balancing can lead to suboptimal learning efficiency, especially in high-dimensional spaces.

- **Sampling Bias:** Dasgupta [16] highlights that active learning can introduce sampling bias, where the selected samples may not represent the overall data distribution. This bias occurs because active learning tends to select informative points that are often near the decision boundary of the current model, which may not reflect the true underlying distribution of the data.

B -INODE Mathematical Formalism

This section comprehensively overviews the mathematical foundations underlying the INODE method and the proof of a universal approximation theorem. We begin by giving the theoretical background of Initial Value Problems (IVP) concerning discontinuous Ordinary Differential Equations (ODEs) based on notation and results from [68] (refer to Section B.1). Next, we present proof of the existence and uniqueness of the solution provided by the INODE method (see Section B.2.1). Finally, we show a universal approximation theorem for the INODE method that applies to all essentially bounded, measurable trajectories (see Section B.2.2).

In the remainder of this section, for simplicity's sake, we will use the following notions:

- $\mathcal{I}, \mathcal{J}$ are intervals in $\mathbb{R}$.
- $\|\cdot\|$ is the Euclidean norm.
- $\eta : \mathcal{I} \rightarrow \mathbb{R}^n$ is a function of time.
- $\omega : \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_2}$ is a vector function.
- $\zeta : \mathcal{I} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ defines an ODE.
- $\xi : \mathcal{J} \rightarrow \mathbb{R}^n$ is a solution to the above ODE ζ .
- $\mathbf{x} \in \mathbb{R}^n$ denotes a state of the solution (that is values $\xi(t)$).

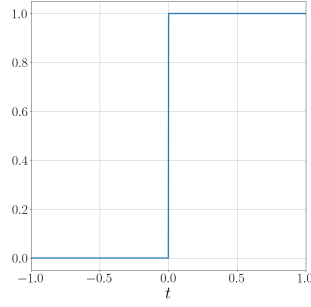
B.1 .Background

This section focuses on established results from Lebesgue measure theory and the theory of Ordinary Differential Equations. These results extend the traditional Picard-Lindelöf theorem to include non-continuous ODEs with respect to time, as long as they remain locally integrable. For a more detailed introduction, we refer the reader to [68].

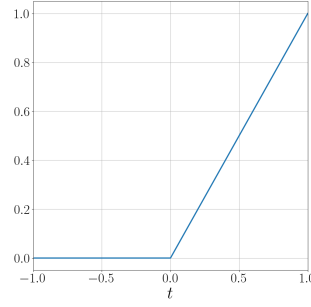
As an illustrating example (see [14]), let us consider the following ODE:

$$\begin{cases} \frac{d\xi}{dt}(t) = H(t) \\ \xi(0) = 0 \end{cases} \quad (\text{B.1})$$

where $H(\cdot)$ is the Heaviside function. As the right-hand side of this equation is discontinuous, neither the conditions of the Picard-Lindelöf theorem, nor those of the Peano theorem are satisfied. However, the ReLU function is a



Heaviside function



ReLU function

Figure B.1: $\xi(t) = \text{ReLU}(t)$ when $\frac{d\xi}{dt}(t) = H(t)$ with $\xi(0) = 0$.

good candidate to be a solution to this problem (see Figure B.1), with the unique problem of its non-differentiability in 0. The following definitions tackle the issue of discontinuous right-hand sides in differential equations, setting the stage for stating Carathéodory's existence theorem.

Definition 5 (Absolute Continuity). *A function $\eta: \mathcal{I} \rightarrow \mathbb{R}^n$ is said to be absolutely continuous if it satisfies the following property: for each $\varepsilon > 0$ there is a $\delta > 0$ such that, for every k and for every finite sequence of sub-segment of $\mathcal{I}$, $([a_i, b_i])_{i \in \llbracket 1, k \rrbracket}$,*

$$\text{if } \sum_{i=1}^k (b_i - a_i) < \delta, \text{ then } \sum_{i=1}^k \|\eta(b_i) - \eta(a_i)\| < \varepsilon.$$

Note that every absolutely continuous function is continuous. However, the converse is not true. A well-known counterexample is the function $\eta: \mathbb{R} \rightarrow \mathbb{R}$ where $\eta(t) = \begin{cases} 0, & \text{if } t = 0 \\ t \sin(1/t), & \text{otherwise} \end{cases}$ which is continuous even at 0 since $\lim_{t \rightarrow 0} \eta(t) = 0$. It is not absolutely continuous because it lacks a derivative at 0. The derivative $\frac{d\eta(t)}{dt} = \sin(1/t) - \frac{1}{t} \cos(1/t)$ becomes undefined as t tends to 0. Let's focus on the behavior of the function near 0 (See Figure B.2). The function oscillates as t tends to 0 from the positive side due to the $t \sin(1/t)$ term, doing so infinitely many times around 0. The oscillatory behaviour of this function close to 0 prevents it from having bounded difference of values on any interval that includes 0. We can point out that this function is absolutely continuous on an interval not containing 0.

Moreover, the definition of absolute continuity can be rephrased by using the concept of derivative.

Property 1 (Equivalent Definition). *The following properties are equivalent:*

- function $\eta: \mathcal{I} \rightarrow \mathbb{R}^n$ is absolutely continuous;
- For almost all $t \in \mathcal{I}$, η has a derivative $\frac{d\eta}{dt}$ that is Lebesgue integrable such

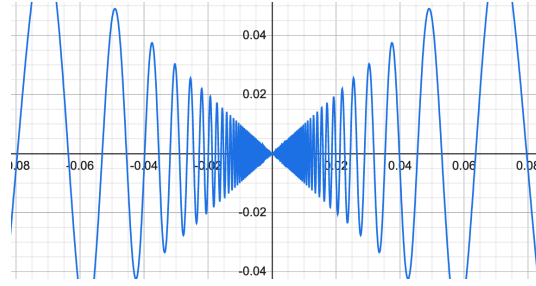


Figure B.2: Behavior of function $\eta(t)$ where $\eta(t) = (1/t) \sin(1/t)$ if $t \neq 0$ and $\eta(0) = 0$.

that:

$$\eta(t) = \eta(t_0) + \int_{t_0}^t \frac{d\eta}{dt}(\tau) d\tau$$

We require a definition weaker than absolute continuity, focusing specifically on functions that demonstrate absolute continuity across all closed intervals of the domain.

Definition 6 (Local Absolute Continuity). *A function $\eta: \mathcal{I} \rightarrow \mathbb{R}^n$ is said to be locally absolutely continuous, if for each sub-segment $[a, b]$ of $\mathcal{I}$, the restriction $\eta|_{[a,b]}$ of η to $[a, b]$ is absolutely continuous.*

Note that the function $\eta: (0, \infty) \rightarrow \mathbb{R}$ where $\eta(t) = \begin{cases} 0, & \text{if } t = 0 \\ t \sin(1/t), & \text{otherwise} \end{cases}$ is locally absolutely continuous.

