VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
INTERNATIONAL UNIVERSITY
DEPARTMENT OF PHYSICS
SPACE ENGINEERING

PHYSICS-INFORMED NEURAL NETWORK SURROGATE FOR ROCKET ASCENT TRAJECTORY OPTIMIZATION

HA QUANG HUY

A THESIS SUBMITTED TO
DEPARTMENT OF PHYSICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF ENGINEER IN SPACE ENGINEERING

HO CHI MINH CITY, VIET NAM
2025

Physics-Informed Neural Network Surrogate for Rocket Ascent Trajectory Optimization

HÀ QUANG HUY

Under the guidance and approval of the committee, and approved by its members, this thesis has been accepted in partial fulfillment of the requirements for the degree.

Approved:

_____
Chairperson

_____
Committee Member

_____
Committee Member

_____
Committee Member

_____
Committee Member

HONESTY DECLARATION

My name is Hà Quang Huy. I would like to declare that, apart from the acknowledged references, this thesis either does not use language, ideas, or other original material from anyone, or has not been previously submitted to any other educational and research programs or institutions. I fully understand that any writings in this thesis that contradict the above statement will automatically lead to rejection from the SE program at the International University – Vietnam National University Ho Chi Minh City.

Date: December 24, 2025

Student Signature:

Hà Quang Huy

TURNITIN DECLARATION

Name of Student: Hà Quang Huy

Date: December 24, 2025

Suppervisor Signature:

Student Signature:

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Nguyen Quang, for his invaluable guidance, support, and encouragement throughout this research. His expertise and insights have been instrumental in shaping the direction of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS AND NOTATIONS

6-DOF : six-degree-of-freedom

Adam : Adaptive Moment Estimation

AN : Direction AN (neural network architecture)

CasADi : Computer Algebra System for Automatic Differentiation and Integration

CPU : Central Processing Unit

GPU : Graphics Processing Unit

HDF5 : Hierarchical Data Format version 5

HSL : Harwell Subroutine Library

Hz : Hertz

IPOPT : Interior Point OPTimizer

LibTorch : C++ frontend for PyTorch

MSE : mean-squared error

MLP : Multilayer Perceptron

MUMPS : MUltifrontal Massively Parallel sparse direct Solver

NLP : nonlinear programme

NumPy : Numerical Python

OCP : Optimal Control Problem

ODE : Ordinary Differential Equation

PINN : Physics-Informed Neural Network

PyTorch : Python-based deep learning framework

RMSE : root mean square error

ABSTRACT

This thesis investigates physics-informed neural network (PINN) surrogates for six-degree-of-freedom (6-DOF) launch vehicle ascent dynamics, with the goal of approximating high-fidelity trajectory solutions at substantially reduced computational cost. Starting from a rigid-body 6-DOF dynamics model and a direct optimal control formulation, a dataset of numerically optimal ascent trajectories is generated using collocation-based methods. These trajectories serve as supervision for training structured neural surrogates that predict full state evolution from time and mission context parameters.

The primary contribution of this work is the design and evaluation of a structured PINN architecture, referred to as Direction AN, which combines Fourier time embeddings, a context encoder for physical parameters, a shared residual backbone, and specialized output branches for translational motion, rotational dynamics, and mass depletion. Physical consistency is enforced through a physics residual loss constructed from finite-difference approximations of the continuous-time equations of motion, together with additional structural constraints such as quaternion normalization, mass monotonicity, and boundary consistency. A phased training strategy is employed to balance data fidelity, physics regularization, and smoothing terms, improving stability during learning.

The resulting surrogate accurately reconstructs position, velocity, attitude, and mass trajectories over the launch phase (0–30 s) across a range of sounding-rocket configurations, producing smooth and physically plausible ascent profiles consistent with a high-fidelity truth integrator. While the surrogate is not used directly to optimize control inputs in this study, its differentiable structure and physics-aware design make it suitable for future integration into gradient-based trajectory optimization and guidance frameworks. Overall, the results demonstrate that carefully regularized PINN surrogates can serve as effective and scalable approximations of

complex rocket dynamics, providing a foundation for fast analysis and optimization in ascent trajectory design.

CHAPTER I

INTRODUCTION

1.1 Background and Motivation

Trajectory optimization for rocket ascent is a classical and central problem in aerospace engineering. The objective is to determine thrust and steering profiles that guide a launch vehicle from liftoff to a desired terminal state while satisfying nonlinear dynamics, path constraints, and structural or operational limits. Traditionally, this problem has been addressed using indirect methods derived from Pontryagin's Maximum Principle or direct transcription approaches such as collocation and pseudospectral methods, which convert the continuous-time optimal control problem into a large-scale nonlinear programming problem [3, 11, 2].

These classical methods are theoretically rigorous and widely used in practice, but they rely on repeated numerical integration of nonlinear ordinary differential equations and the solution of large constrained optimization problems. As a result, they can be computationally expensive, sensitive to initialization, and difficult to embed within higher-level design loops or real-time decision-making frameworks. This limitation has motivated growing interest in surrogate modelling techniques that can approximate system dynamics while remaining computationally efficient and differentiable.

In parallel, advances in scientific machine learning have introduced physics-informed neural networks (PINNs) as a framework for incorporating physical laws directly into neural network training. Originally proposed by Raissi et al. [10], PINNs augment standard data-driven loss functions with penalties enforcing governing differential equations, enabling neural networks to learn solutions that are consistent with known physics. Subsequent work has extended this idea to a wide

range of forward and inverse problems in physics and engineering, highlighting both the promise and the challenges of the approach [5, 4].