We will also recall the definition of Local Integrability (a function that is integrable and finite on every compact subset of its domain of definition). More formally,

Definition 7 (Local Integrability). *Let $\eta: \mathcal{I} \rightarrow \mathbb{R}^n$ be a Lebesgue measurable function. η is said to be locally integrable if, for each $a, b \in \mathcal{I}$, $\int_a^b \|\eta(\tau)\| d\tau < \infty$.*

Note that functions with good properties (such as being continuous or integrable) are locally integrable. The function $f(t) = \frac{1}{\sqrt{t}}$ defined on the interval $(0, 1)$ is locally integrable. Note that this function becomes unbounded as t tends 0. However, since local integrability only requires the function to be integrable over compact subsets within $(0, 1)$, such as $[a, b]$ where $0 < a < b < 1$, this unbounded behavior close to 0 does not affect the local integrability over $(0, 1)$.

Functions we will consider are bounded almost everywhere, meaning they're bounded except on a set of points with measure zero. To formalize this:

Definition 8 (Essential Boundedness). Let $\eta : \mathcal{I} \rightarrow \mathbb{R}^n$ be a Lebesgue measurable function. η is said to be essentially bounded if there is a $M > 0$ such that for almost all $t \in \mathcal{I}$, $\|\eta(t)\| \leq M$.

Similar to integration and absolute continuity, Local Essential Boundedness can be defined as follows:

Definition 9 (Local Essential Boundedness). Let $\eta : \mathcal{I} \rightarrow \mathbb{R}^n$ be a Lebesgue measurable function. η is said to be locally essentially bounded if, for each $a, b \in \mathcal{I}$, there is a $M > 0$ such that for almost all $t \in]a, b[$, $\|\eta(t)\| \leq M$.

By observing the previous definitions, the next property can be stated.

Property 2 (Folk properties).

- Any absolutely continuous function is continuous
- Any continuous function is locally bounded.
- Any measurable, locally essentially bounded function is locally integrable.
- Any integrable function is locally integrable.

Let us recall the definition of Lipschitz continuity and its local variant.

Definition 10 (Lipschitz Continuity). A function $\omega : \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_2}$ is said to be Lipschitz continuous, if there exists $k \geq 0$, such that for all $\mathbf{x}_1, \mathbf{x}_2$ of $\mathbb{R}^{n_1}$, $\|\omega(\mathbf{x}_2) - \omega(\mathbf{x}_1)\| \leq k \|\mathbf{x}_2 - \mathbf{x}_1\|$.

Definition 11 (Local Lipschitz Continuity). A function $\omega : \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_2}$ is said to be locally Lipschitz continuous, if ω is Lipschitz continuous on every compact subset of $\mathbb{R}^{n_1}$.

Now, we can define an Initial Value Problem (IVP), which corresponds to calculating the solution to an ordinary differential equation:

Definition 12 (Initial Value Problem (IVP)). Let $\zeta : \begin{cases} \mathcal{I} \times \mathbb{R}^n & \rightarrow \mathbb{R}^n \\ \mathbf{x}, t & \rightarrow \zeta(\mathbf{x}, t) \end{cases}$ be a function such that the following two assumptions hold:

(H1) ζ is measurable on t : $\zeta(\mathbf{x}, \cdot)$ is measurable for each $\mathbf{x} \in \mathbb{R}^n$;

(H2) ζ is continuous on $\mathbf{x}$: $\zeta(\cdot, t)$ is continuous for each $t \in \mathcal{I}$.

Let $(t_0, \xi_0) \in \mathcal{I} \times \mathbb{R}^n$ be an initial condition.

A function ξ is said to be a solution of the Initial Value Problem (IVP (ζ, t_0, ξ_0)) on an interval $\mathcal{J} \subseteq \mathcal{I}$ with $t_0 \in \mathcal{J}$ if $\xi : \mathcal{J} \rightarrow \mathbb{R}^n$ is a locally absolutely continuous function such that

$$\text{for all } t \in \mathcal{J}, \xi(t) = \xi_0 + \int_{t_0}^t \zeta(\xi(\tau), \tau) d\tau,$$

in other words,

$$\text{for almost every } t \in \mathcal{J}, \frac{d\xi}{dt} = \zeta(\xi(t), t), \text{ and } \xi(t_0) = \xi_0 \text{ where } \xi(t) \in \mathbb{R}^n. \text{ (IVP)}$$

Recall that the notation $f|_A(x) = f(x)$ signifies that the restriction of f to the set A is essentially the same function as f , but it is defined exclusively on the subset A .

Definition 13 (Maximal Solution). *Let ζ, t_0, ξ_0 defining an Initial Value Problem. Let ξ be a solution of IVP (ζ, t_0, ξ_0) on $\mathcal{J}$. ξ is called a maximal solution of IVP (ζ, t_0, ξ_0) if for any other solution ξ' on an interval $\mathcal{J}'$, such that $\xi|_{\mathcal{J} \cap \mathcal{J}'} = \xi'|_{\mathcal{J} \cap \mathcal{J}'}$ we have $\mathcal{J}' \subseteq \mathcal{J}$:*

The following theorem in the Carathéodory Existence Theorem concerns the existence of a maximal solution on a neighborhood of the initial condition.

Theorem 5 (Carathéodory Existence Theorem). *Let $\zeta : \mathbb{R}^n \times \mathcal{I} \rightarrow \mathbb{R}^n$ satisfying the assumptions (H1) and (H2). Let's suppose that ζ satisfies two additional assumptions :*

(H3) *ζ is locally integrable on t . That is, for each $\mathbf{x} \in \mathbb{R}^n$, $t \rightarrow \zeta(\mathbf{x}, t)$ is locally integrable. Furthermore, for each $\mathbf{x}_0 \in \mathbb{R}^n$, there is a locally integrable function $\beta_{\mathbf{x}_0} : \mathcal{I} \rightarrow \mathbb{R}_+$ such that for almost all $t \in \mathcal{I}$, $\|\zeta(\mathbf{x}_0, t)\| \leq \beta_{\mathbf{x}_0}(t)$.*

(H4) *ζ is locally Lipschitz continuous on $\mathbf{x}$. That is, for each $t \in \mathcal{I}$, $\mathbf{x} \rightarrow \zeta(\mathbf{x}, t)$ is locally Lipschitz continuous. For each $\mathbf{x}_0 \in \mathbb{R}^n$, there is a locally integrable function $\alpha_{\mathbf{x}_0} : \mathcal{I} \rightarrow \mathbb{R}_+$ on a ball $B_\rho(\mathbf{x}_0)$ centered at $\mathbf{x}_0$ of radius ρ such that for all $t \in \mathcal{I}$, for all $\mathbf{x}_1, \mathbf{x}_2 \in B_\rho(\mathbf{x}_0)$, $\|\zeta(\mathbf{x}_2, t) - \zeta(\mathbf{x}_1, t)\| \leq \alpha_{\mathbf{x}_0}(t) \|\mathbf{x}_2 - \mathbf{x}_1\|$*

Then, for any initial condition $(t_0, \xi_0) \in \mathcal{I} \times \mathbb{R}^n$, there exists a unique maximal solution of IVP (ζ, t_0, ξ_0) .

Definition 14 (Global Solution). *ξ is said to be a global solution of IVP (ζ, t_0, ξ_0) if the validity interval $\mathcal{J}$ of ξ is the whole interval $\mathcal{I}$.*

This following theorem focuses on establishing the uniqueness of a solution's existence over the entire the domain of the function.

Theorem 6 (Global Carathéodory Existence Theorem). *Let $\zeta : \mathbb{R}^n \times \mathcal{I} \rightarrow \mathbb{R}^n$ satisfying the assumptions (H1) and (H2), (H3). Let's replace the assumption (H4) with a stronger global assumption:*

(H4*) *ζ is Lipschitz continuous on $\mathbf{x}$. That is, $\mathbf{x} \rightarrow \zeta(\mathbf{x}, t)$ is Lipschitz continuous for each $t \in \mathcal{I}$. Furthermore, there is a locally integrable function $\alpha : \mathcal{I} \rightarrow \mathbb{R}_+$ such that for all $t \in \mathcal{I}$, for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, $\|\zeta(\mathbf{x}_2, t) - \zeta(\mathbf{x}_1, t)\| \leq \alpha(t) \|\mathbf{x}_2 - \mathbf{x}_1\|$*

Then, for any initial condition $(t_0, \xi_0) \in \mathcal{I} \times \mathbb{R}^n$, there exists a unique global maximal solution of IVP (ζ, t_0, ξ_0) .

B.2 .Technical proofs

B.2.1 .Existence and Uniqueness Theorem

We consider a *observable trajectory* as a function $u : [t_0, T] \times \mathbb{R}^p \rightarrow \mathbb{R}^n$ where $\mathbb{R}^p$ denotes a space of *control parameter* $\mathbf{c} \in \mathbb{R}^p$ which dictate the system's dynamics. Under the assumption that u is L^∞ (that is integrable and essentially bounded), this section demonstrates how a simple *implicit function* $g : \mathbb{R}^n \times \mathbb{R}^m \times [t_0, T] \rightarrow \mathbb{R}^m$ can lead to additional *implicit trajectory* under the assumption that g is sufficiently smooth. Let us recall the two following definitions:

Definition 15 (Observable trajectory). Let $u : \begin{cases} [t_0, T] \times \mathbb{R}^p & \rightarrow \mathbb{R}^n \\ t, \mathbf{c} & \rightarrow u(t, \mathbf{c}) \end{cases}$. Function u is a observable trajectory if u is L^∞ , that is integrable and essentially bounded.

Definition 16 (Implicit Function). Let $g : \begin{cases} \mathbb{R}^n \times \mathbb{R}^m \times [t_0, T] & \rightarrow \mathbb{R}^m \\ \mathbf{x}_u, \mathbf{x}_v, t & \rightarrow g(\mathbf{x}_u, \mathbf{x}_v, t) \end{cases}$. Function g is a implicit function if it satisfied the three properties:

- **Lipschitz Continuity in State Space:** g is uniformly Lipschitz continuous on $\mathbf{x}_u$. That is, for each $\mathbf{x}_v, t \in \mathbb{R}^m \times [t_0, T]$, $\mathbf{x}_u \rightarrow g(\mathbf{x}_u, \mathbf{x}_v, t)$ is Lipschitz continuous and there exists $k_{\mathbf{x}_u}$ such that for all $\mathbf{x}_u \in \mathbb{R}^n$, for all $t \in [t_0, T]$, for all $\mathbf{x}_{u1}, \mathbf{x}_{u2} \in \mathbb{R}^n$, $\|g(\mathbf{x}_{u2}, \mathbf{x}_v, t) - g(\mathbf{x}_{u1}, \mathbf{x}_v, t)\| \leq k_{\mathbf{x}_u} \|\mathbf{x}_{u2} - \mathbf{x}_{u1}\|$.
- **Lipschitz Continuity in Integral State Space:** g is uniformly Lipschitz continuous on $\mathbf{x}_v$. That is, for each $\mathbf{x}_u, t \in \mathbb{R}^n \times [t_0, T]$, $\mathbf{x}_v \rightarrow g(\mathbf{x}_u, \mathbf{x}_v, t)$ is Lipschitz continuous and there exists $k_{\mathbf{x}_v}$ such that for all $\mathbf{x}_u \in \mathbb{R}^n$, for all $t \in [t_0, T]$, for all $\mathbf{x}_{v1}, \mathbf{x}_{v2} \in \mathbb{R}^m$, $\|g(\mathbf{x}_u, \mathbf{x}_{v2}, t) - g(\mathbf{x}_u, \mathbf{x}_{v1}, t)\| \leq k_{\mathbf{x}_v} \|\mathbf{x}_{v2} - \mathbf{x}_{v1}\|$.
- **Continuity over time:** g is continuous on t . That is, for each $\mathbf{x}_u, \mathbf{x}_v \in \mathbb{R}^n \times \mathbb{R}^m$, $t \rightarrow g(\mathbf{x}_u, \mathbf{x}_v, t)$ is continuous.

Theorem 7 (Implicit Trajectory Existence). Let g be an implicit function and f_θ be a multi layer perceptron. For any initial condition $v_0(\mathbf{c})$, there exists a unique function v such that for all $t \in [t_0, T]$, for all $\mathbf{c} \in \mathbb{R}^p$, $\frac{\partial v}{\partial t}(t, \mathbf{c}) = g(f_\theta(v(t, \mathbf{c}), t, \mathbf{c}), v(t, \mathbf{c}), t)$ and $v(t_0, \mathbf{c}) = v_0(\mathbf{c})$

Proof. To establish this theorem, it suffices to show that, for each $\mathbf{c} \in \mathbb{R}^p$, $t \rightarrow v(t, \mathbf{c})$ is a global solution of an IVP.