Together, these developments suggest an opportunity to revisit rocket ascent modelling from a new perspective, in which neural networks are not used purely as black-box approximators, but as physics-guided surrogates that retain computational efficiency, differentiability, and physical structure while reducing reliance on repeated numerical integration.

1.1.1 State of the Art

Recent research in physics-informed machine learning has demonstrated the potential of PINNs to model complex dynamical systems governed by ordinary and partial differential equations. Comprehensive surveys have summarized the strengths and limitations of the PINN framework, including challenges related to optimization stiffness, loss balancing, and generalization behaviour [5, 4]. Analytical studies have further examined PINN training dynamics and failure modes, emphasizing the importance of architectural design and loss construction for practical success [14, 8].

In aerospace applications, surrogate modelling has traditionally relied on response surfaces, reduced-order models, or purely data-driven neural networks. While such models can be effective within narrow operating regimes, they often struggle to extrapolate beyond the training domain and provide limited physical interpretability. PINNs offer a middle ground by combining data fidelity with physics-based regularization, making them particularly attractive for modelling rocket ascent dynamics, where governing equations are well understood but computationally expensive to evaluate repeatedly.

Despite this promise, the application of PINNs to constrained optimal control problems remains relatively underexplored, especially for rocket ascent. Existing studies predominantly focus on forward or inverse dynamics problems, while fewer

works investigate structured PINN surrogates as replacements for high-fidelity dynamics models within trajectory optimization pipelines. This gap is especially pronounced for six-degree-of-freedom (6-DOF) ascent problems, which involve strong nonlinearities, coupling between translational and rotational motion, and mass depletion effects.

1.1.2 Rationale

The central motivation of this thesis is to investigate whether a structured physics-informed neural network can serve as a differentiable surrogate for rocket ascent dynamics. Rather than repeatedly integrating nonlinear ordinary differential equations, the surrogate aims to approximate the full state evolution directly as a function of time and mission parameters, while remaining consistent with the underlying physical laws.

Such surrogates are particularly appealing for ascent analysis and preliminary design studies, where rapid evaluation of trajectory behaviour and sensitivity to physical parameters is required. By shifting the computational burden to an offline training phase, the learned model can potentially support future optimization and guidance frameworks that require fast, smooth, and physically consistent approximations of rocket dynamics.

1.1.3 Problem Statement

The problem addressed in this thesis is the development of a physics-informed neural network surrogate capable of approximating the three-dimensional ascent dynamics of a rocket during the launch phase. The surrogate is trained on full-state trajectories generated from high-fidelity numerical simulations of a 6-DOF rigid-body dynamics model and is designed to reproduce the coupled translational, rotational, and mass-depletion behaviour of rocket ascent.

While the broader objective is to support constrained trajectory optimization,

the focus of this study is on the accurate and physically consistent approximation of ascent dynamics, rather than on the direct computation of optimal control inputs using the surrogate.

1.1.4 Objectives

The specific objectives of this thesis are as follows:

1. To formulate a three-dimensional rocket ascent model suitable for physics-informed learning.

2. To design a structured PINN architecture incorporating Fourier time embeddings, context encoding, and residual multilayer perceptrons.

3. To train the surrogate using full-state trajectory data generated from high-fidelity numerical simulations.

4. To evaluate the accuracy, stability, and physical consistency of the learned surrogate during the launch phase.

5. To assess the suitability of the surrogate as a building block for future integration into constrained trajectory optimization frameworks.

1.1.5 Scope and Limitations

The scope of this work is limited to three-dimensional rocket ascent dynamics without wind disturbances. The surrogate is trained on trajectory data covering the initial 30 seconds of flight, corresponding to the launch phase. As a result, conclusions regarding full-ascent modelling, closed-loop control optimization, and generalization beyond the training regime are outside the scope of this thesis.

In particular, the study does not demonstrate closed-loop optimization of control inputs using the learned surrogate. These limitations are acknowledged and discussed as directions for future work.

1.1.6 Research Framework

The research framework follows a multi-level structure. At the data generation level, high-fidelity numerical simulations and optimal control solvers are used to produce dynamically feasible ascent trajectories. At the learning level, these trajectories are used to train a physics-informed neural network surrogate that approximates state evolution over time. At the application level, the trained surrogate is evaluated as a candidate replacement for numerical integration in future optimization and guidance workflows.

1.1.7 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 reviews the theoretical background, related work in physics-informed neural networks and optimal control, they are rocket dynamics model and problem formulation, describes the PINN architecture, training methodology, and surrogate evaluation then describes the methodological workflow. Chapter 3 presents the results of the study. Chapter 4 discusses the findings and limitations of the study. Chapter 5 concludes the thesis and outlines directions for future research.

CHAPTER II

MATERIALS AND METHODS

2.1 Overview of the Methodological Workflow

The methodology was organized around three sequential stages. First, a physical model of six-degree-of-freedom (6-DOF) rocket dynamics was formulated and implemented numerically. Second, optimal control trajectories were generated using direct collocation methods and used to construct training datasets. Third, physics-informed neural network (PINN) surrogates were trained on these datasets to approximate the dynamics. The relationship between the optimal control solver, truth integrator, and PINN surrogate is illustrated schematically in Figure 2.1.