For any given $\mathbf{c} \in \mathbb{R}^p$, we must show that the function $g_{u,\mathbf{c}} : t, \mathbf{x}_v \rightarrow g(u(t, \mathbf{c}), \mathbf{x}_v, t)$ satisfies all the four assumptions of the Global Carathéodory Existence Theorem 6 (i.e. Assumptions (H1), (H2), (H3) and (H4*)).

First, we focus on Assumption (H1). We aim to prove that function $g_{u,\mathbf{c}}$ is measurable with respect to t ; in other words, function $\psi_{\mathbf{x}_v} : t \rightarrow g_{u,\mathbf{c}}(t, \mathbf{x}_v) = g(u(t, \mathbf{c}), \mathbf{x}_v, t)$ is measurable for any $\mathbf{x}_v \in \mathbb{R}^m$. To establish this, $\psi_{\mathbf{x}_v}$ results from the composition of functions u and g . By definition of observable trajectory, u is integrable, and therefore measurable. Since g is continuous, it is also measurable. Consequently, $\psi_{\mathbf{x}_v}$ is a composition of measurable function. We conclude that $\psi_{\mathbf{x}_v}$ is measurable for any $\mathbf{x}_v \in \mathbb{R}^m$. So, Assumption (H1) holds.

Second, we consider on Assumption (H2): We want to prove that function $g_{u,\mathbf{c}}$ is continuous with respect to $\mathbf{x}_v$. To prove this, we show that $\varphi_t : \mathbf{x}_v \rightarrow g_{u,\mathbf{c}}(t, \mathbf{x}_v) = g(u(t, \mathbf{c}), \mathbf{x}_v, t)$ is continuous for each $t \in [t_0, T]$. Let us fix an arbitrary t in $[t_0, T]$. Consider any two points $\mathbf{x}_{v,1}, \mathbf{x}_{v,2} \in \mathbb{R}^m$. Then, we have:

$$\begin{aligned} \|\varphi_t(\mathbf{x}_{v,2}) - \varphi_t(\mathbf{x}_{v,1})\| &= \|g(u(t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(u(t, \mathbf{c}), \mathbf{x}_{v,1}, t)\| \\ &\leq k_v \|\mathbf{x}_{v,2} - \mathbf{x}_{v,1}\| \end{aligned}$$

This inequality verifies that φ_t is a Lipschitz continuous function with respect to $\mathbf{x}_v$ (and also continuous), allowing us to conclude that $g_{u,\mathbf{c}}$ is continuous on $\mathbf{x}_v$. Thus, Assumption (H2) is satisfied.

Third, we turn our attention to Assumption (H3) corresponding to prove that $g_{u,\mathbf{c}}$ is locally integrable on t . We aim to demonstrate that $\psi_{\mathbf{x}_v}$ is essentially bounded for any $\mathbf{x}_v \in \mathbb{R}^m$.

Let $\mathbf{x}_v \in \mathbb{R}^m$. We show that there is a $M > 0$ such that for almost all $t \in [t_0, T]$, $\psi_{\mathbf{x}_v}(t) \leq M$.

Note that observable trajectory u is essentially bounded. Therefore, there exists a constant $M_u > 0$ such that for almost all $t \in [t_0, T]$, $\|u(t, \mathbf{c})\| \leq M_u$. Furthermore, as g is a continuous function, g is bounded on the compact subset $\mathcal{B}_{M_u}[0] \times \{\mathbf{x}_v\} \times [t_0, T]$, where $\mathcal{B}_{M_u}[0]$ is the closed ball of $\mathbb{R}^n$ centered on 0 and of radius M_u . Therefore, there exists a value M such that for all $\mathbf{x}_u, t \in \mathcal{B}_{M_u}[0] \times [t_0, T]$, $g(\mathbf{x}_u, \mathbf{x}_v, t) \leq M$. Finally, as for almost all $t \in [t_0, T]$, $u(t, \mathbf{c}) \in \mathcal{B}_{M_u}[0]$, we have for almost all $t \in [t_0, T]$, $g(u(t, \mathbf{c}), \mathbf{x}_v, t) \leq M$, that is $\psi_{\mathbf{x}_v}(t) \leq M$.

Thus, $\psi_{\mathbf{x}_v}$ is essentially bounded for any $\mathbf{x}_v \in \mathbb{R}^m$. As previously demonstrated, function $\psi_{\mathbf{x}_v} : t \rightarrow g_{u,\mathbf{c}}(t, \mathbf{x}_v) = g(u(t, \mathbf{c}), \mathbf{x}_v, t)$ is measurable for any $\mathbf{x}_v \in \mathbb{R}^m$ (refer to the paragraph concerning Assumption (H1)). Let fix an arbitrary $\mathbf{x}_v \in \mathbb{R}^m$ Proposition 2) states that $\psi_{\mathbf{x}_v}$ being measurable and essentially bounded (and so, locally essentially bounded), $\psi_{\mathbf{x}_v}$ is locally integrable. Thus, Assumption (H3) is satisfied.

Fourth, we focus on Assumption (H4*). Let us denote by $\forall t \in [t_0, T], \varphi_t : \mathbf{x}_v \rightarrow g_{u, \mathbf{c}}(t, \mathbf{x}_v) = g(u(t, \mathbf{c}), \mathbf{x}_v, t)$ the partial application of $g_{u, \mathbf{c}}$ with respect to $\mathbf{x}_v$. The proof of this assumption should be split into two parts:

1. We will prove that function $\varphi_t : \mathbf{x}_v \rightarrow g_{u, \mathbf{c}}(t, \mathbf{x}_v) = g(u(t, \mathbf{c}), \mathbf{x}_v, t)$ is Lipschitz continuous for each $t \in [t_0, T]$. For any given $t \in [t_0, T]$, for all $\mathbf{x}_{v1}, \mathbf{x}_{v2} \in \mathbb{R}^m$, we have

$$\begin{aligned} \|\varphi_t(\mathbf{x}_{v2}) - \varphi_t(\mathbf{x}_{v1})\| &= \|g(u(t, \mathbf{c}), \mathbf{x}_{v2}, t) - g(u(t, \mathbf{c}), \mathbf{x}_{v1}, t)\| \\ &\leq k_{\mathbf{x}_v} \|\mathbf{x}_{v2} - \mathbf{x}_{v1}\| \end{aligned}$$

The latter inequation is obtained due to the fact g is uniformly Lipschitz continuous on $\mathbf{x}_v$. Hence for all $t \in [t_0, T]$, φ_t is Lipschitz continuous.

2. Furthermore, it remains to be demonstrated that there is a locally integrable function $\alpha : [t_0, T] \rightarrow \mathbb{R}_+$ satisfying the condition that for all $t \in [t_0, T]$, for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, the inequality $\|g_{u, \mathbf{c}}(t, \mathbf{x}_1) - g_{u, \mathbf{c}}(t, \mathbf{x}_2)\| \leq \alpha(t) \|\mathbf{x}_2 - \mathbf{x}_1\|$ holds. So, using the first point, the function α is defined as: for any t , $\alpha(t) = k_{\mathbf{x}_v}$. Consider $(a, b) \in [t_0, T]$. We have $\int_a^b \|\alpha(t)\| dt = (b - a)k_{\mathbf{x}_v} < +\infty$. So, the function α is locally integrable.

We have proved that the function $g_{u, \mathbf{c}} : t, \mathbf{x}_v \rightarrow g(u(t, \mathbf{c}), \mathbf{x}_v, t)$ satisfies all the four assumptions of the Global Carathéodory Existence Theorem 6 for any given $\mathbf{c} \in \mathbb{R}^p$. Using the Global Carathéodory Existence Theorem 6, we conclude the existence and unicity of a global solution v on the whole interval $[t_0, T]$, and the theorem holds. $\square$

Definition 17. *The unique solution guaranteed by Theorem 7 is denoted by $\mathbf{G}_{g, v_0}(f_\theta)$.*

B.2.2 . Universal Approximation Theorem

In this section, we show that, knowing the value of the implicit trajectory, there exists a neural network approximating as closely as desired the observable trajectory.

In a second time, we show that knowing this neural network and the implicit function, we can reconstruct the implicit trajectory and the observable trajectory as closely as desired.

Theorem 8 (INode solution existence). *Let g be an implicit function and f_θ be a multi layer perceptron. For any initial condition $v_0(\mathbf{c})$, there exists a unique function v such that for all $t \in [t_0, T]$, for all $\mathbf{c} \in \mathbb{R}^p$, $\frac{\partial v}{\partial t}(t, \mathbf{c}) = g(f_\theta(v(t, \mathbf{c}), t, \mathbf{c}), v(t, \mathbf{c}), t)$ and $v(t_0, \mathbf{c}) = v_0(\mathbf{c})$*

Proof. Let $\mathbf{c} \in \mathbb{R}^p$

Here we can apply the Picard-Lindelöf Theorem (see Appendix B) by verifying the continuity of the right-term of the ODE.

1. *Continuity.*

$\forall \mathbf{x}_v \in \mathbb{R}^m, t \rightarrow g(f_\theta(\mathbf{x}_v, t, \mathbf{c}), \mathbf{x}_v, t)$ is continuous as a composition of continuous functions.

2. *Lipschitz Continuity.*

Let $(t, \mathbf{x}_{v,1}, \mathbf{x}_{v,2}) \in [t_0, T] \times \mathbb{R}^m \times \mathbb{R}^m$.

$$\begin{aligned} & \|g(f_\theta(\mathbf{x}_{v,2}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,1}, t)\| \\ &= \|g(f_\theta(\mathbf{x}_{v,2}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) + g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,1}, t)\| \\ &\leq \|g(f_\theta(\mathbf{x}_{v,2}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,2}, t)\| + \|g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,2}, t) - g(f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c}), \mathbf{x}_{v,1}, t)\| \\ &\leq k_{\mathbf{x}_u} \|f_\theta(\mathbf{x}_{v,2}, t, \mathbf{c}) - f_\theta(\mathbf{x}_{v,1}, t, \mathbf{c})\| + k_{\mathbf{x}_v} \|\mathbf{x}_{v,2} - \mathbf{x}_{v,1}\| \\ &\leq k_{\mathbf{x}_u} k_{f_\theta} \|\mathbf{x}_{v,2} - \mathbf{x}_{v,1}\| + k_{\mathbf{x}_v} \|\mathbf{x}_{v,2} - \mathbf{x}_{v,1}\| \\ &= (k_{\mathbf{x}_u} k_{f_\theta} + k_{\mathbf{x}_v}) \|\mathbf{x}_{v,2} - \mathbf{x}_{v,1}\| \end{aligned}$$

We can apply the Picard-Lindelöf Theorem to conclude. $\square$

Definition 18. The unique solution guaranteed by Theorem 8 is denoted by $\mathbf{G}_{g, \mathbf{v}_0}(f_\theta)$.

Following the notation from [37], we denote by $\mathcal{NN}_{m,n,k}^\rho$ the set of multilayer perceptron that have m neurons in the input layer, n neurons in the output layer and an arbitrary number of hidden layer each containing at most k neurons each, using the activation function ρ .

Theorem 9 (Universal Approximation of the Pretext Task). *Let $\mathbf{u}$ be an observable trajectory and g be a implicit function. Let $\mathbf{v}$ be the implicit trajectory of $\mathbf{u}$ guaranteed by Theorem 1: $\mathbf{v} = \mathcal{F}_{g, \mathbf{v}_0}(\mathbf{u})$.*

For all $\varepsilon > 0$, for all $k > 0$, there exists $f_\theta \in \mathcal{NN}_{m+p+1, n, \max(m+n+2, n)}^{\text{ReLU}}$ such for all $t, \mathbf{c} \in [t_0, T] \times \mathbb{R}^p$, $\|f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| < \varepsilon$, and f_θ is a k -uniformly Lipschitz continuous over the implicit space.

Proof. Multi-layer perceptrons are Lipschitz continuous with a Lipschitz constant equal to the product of the spectral norm of the weight matrices. In other words, there exists a constant $k_{f_\theta} > 0$ such that for all $\mathbf{x}_{v,1}, \mathbf{x}_{v,2} \in \mathbb{R}^m$, for all $t_1, t_2 \in [t_0, T]$, for all $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^p$,

$$\|f_\theta(\mathbf{x}_{v,2}, t_2, \mathbf{c}_2) - f_\theta(\mathbf{x}_{v,1}, t_1, \mathbf{c}_1)\| \leq k_{f_\theta} \left\| \begin{pmatrix} \mathbf{x}_{v,2} \\ t_2 \\ \mathbf{c}_2 \end{pmatrix} - \begin{pmatrix} \mathbf{x}_{v,1} \\ t_1 \\ \mathbf{c}_1 \end{pmatrix} \right\|$$

By setting $t = t_1 = t_2$ and $\mathbf{c} = \mathbf{c}_1 = \mathbf{c}_2$, in the previous inequality, we obtain:

$$\|f_\theta(\mathbf{x}_{\mathbf{v}_2}, t, \mathbf{c}) - f_\theta(\mathbf{x}_{\mathbf{v}_1}, t, \mathbf{c})\| \leq k_{f_\theta} \left\| \begin{pmatrix} \mathbf{x}_{\mathbf{v}_2} - \mathbf{x}_{\mathbf{v}_1} \\ 0 \\ 0 \end{pmatrix} \right\| = k_{f_\theta} \|\mathbf{x}_{\mathbf{v}_2} - \mathbf{x}_{\mathbf{v}_1}\|$$

We conclude that multi-layer perceptrons are smooth estimators.