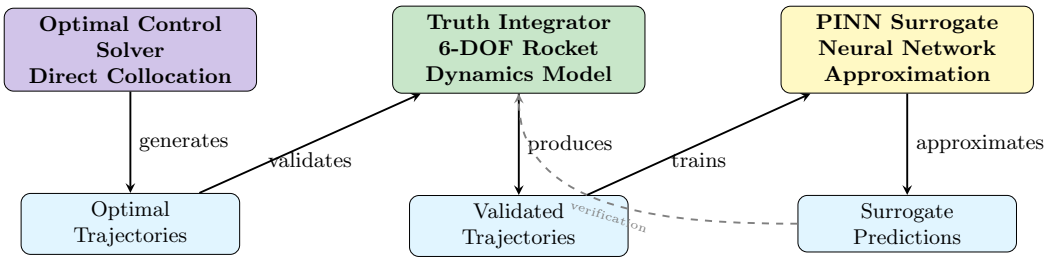

Figure 2.1: Schematic diagram illustrating the relationship between the optimal control solver, truth integrator, and PINN surrogate in the methodological workflow.

2.2 Rocket Dynamics Model

2.2.1 Reference Frames and Coordinate Systems

Two primary reference frames were used. The inertial frame was defined as Earth-centred and non-rotating, with the z-axis pointing upward (opposite to gravity). The body frame was defined as vehicle-fixed, with the x-axis aligned with the nominal thrust direction. Transformations between frames were represented using unit

quaternions.

The state vector $\mathbf{x}(t) \in \mathbb{R}^{14}$ was defined as

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{r}_{\mathrm{i}}^{\top} & \mathbf{v}_{\mathrm{i}}^{\top} & \mathbf{q}^{\top} & \boldsymbol{\omega}_{\mathrm{b}}^{\top} & m \end{bmatrix}^{\top},$$

where $\mathbf{r}_{\mathrm{i}} \in \mathbb{R}^3$ was the position in the inertial frame, $\mathbf{v}_{\mathrm{i}} \in \mathbb{R}^3$ was the inertial velocity, $\mathbf{q} = [q_0, q_1, q_2, q_3]^{\top}$ was a unit quaternion representing the attitude (body-to-inertial rotation), $\boldsymbol{\omega}_{\mathrm{b}} \in \mathbb{R}^3$ was the angular velocity expressed in the body frame, and $m \in \mathbb{R}$ was the mass of the vehicle.

The control vector $\mathbf{u}(t) \in \mathbb{R}^4$ was defined as

$$\mathbf{u}(t) = \begin{bmatrix} T & \theta_{\mathrm{g}} & \phi_{\mathrm{g}} & \delta \end{bmatrix}^{\top},$$

where T was the commanded thrust magnitude, θ_{g} and ϕ_{g} were the thrust gimbal pitch and yaw angles, and δ was a representative control surface deflection.

The body-to-inertial rotation matrix $R_{\mathrm{b}\to\mathrm{i}}(\mathbf{q})$ was obtained from the quaternion $\mathbf{q}$. Its transpose $R_{\mathrm{i}\to\mathrm{b}} = R_{\mathrm{b}\to\mathrm{i}}^{\top}$ transformed vectors from the inertial frame to the body frame.

2.2.2 Equations of Motion

The translational motion was governed by Newton's second law:

$$\dot{\mathbf{r}}_{\mathrm{i}} = \mathbf{v}_{\mathrm{i}}, \qquad \dot{\mathbf{v}}_{\mathrm{i}} = \frac{1}{m}\,\mathbf{F}_{\mathrm{i}}(\cdot) + \mathbf{g}_{\mathrm{i}}(\mathbf{r}_{\mathrm{i}}),$$

where $\mathbf{F}_{\mathrm{i}}$ was the total non-gravitational force in the inertial frame and $\mathbf{g}_{\mathrm{i}}$ was the gravitational acceleration. A constant gravitational field $\mathbf{g}_{\mathrm{i}} = [0, 0, -g_0]^{\top}$ was employed, where $g_0 = 9.81$ m/s^2 was the nominal surface gravity.

Rotational dynamics were expressed as

$$\dot{\mathbf{q}} = \tfrac{1}{2}\,\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{\mathrm{b}} \end{bmatrix}, \qquad \dot{\boldsymbol{\omega}}_{\mathrm{b}} = \mathbf{I}_{\mathrm{b}}^{-1}\Big(\mathbf{M}_{\mathrm{b}} - \boldsymbol{\omega}_{\mathrm{b}} \times (\mathbf{I}_{\mathrm{b}}\boldsymbol{\omega}_{\mathrm{b}})\Big),$$

where $\mathbf{I}_{\mathrm{b}}$ was the inertia tensor in the body frame and $\mathbf{M}_{\mathrm{b}}$ was the sum of aerodynamic and thrust moments.

The mass dynamics followed the standard rocket equation [12]:

$$\dot{m} = -\frac{T}{I_{\mathrm{sp}} g_0},$$

with specific impulse I_{sp} and surface gravity g_0. A dry-mass limit was enforced in the implementation to prevent unphysical mass depletion.

These equations follow standard formulations for rigid-body rocket dynamics [3, 2].

2.2.3 Aerodynamic and Propulsion Models

The atmosphere was modelled by an exponential density profile:

$$\rho(h) = \rho_0 \exp\!\left(-\frac{h}{h_{\mathrm{scale}}}\right),$$

where h was altitude above sea level, $\rho_0 = 1.225$ kg/m^3 was the sea-level density, and $h_{\mathrm{scale}} = 8400$ m was a scale height. The dynamic pressure was computed as

$$q_{\mathrm{dyn}} = \frac{1}{2}\,\rho(h)\,\|\mathbf{v}_{\mathrm{rel,b}}\|^2,$$

where $\mathbf{v}_{\mathrm{rel,b}}$ was the relative wind velocity in the body frame.