Moreover, since observable trajectory $\mathbf{u}$ is L^∞ (integrable and essentially bounded), the approximation result is derived from the universal approximation theorem of [48]. This means that, for all $\varepsilon > 0$, there exists a neural network $g_\theta \in \mathcal{NN}_{p+1, n, \max(p+2, n)}^{\text{ReLU}}$ such that for all $t, \mathbf{c} \in [t_0, T] \times \mathbb{R}^p$, $\|g_\theta(t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| < \varepsilon$.

Note that g_θ takes $t, \mathbf{c}$ as input. We aim to build a neural network that takes $\mathbf{v}, t, \mathbf{c}$ as input. To achieve this, we build a neural network by expanding the layers and incorporating additional neurons with zero weights. Extending g_θ by adding 0 weights on the first layer, we can construct f_θ such that for all $t \in [t_0, T]$, for all $\mathbf{c} \in \mathbb{R}^p$, we have $f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}) = g_\theta(t, \mathbf{c})$ which concludes the proof of the theorem. $\square$

Theorem 10 (INODE Universal Approximation). *Let $\mathbf{u}$ be an observable trajectory and g be an implicit function. Let $\mathbf{v}$ be the implicit trajectory of $\mathbf{u}$ corresponding to the solution guaranteed by Theorem 1: $\mathbf{v} = \mathcal{F}_{g, \mathbf{v}_0}(\mathbf{u})$.*

For all $\varepsilon > 0$, there exists $\varepsilon_1 > 0$ such that all $f_\theta \in \mathcal{NN}_{m+p+1, n, \max(m+p+2, n)}^{\text{ReLU}}$ verifying: for all $t, \mathbf{c} \in [t_0, T] \times \mathbb{R}^p$, $\|f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| < \varepsilon_1$, also verifies: $\|f_\theta(\mathbf{G}_{g, \mathbf{v}_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| < \varepsilon$.

Furthermore, if we denote by $k_{\mathbf{x}_\mathbf{u}}$ the Lipschitz constant of the implicit function g over the Observable Space, and by k the Lipschitz constant of the neural network over the implicit trajectory variable, we also have:

$$\|f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| < \varepsilon_1 \Rightarrow \|f_\theta(\mathbf{G}_{g, \mathbf{v}_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| < (1 + k k_{\mathbf{x}_\mathbf{u}}(T - t_0)) \varepsilon_1$$

Proof. Let $\varepsilon_1 > 0$, $k > 0$. Theorem 9 states that there exists at least a neural network f_θ satisfying the above properties. Let f_θ be a neural network satisfying the above properties.

$$\begin{aligned} \|g(f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}), \mathbf{v}(t, \mathbf{c}), t) - g(\mathbf{u}(t, \mathbf{c}), \mathbf{v}(t, \mathbf{c}), t)\| &\leq k_{\mathbf{x}_\mathbf{u}} \|f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| \\ &\leq k_{\mathbf{x}_\mathbf{u}} \varepsilon_1 \end{aligned}$$

Integrating the term $g(f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}), \mathbf{v}(t, \mathbf{c}), t) - g(\mathbf{u}(t, \mathbf{c}), \mathbf{v}(t, \mathbf{c}), t)$, the previous inequality can be rewritten as follows:

$$\begin{aligned}
\|\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}) - \mathbf{v}(t, \mathbf{c})\| &= \left\| \int_{t_0}^t g(f_\theta(\mathbf{v}(\tau, \mathbf{c}), \tau, \mathbf{c}), \mathbf{v}(\tau, \mathbf{c}), \tau) - g(\mathbf{u}(\tau, \mathbf{c}), \mathbf{v}(\tau, \mathbf{c}), \tau) d\tau \right\| \\
&\leq \int_{t_0}^t \|g(f_\theta(\mathbf{v}(\tau, \mathbf{c}), \tau, \mathbf{c}), \mathbf{v}(\tau, \mathbf{c}), \tau) - g(\mathbf{u}(\tau, \mathbf{c}), \mathbf{v}(\tau, \mathbf{c}), \tau)\| d\tau \\
&\leq (t - t_0) k_{\mathbf{x}_u} \varepsilon_1 \\
&\leq (T - t_0) k_{\mathbf{x}_u} \varepsilon_1
\end{aligned}$$

Finally, using f_θ for the final estimation, we have

$$\begin{aligned}
&\|f_\theta(\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| \\
&= \|f_\theta(\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}) + f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| \\
&\leq \|f_\theta(\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}), t, \mathbf{c}) - f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c})\| + \|f_\theta(\mathbf{v}(t, \mathbf{c}), t, \mathbf{c}) - \mathbf{u}(t, \mathbf{c})\| \\
&\leq k \|\mathbf{G}_{g,v_0}(f_\theta)(t, \mathbf{c}) - \mathbf{v}(t, \mathbf{c})\| + \varepsilon_1 \\
&\leq (k(T - t_0) k_{\mathbf{x}_u} + 1) \varepsilon_1
\end{aligned}$$

Choosing $\varepsilon_1 = \frac{\varepsilon}{1 + k(T - t_0) k_{\mathbf{x}_u}}$, we conclude the proof. $\square$

B.2.3 .Folk Knowledge

To show that the Lipschitz constant of a linear layer is equal to the largest singular value of its weight matrix, consider the weight matrix W .

For a linear layer, consider the function $f(x) = Wx + b$. We want to find the smallest k such that

$$\forall x_1, x_2, \quad \|Wx_1 - Wx_2\| \leq k \|x_1 - x_2\| \quad (\text{B.2})$$

By setting $y = x_1 - x_2$, we can reformulate the problem by finding k such that

$$\forall y, \quad \|Wy\| \leq k \|y\| \quad (\text{B.3})$$

So the Lipschitz constant k is given by the maximum value of the ratio $\frac{\|Wy\|}{\|y\|}$ for any $y \neq 0$. Formally, we have

$$k = \sup_{y \neq 0} \frac{\|Wy\|}{\|y\|} = \|W\|_{\text{op}}, \quad (\text{B.4})$$

where $\|W\|_{\text{op}}$ denotes the operator norm (or spectral norm) of W , which corresponds to the largest singular value of W .

To show that the operative norm is indeed the largest singular value, we use the singular value decomposition (SVD) of W :

$$W = U\Sigma V^\top \quad (\text{B.5})$$

where U and V are orthogonal matrices, and Σ is a diagonal matrix containing the singular values of W , sorted in descending order. The largest singular value, $\sigma_{\max}$, is the first diagonal element of Σ . For a vector y , we have

$$\|Wy\| = \|U\Sigma V^\top y\| = \|\Sigma V^\top y\| \quad (\text{B.6})$$

because U is orthogonal and preserves the norm. So we want to maximise the ratio $\frac{\|\Sigma z\|}{\|z\|}$, where $z = V^\top y$. Since Σ is diagonal with non-negative elements, the ratio is maximised when z is a standard basis vector corresponding to the largest singular value, $\sigma_{\max}$. Thus, we have

$$\|W\|_{\text{op}} = \sigma_{\max}$$

In short, the Lipschitz constant of the linear transformation W is equal to the largest singular value of the weight matrix, which corresponds to the operator norm (or spectral norm) of W .

Bibliography

- [1] Andreas Adelman. "On Nonintrusive Uncertainty Quantification and Surrogate Model Construction in Particle Accelerator Modeling". In: *SIAM/ASA Journal on Uncertainty Quantification* 7.2 (2019), pp. 383–416.
- [2] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.
- [3] Reza Alizadeh, Janet K. Allen, and Farrokh Mistree. "Managing computational complexity using surrogate models: a critical review". In: *Research in Engineering Design* 31.3 (July 2020), pp. 275–298.
- [4] Kartik B Ariyur and Miroslav Krstic. *Real-Time Optimization by Extremum-Seeking Control*. John Wiley & Sons, Ltd, 2003.
- [5] A. Bacci, V. Petrillo, and M. Rossetti Conti. "GIOTTO: A Genetic Code for Demanding Beam-dynamics Optimizations". In: *Proc. of International Particle Accelerator Conference (IPAC'16), Busan, Korea, May 8-13, 2016*. International Particle Accelerator Conference 7. Geneva, Switzerland: JACoW, June 2016, pp. 3073–3076.
- [6] A. Bacci et al. "Maximizing the brightness of an electron beam by means of a genetic algorithm". In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 263.2 (2007), pp. 488–496.
- [7] Ivan V. Bazarov and Charles K. Sinclair. "Multivariate optimization of a high brightness dc gun photoinjector". In: *Phys. Rev. ST Accel. Beams* 8 (3 Mar. 2005), p. 034202.
- [8] S. Biedron et al. "Snowmass21 Accelerator Modeling Community White Paper". In: *US Community Study on the Future of Particle Physics (Snowmass 2021)*. 2022.
- [9] Y-Lan Boureau, Jean Ponce, and Yann LeCun. "A theoretical analysis of feature pooling in visual recognition". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Omnipress, 2010, pp. 111–118.
- [10] Christelle Bruni et al. "ThomX : un démonstrateur de source compacte de rayons X". In: *Reflète de la physique* 50 (Sept. 1, 2016), pp. 14–19.

- [11] R. Qi Charles et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *CVPR*. 2017.
- [12] Ricky T. Q. Chen et al. "Neural Ordinary Differential Equations". In: *NeurIPS*. 2018.
- [13] Zhao Chen et al. "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 794–803.
- [14] Wikipedia contributors. *Carathéodory's existence theorem — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Carath%C3%A9odory%27s_existence_theorem&oldid=1180000109.
- [15] Owen Convery et al. "Uncertainty quantification for virtual diagnostic of particle accelerators". In: *Phys. Rev. Accel. Beams* 24 (7 July 2021), p. 074602.
- [16] Sanjoy Dasgupta. "Two faces of active learning". In: *Theoretical Computer Science* 412.19 (2011), pp. 1767–1781.
- [17] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. "Unsupervised Visual Representation Learning by Context Prediction". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.
- [18] J. Duris et al. "Bayesian Optimization of a Free-Electron Laser". In: *PRL* (Mar. 2020).
- [19] A. L. Edelen et al. "Neural Networks for Modeling and Control of Particle Accelerators". In: *IEEE Transactions on Nuclear Science* 63.2 (2016), pp. 878–897.
- [20] Auralee Edelen et al. "Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems". In: *PRAB* (2020).
- [21] C. Emma et al. "Machine learning-based longitudinal phase space prediction of particle accelerators". In: *Phys. Rev. Accel. Beams* 21 (11 Nov. 2018), p. 112802.
- [22] Klaus Floettmann. *ASTRA: A space charge tracking algorithm, manual*, 2017. Version 3.2. 2017.
- [23] Alexander Forrester, Andras Sobester, and Andy Keane. "Sampling Plans". In: *Engineering Design via Surrogate Modelling*. John Wiley & Sons, Ltd, 2008. Chap. 1, pp. 1–31. ISBN: 9780470770801.

- [24] Alexis Gamelin. "Collective effects in a transient microbunching regime and ion cloud mitigation in ThomX". PhD thesis. Université Paris Saclay (COMUE), Sept. 2018.
- [25] H. Grote and F. Schmidt. *MAD-X: An Upgrade from MAD8*. CERN, 2003. URL: <https://cern.ch/mad/>.
- [26] Ryoichi Hajima et al. "Optimization of wiggler magnets ordering using a genetic algorithm". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 318.1 (1992), pp. 822–824.
- [27] J. M Hammersley and D. C. Handscomb. *Monte Carlo methods*. eng. Monographs on Statistics and Applied Probability. Springer Dordrecht, 1964.
- [28] Adi Hanuka et al. "Physics model-informed Gaussian process for online optimization of particle accelerators". In: *PRAB* (July 2021).
- [29] Frank E. Harrell and C. E. Davis. "A New Distribution-Free Quantile Estimator". In: *Biometrika* 69.3 (1982), pp. 635–640.
- [30] Roger W Hockney and James W Eastwood. *Computer simulation using particles*. crc Press, 1988.
- [31] Alicia Hofler et al. "Innovative applications of genetic algorithms to problems in accelerator physics". In: *Phys. Rev. ST Accel. Beams* 16 (1 Jan. 2013), p. 010101.
- [32] Xiaobiao Huang. "Robust simplex algorithm for online optimization". In: *Phys. Rev. Accel. Beams* 21 (10 Oct. 2018), p. 104601.
- [33] Martin Jaggi. "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 1. Atlanta, Georgia, USA: PMLR, June 2013, pp. 427–435.
- [34] W. Kaabi et al. "PERLE: A High Power Energy Recovery Facility". In: (June 2019).
- [35] Verena Kain et al. "Sample-efficient reinforcement learning for CERN accelerator control". In: *Phys. Rev. Accel. Beams* 23 (12 Dec. 2020), p. 124801.
- [36] Jan Kaiser, Oliver Stein, and Annika Eichler. "Learning-based Optimisation of Particle Accelerators Under Partial Observability Without Real-World Training". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 10575–10585.