Aerodynamic forces were naturally expressed in the body frame. The relative

wind velocity was computed as

$$\mathbf{v}_{\text{rel,i}} = \mathbf{v}_{\text{i}} - \mathbf{v}_{\text{wind,i}}, \qquad \mathbf{v}_{\text{rel,b}} = R_{\text{i}\to\text{b}}(\mathbf{q})\,\mathbf{v}_{\text{rel,i}},$$

where $\mathbf{v}_{\text{wind,i}}$ was the wind velocity in the inertial frame. Wind disturbances were neglected, so $\mathbf{v}_{\text{wind,i}} = \mathbf{0}$.

Drag and lift forces in the body frame took the form

$$\mathbf{F}_{\text{D,b}} = -q_{\text{dyn}} S_{\text{ref}} C_{\text{D}}\,\hat{\mathbf{v}}_{\text{rel,b}}, \qquad \mathbf{F}_{\text{L,b}} = q_{\text{dyn}} S_{\text{ref}} C_{\text{L}\alpha}\,\alpha\,\hat{\mathbf{e}}_{\text{L}},$$

where S_{ref} was a reference area, C_{D} was the drag coefficient, $C_{\text{L}\alpha}$ was a lift-curve slope, α was the angle of attack computed from $\mathbf{v}_{\text{rel,b}}$, and $\hat{\mathbf{e}}_{\text{L}}$ was a unit lift direction approximately perpendicular to the body x-axis and the relative velocity.

The thrust vector in the body frame was

$$\mathbf{u}_T = \begin{bmatrix} \cos\theta_{\text{g}} \cos\phi_{\text{g}} \\ \sin\phi_{\text{g}} \\ \sin\theta_{\text{g}} \cos\phi_{\text{g}} \end{bmatrix}, \qquad \mathbf{F}_{\text{T,b}} = T\,\frac{\mathbf{u}_T}{\|\mathbf{u}_T\|},$$

so that the total non-gravitational force in the body frame was

$$\mathbf{F}_{\text{b}} = \mathbf{F}_{\text{T,b}} + \mathbf{F}_{\text{D,b}} + \mathbf{F}_{\text{L,b}}, \qquad \mathbf{F}_{\text{i}} = R_{\text{b}\to\text{i}}(\mathbf{q})\,\mathbf{F}_{\text{b}}.$$

2.3 Numerical Trajectory Generation via Optimal Control

2.3.1 Optimal Control Problem Formulation

Numerically optimal ascent trajectories were generated by solving a constrained optimal control problem (OCP), The OCP was discretised using a direct collocation method based on the Hermite–Simpson scheme [2, 11]. Which served as a data-generation mechanism for training and evaluating the physics-informed surrogate

model. The OCP was formulated as a continuous-time optimal control problem over a finite horizon $t \in [0, t_f]$, with the 6-DOF dynamics as constraints. The objective function was formulated to minimise fuel consumption, subject to path and terminal constraints on altitude, velocity, attitude, and structural loads. The time horizon was fixed at $t_f = 30$ s.

The control vector $\mathbf{u}(t) \in \mathbb{R}^4$ was defined as

$$\mathbf{u}(t) = \begin{bmatrix} T & \theta_{\mathrm{g}} & \phi_{\mathrm{g}} & \delta \end{bmatrix}^\top,$$

with bounds on thrust magnitude $T \in [0, T_{\max}]$, gimbal angles $\theta_{\mathrm{g}}, \phi_{\mathrm{g}} \in [-\theta_{\max}, \theta_{\max}]$, and control surface deflection $\delta \in [-\delta_{\max}, \delta_{\max}]$. Path constraints enforced maximum dynamic pressure $q_{\mathrm{dyn}} \leq q_{\mathrm{dyn,max}}$ and maximum load factor $n \leq n_{\max}$.

The OCP was discretised using a direct collocation method. The continuous state and control trajectories were approximated on a uniform grid of nodes $\{t_k\}_{k=0}^{N}$ with step size $h = t_f/N$. At each node, the state $x_k \approx x(t_k)$ and control $u_k \approx u(t_k)$ became decision variables in a finite-dimensional nonlinear programme (NLP). The dynamics were enforced via Hermite–Simpson collocation, which provided third-order accuracy by introducing a collocation point at the midpoint of each interval and enforcing a defect constraint of the form

$$x_{k+1} = x_k + \frac{h}{6}\big(f(x_k, u_k) + 4f(x_m, u_m) + f(x_{k+1}, u_{k+1})\big),$$

where $f(x, u)$ denotes the state derivative, x_m is a midpoint state constructed from (x_k, x_{k+1}) and $(f(x_k, u_k), f(x_{k+1}, u_{k+1}))$, and u_m is the midpoint control. The resulting collocation defects were enforced as equality constraints in the NLP.

2.3.2 Numerical Solvers and Software

The 6-DOF dynamics, including aerodynamics, thrust, and mass depletion, were implemented symbolically in CasADi. This provided exact Jacobians and Hessians

to the NLP solver, which was crucial for robustness and performance in the presence of stiff dynamics and tight path constraints. Special care was taken in the implementation to avoid non-differentiabilities and division by small quantities, for example by using smooth norm approximations and explicit clamping of mass and thrust.

CasADi version 3.6.7 and IPOPT version 3.14.16 were used in all numerical optimization experiments, with linear solver backends selected automatically based on availability.

The discretised OCP was solved using IPOPT, a large-scale interior-point nonlinear optimiser. State and control variables were scaled using reference length, velocity, time, mass, and force scales for conditioning, and linear solver backends (such as MUMPS or HSL variants) were selected automatically depending on availability. The framework supported both fixed and free final time, with appropriate bounds on t_f when treated as a decision variable.