- [37] Patrick Kidger and Terry Lyons. "Universal Approximation with Deep Narrow Networks". In: *Proceedings of Thirty Third Conference on Learning Theory*. Ed. by Jacob Abernethy and Shivani Agarwal. Vol. 125. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 2306–2327.
- [38] M. Kranj čevi ć et al. "Multiobjective optimization of the dynamic aperture using surrogate models based on artificial neural networks". In: *Phys. Rev. Accel. Beams* 24 (1 Jan. 2021), p. 014601.
- [39] Lauwerens Kuipers and Harald Niederreiter. *Uniform distribution of sequences*. eng. Pure and applied mathematics a Wiley-Interscience series of texts, monographs, and tracts. J. Wiley & Sons, 1974.
- [40] Vitaly Kurin et al. "In Defense of the Unitary Scalarization for Deep Multi-Task Learning". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 12169–12183.
- [41] Remi Lam et al. "Learning skillful medium-range global weather forecasting". In: *Science* 382.6677 (2023), pp. 1416–1421.
- [42] Yongjun Li et al. "Genetic algorithm enhanced by machine learning in dynamic aperture optimization". In: *Phys. Rev. Accel. Beams* 21 (5 May 2018), p. 054601.
- [43] Bo Liu et al. "Conflict-Averse Gradient Descent for Multi-task learning". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 18878–18890.
- [44] R. J. Beckman M. D. McKay and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code". In: *Technometrics* 42.1 (2000), pp. 55–61.
- [45] L.J. Nevay et al. "BDSIM: An accelerator tracking code with particle-matter interactions". In: *Computer Physics Communications* 252 (2020), p. 107200.
- [46] Yoshihiko Ozaki et al. "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems". In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. GECCO '20. Association for Computing Machinery, 2020, pp. 533–541.

- [47] Chunghyun Park et al. "Fast Point Transformer". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 16949–16958.
- [48] Sejun Park et al. "Minimum Width for Universal Approximation". In: *International Conference on Learning Representations*. 2021.
- [49] E. Piselli et al. "CERN Accelerators Beam Optimization Algorithm". In: *Proc. ICALEPCS'19. International Conference on Accelerator and Large Experimental Physics Control Systems 17*. JACoW Publishing, Geneva, Switzerland, Aug. 2020, pp. 1379–1384.
- [50] Pulsar Physics. *General Particle Tracer (GPT)*. URL: <http://www.pulsar.nl/gpt>.
- [51] Charles R. Qi et al. "PointNet++: deep hierarchical feature learning on point sets in a metric space". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17*. Curran Associates Inc., 2017, pp. 5105–5114.
- [52] R. Ramjiawan et al. "Design and operation of transfer lines for plasma wakefield accelerators using numerical optimizers". In: *Phys. Rev. Accel. Beams* 25 (10 Oct. 2022), p. 101602.
- [53] Martin Reiser. *Theory and Design of Charged Particle Beams*. John Wiley & Sons, Ltd, 2008.
- [54] Miha Rescic, Rebecca Seviour, and Willem Blokland. "Predicting particle accelerator failures using binary classifiers". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 955 (2020), p. 163240.
- [55] Ryan Roussel et al. *Bayesian Optimization Algorithms for Accelerator Physics*. 2024. arXiv: [2312.05667](https://arxiv.org/abs/2312.05667).
- [56] Nicholas Roy and Andrew McCallum. "Toward Optimal Active Learning through Sampling Estimation of Error Reduction". In: *Proceedings of the 18th International Conference on Machine Learning*. Morgan Kaufmann, 2001, pp. 441–448.
- [57] Waldemar H. Scharf and Waldemar Scharf. *Particle Accelerators and Their Uses*. CRC Press, 1991. 700 pp.
- [58] A. Scheinker. "Adaptive machine learning for time-varying systems: low dimensional latent space tuning". In: *Journal of Instrumentation* 16.10 (Oct. 2021), P10008.
- [59] Alexander Scheinker and Spencer Gessner. "Adaptive method for electron bunch profile prediction". In: *Phys. Rev. ST Accel. Beams* 18 (10 Oct. 2015), p. 102801.

- [60] 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 (2022), pp. 2261–2268.
- [61] Alexander Scheinker, Xiaoying Pang, and Larry Rybarcyk. "Model-independent particle accelerator tuning". In: *Phys. Rev. ST Accel. Beams* 16 (10 Oct. 2013), p. 102803.
- [62] Alexander Scheinker et al. "An adaptive approach to machine learning for compact particle accelerators". In: *Scientific Reports* (Sept. 2021).
- [63] Ozan Sener and Vladlen Koltun. "Multi-Task Learning as Multi-Objective Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.
- [64] Burr Settles. "Active learning literature survey". In: (2009).
- [65] H. S. Seung, M. Oppen, and H. Sompolinsky. "Query by committee". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1992, pp. 287–294.
- [66] Shaoshuai Shi et al. "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [67] I.M Sobol'. "On the distribution of points in a cube and the approximate evaluation of integrals". In: *USSR Computational Mathematics and Mathematical Physics* 7.4 (1967), pp. 86–112.
- [68] Eduardo D Sontag. *Mathematical control theory: deterministic finite dimensional systems*. Vol. 6. Springer Science & Business Media, 2013.
- [69] Hugues Thomas et al. "KPConv: Flexible and Deformable Convolution for Point Clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.
- [70] K. Tian, J. Safranek, and Y. Yan. "Machine based optimization using genetic algorithms in a storage ring". In: *Phys. Rev. ST Accel. Beams* 17 (2 Feb. 2014), p. 020703.
- [71] Alessandro Variola et al. *ThomX Technical Design Report*. Tech. rep. Orsay, France: Laboratoire de l'Accélérateur Linéaire - UMR 8607, 2014, p. 164.

- [72] Yue Wang et al. "Dynamic Graph CNN for Learning on Point Clouds". In: *ACM Trans. Graph.* 38.5 (Oct. 2019).
- [73] Helmut Wiedemann. *Particle Accelerator Physics*. Graduate Texts in Physics. Springer International Publishing, 2015.
- [74] Alan Williams et al. "Experimental Safe Extremum Seeking for Accelerators". In: *IEEE Transactions on Control Systems Technology* (2024), pp. 1–10.
- [75] Derrick Xin et al. "Do Current Multi-Task Optimization Methods in Deep Learning Even Help?" In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 13597–13609.
- [76] Tianhe Yu et al. "Gradient Surgery for Multi-Task Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 5824–5836.
- [77] Imene Zemzemi. "High-performance computing and numerical simulation for laser wakefield acceleration with realistic laser profiles". PhD thesis. Institut Polytechnique de Paris, Dec. 2020.
- [78] Hengshuang Zhao et al. "Point Transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 16259–16268.