Path constraints on dynamic pressure q_{dyn}, load factor n, and mass were enforced directly in the NLP using auxiliary CasADi functions for these quantities. Operational limits on gimbal angles, control-surface deflections, and angle of attack were also encoded in the state and control bounds.

In addition to the collocation-based OCP solver, a high-fidelity C++ integrator for the same 6-DOF dynamics was implemented and used as a truth model for validation and data generation. This integrator supported inverse-square gravity, exponential atmosphere, aerodynamic forces and moments, wind callbacks, and detailed diagnostic outputs such as dynamic pressure, load factor, and constraint violations.

For data-driven approximation, a differentiable version of the dynamics and the physics-informed neural network models were implemented using PyTorch [9] version 2.x (or LibTorch 2.3.1 as an C++ frontend for PyTorch). This module mirrored the CasADi model, including the state and control definitions, aerodynamic coefficients, and mass dynamics, but used tensor operations to enable automatic differen-

tiation.

The time discretization was fixed at $N = 1501$ points, corresponding to a 30-second flight duration sampled at 50 Hz.

2.4 Dataset Construction and Preprocessing

2.4.1 Dataset Structure

The data pipeline produced a hierarchy of datasets. At the lowest level, raw cases were stored as individual HDF5 files containing time grids, full 14-D state trajectories, control histories, monitor signals (e.g., dynamic pressure and load factor), and rich metadata including physical parameters, solver statistics, and configuration hashes. These raw cases were then aggregated into processed split files (train/validation/test) in nondimensional form.

The processed datasets contained approximately 120 training cases, 20 validation cases, and 20 test cases. Each trajectory was discretised on a uniform time grid with $N = 1501$ points, corresponding to a 30-second flight duration sampled at 50 Hz. Each processed dataset exposed a time grid, a context vector $\mathbf{c} \in \mathbb{R}^7$ encoding key physical and environmental parameters (initial mass m_0, specific impulse I_{sp}, drag coefficient C_{D}, lift-curve slope $C_{\text{L}\alpha}$, pitch-moment coefficient $C_{\text{m}\alpha}$, maximum thrust T_{max}, and wind magnitude), and normalised state targets.

The state tensor shape was state $\in \mathbb{R}^{N_{\text{cases}} \times N_{\text{time}} \times 14}$, where N_{cases} is the number of trajectories, $N_{\text{time}} = 1501$ is the number of time points, and 14 is the state dimension.

2.4.2 Nondimensionalization and Scaling

All state and control variables were nondimensionalized using physics-aware reference scales for numerical conditioning and intended to be scale-invariant learning. The reference scales were chosen to yield quantities of order unity, as summarized in Table 2.1.

Table 2.1: Reference scales for nondimensionalization

Quantity	Symbol	Value	Motivation
Length	L_{ref}	10,000 m	Typical altitude
Velocity	V_{ref}	313 m/s	$\sqrt{g_0 L_{\text{ref}}}$
Time	T_{ref}	31.62 s	$L_{\text{ref}}/V_{\text{ref}}$
Mass	M_{ref}	50 kg	Nominal wet mass
Force	F_{ref}	490 N	$M_{\text{ref}} g_0$

The scaling equation was

$$\tilde{x} = \frac{x}{x_{\text{ref}}},$$

where $\tilde{x}$ is the nondimensionalized quantity and x_{ref} is the corresponding reference scale. This nondimensionalization ensured that all state components were typically in the range $[0.1, 10]$, which improved gradient flow during neural network training and reduced numerical errors in the optimization solver. The scaling was inverted during evaluation to recover dimensional quantities. This nondimensionalization follows standard practice in numerical simulation for conditioning and training stability [1].

2.5 Physics-Informed Neural Network Model

2.5.1 Model Inputs and Outputs

The model inputs were time t (a scalar) and a context vector $\mathbf{c} \in \mathbb{R}^7$ encoding physical and environmental parameters. The model output was the state vector $\mathbf{x}(t) \in \mathbb{R}^{14}$.

2.5.2 Network Architecture (Direction AN)

The Direction AN architecture was organized into three stages (see Figure 2.2). A shared stem first embedded the normalised time and context: time was expanded and encoded by Fourier features embedding [13] with 8 frequencies, while the context vector was passed through a small encoder (a shallow multilayer perceptron) and

then concatenated with the time embedding. This combined input was processed by a residual multilayer perceptron with 4 layers, hidden dimension 128, skip connections, tanh activation, and layer normalization, yielding a latent feature sequence $z(t, c)$ of fixed dimension along the trajectory.

On top of this shared latent representation, three mission branches specialized to different parts of the 14-D state. A translation branch with hidden dimensions $[128, 128]$ mapped z to position and velocity components $[\mathbf{r}_\mathrm{i}, \mathbf{v}_\mathrm{i}]$. A rotation branch with hidden dimensions $[256, 256]$ mapped z to quaternion and angular velocity components $[\mathbf{q}, \boldsymbol{\omega}_\mathrm{b}]$ with explicit quaternion renormalization. A mass branch with hidden dimension $[64]$ produced the mass trajectory $m(t)$. The concatenation of these branch outputs formed the predicted state $x_\theta(t)$.

2.5.3 Physics Residual Computation

Although control inputs were explicitly modeled in the optimal control formulation used for data generation, the trained physics-informed neural network surrogate did not take control variables as direct inputs. Consequently, control terms were omitted from the physics residual during training, and the surrogate learned the state evolution implicitly from the resulting state trajectories. This design choice was adopted in order to evaluate the ability of the surrogate to reproduce ascent dynamics without explicit control conditioning.

The physics residual was computed using finite difference derivative approximations, rather than automatic differentiation. For interior points, a central difference scheme was used:

$$\frac{dx}{dt} \approx \frac{x_{i+1} - x_{i-1}}{2\Delta t},$$

where $\Delta t = 0.02$ s is the time step (corresponding to 50 Hz sampling). For boundary points, forward difference (first point) and backward difference (last point) schemes were used.

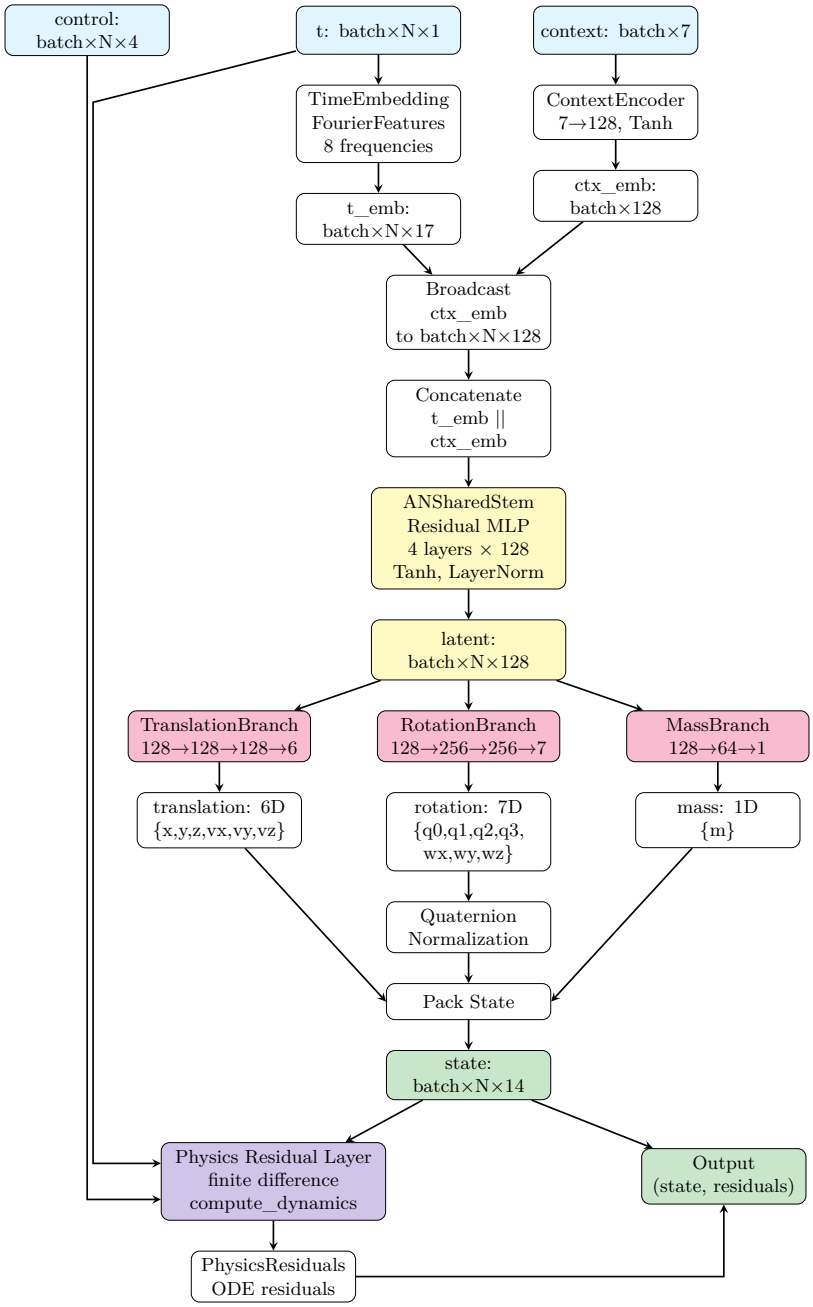

Figure 2.2: Architecture of the Direction AN model. The shared stem processes architecture of the Direction AN model. The shared stem processes time and context inputs to produce a latent representation $z(t, c)$. Three specialized branches (translation, rotation, mass) map this latent space to the corresponding state components. The physics residual layer computes dynamics residuals to enforce physical consistency.

In the PINN setting, explicit control inputs were omitted and set to zero in the physics residual, reflecting the design choice that the network learned state evolution implicitly from trajectories rather than from control commands. The control $\mathbf{u}_k$ in the physics loss was therefore set to zero ($\mathbf{u}_k = [0, 0, 0, 0]^\top$). This zero-control assumption was consistent with the processed dataset format, which did not include control trajectories.

2.5.4 Architecture Selection Procedure

The final Direction AN architecture was obtained through a structured model selection process. Several candidate architectures were evaluated to assess the impact of architectural choices on training stability, physical consistency, and trajectory reconstruction accuracy.

In particular, variants differing in the handling of temporal inputs, mission parameters, and output coupling were examined. Architectures combining time and context inputs at a single shallow layer were evaluated and not retained due to unstable convergence behavior and sensitivity to scaling. In contrast, architectures employing separate encoders for temporal inputs and mission parameters, followed by a shared latent representation, showed improved convergence behavior and more stable physics residuals. These architectures were retained for further evaluation.

Additionally, separating the output into distinct heads for translational states, rotational states, and mass evolution was evaluated and found to facilitate loss balancing and improve interpretability of individual error components. Based on these observations, a shared-stem architecture with specialized output branches was selected as the final Direction AN configuration, as illustrated in Figure 2.3.

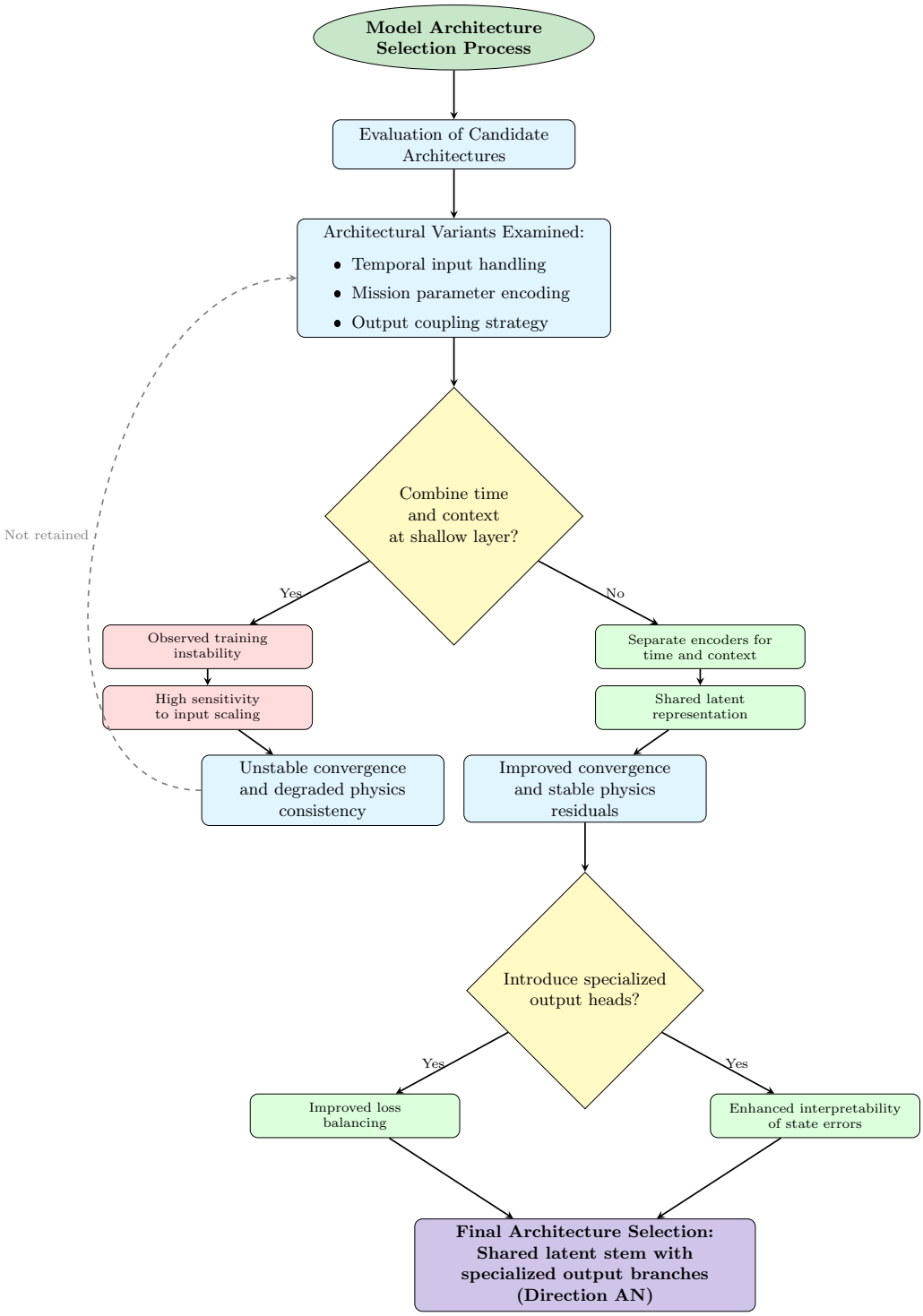

Figure 2.3: Model selection process for the Direction AN architecture. Summarizes the selection process used to refine the network architecture. Architectural choices were guided by observed training stability, sensitivity to input scaling, and the ability to balance heterogeneous loss components, rather than by exhaustive hyperparameter search.

2.6 Training Procedure

2.6.1 Loss Function Definition

The total training loss was constructed as a weighted combination of multiple terms enforcing data fidelity, physical consistency, and structural constraints. Rather than relying on a single objective, the loss function was designed to reflect different aspects of the rocket ascent dynamics following standard physics-informed learning formulations [10].

The primary loss components were grouped into four categories:

1. Data fidelity losses, which penalized deviations between predicted and reference state trajectories obtained from numerical simulation.

2. Physics consistency losses, which penalized violations of the governing equations of motion, including translational dynamics, rotational kinematics, and mass depletion.

3. Structural constraint losses, which enforced properties intrinsic to the system representation, such as unit-norm quaternion constraints and monotonic mass depletion.

4. Regularization losses, which promoted smoothness of the predicted trajectories and suppressed high-frequency numerical artifacts.

The data loss L_{data} was a component-weighted mean-squared error between predicted and true states, with separate group weights for translation, rotation, and mass:

$$L_{\text{data}} = \lambda_{\text{translation}} L_{\text{translation}} + \lambda_{\text{rotation}} L_{\text{rotation}} + \lambda_{\text{mass}} L_{\text{mass}},$$

where $\lambda_{\text{translation}} = 1.0$, $\lambda_{\text{rotation}} = 1.0$, and $\lambda_{\text{mass}} = 1.0$ were the group weights, and $L_{\text{translation}}$, L_{rotation}, and L_{mass} were component-weighted MSE losses for the respective state groups.

The physics loss L_{phys} penalised violations of the continuous dynamics by comparing a finite-difference estimate of the time derivative $\dot{x}_\theta(t_k)$ to the right-hand side $f(x_\theta(t_k), \mathbf{u}_k = \mathbf{0}; \mathbf{p})$ of the 6-DOF model implemented in PyTorch:

$$L_{\text{phys}} = \frac{1}{N} \sum_{k=0}^{N-1} \left\| \dot{x}_\theta(t_k) - f(x_\theta(t_k), \mathbf{u}_k = \mathbf{0}; \mathbf{p}) \right\|^2.$$

A boundary term L_{bc} enforced agreement between predicted and true initial conditions:

$$L_{\text{bc}} = \|\mathbf{x}_\theta(0) - \mathbf{x}^\star(0)\|^2.$$

Additional loss terms were included to suppress non-physical lateral drift, enforce smooth ascent behavior, and stabilize training by penalizing high-frequency numerical artifacts, particularly during the early ascent phase. These loss terms are:

- $L_{\text{mass,res}}$: Mass residual loss ensuring mass decreases monotonically
- $L_{\text{vz,res}}$: Vertical velocity residual loss
- $L_{\text{vxy,res}}$: Horizontal velocity residual loss
- $L_{\text{smooth,z}}$: Temporal smoothing for altitude
- $L_{\text{smooth,vz}}$: Temporal smoothing for vertical velocity
- $L_{\text{pos,vel}}$: Position–velocity consistency loss
- $L_{\text{smooth,pos}}$: Position smoothing loss
- $L_{\text{zero,vxy}}$: Zero horizontal velocity loss
- $L_{\text{zero,axy}}$: Zero horizontal acceleration loss
- L_{hacc}: Horizontal acceleration constraint loss
- $L_{\text{xy,zero}}$: Zero horizontal position loss

The total loss was

$$\begin{aligned}\mathcal{L} = {} & \lambda_{\text{data}} L_{\text{data}} + \lambda_{\text{phys}} L_{\text{phys}} + \lambda_{\text{bc}} L_{\text{bc}} \\ & + \lambda_{\text{mass,res}} L_{\text{mass,res}} + \lambda_{\text{vz,res}} L_{\text{vz,res}} + \lambda_{\text{vxy,res}} L_{\text{vxy,res}} \\ & + \lambda_{\text{smooth,z}} L_{\text{smooth,z}} + \lambda_{\text{smooth,vz}} L_{\text{smooth,vz}} + \lambda_{\text{pos,vel}} L_{\text{pos,vel}} \\ & + \lambda_{\text{smooth,pos}} L_{\text{smooth,pos}} + \lambda_{\text{zero,vxy}} L_{\text{zero,vxy}} + \lambda_{\text{zero,axy}} L_{\text{zero,axy}} \\ & + \lambda_{\text{hacc}} L_{\text{hacc}} + \lambda_{\text{xy,zero}} L_{\text{xy,zero}},\end{aligned} \tag{2.1}$$

where the loss weights were set to $\lambda_{\text{data}} = 1.0$, $\lambda_{\text{phys}} = 0.1$, $\lambda_{\text{bc}} = 1.0$, $\lambda_{\text{mass,res}} = 0.05$, $\lambda_{\text{vz,res}} = 0.05$, $\lambda_{\text{vxy,res}} = 0.01$, $\lambda_{\text{smooth,z}} = 5.0 \times 10^{-4}$, $\lambda_{\text{smooth,vz}} = 1.0 \times 10^{-4}$, $\lambda_{\text{pos,vel}} = 1.0$, $\lambda_{\text{smooth,pos}} = 2.0 \times 10^{-3}$, $\lambda_{\text{zero,vxy}} = 1.0$, $\lambda_{\text{zero,axy}} = 1.0$, $\lambda_{\text{hacc}} = 0.02$, and $\lambda_{\text{xy,zero}} = 5.0$.

2.6.2 Optimization and Training Schedule

The network was trained using the Adam optimizer [6] with learning rate 1.0×10^{-3}, weight decay 1.0×10^{-5}, and batch size 8. Training was conducted for 160 epochs. A cosine annealing learning rate scheduler [7] was used with $T_{\text{max}} = 160$ and minimum learning rate $\eta_{\text{min}} = 1.0 \times 10^{-6}$.

A phased training schedule was employed where loss weights were adjusted over time. In the initial phase (approximately the first 55% of training epochs), the model focused on fitting the data and coarse physics constraints, with lower weights on higher-order regularization terms. In the second phase, these regularization weights were gradually increased using a cosine ramp schedule [7], which was designed to gradually increase the regularization weights over time, designing like this to allow the model to refine its predictions while maintaining physical consistency. This phased training schedule was designed to help the model refine its predictions while maintaining physical consistency.

Early stopping was implemented with patience 25 for phase 1 and patience 40 for

phase 2, with minimum delta 0.0. This early stopping was designed to prevent the model from overfitting to the regularization terms early in training while ensuring that physical constraints are properly enforced in the final model.

2.6.3 Evaluation Metrics

Model performance was evaluated using root mean square error (RMSE) computed on the test set. For a trained Direction AN model, predictions $\hat{\mathbf{x}}_\theta(t_k)$ were generated for all test trajectories on a uniform time grid $\{t_k\}_{k=0}^{N-1}$ spanning the trajectory duration (30 seconds). The total RMSE was computed as

$$\text{RMSE}_{\text{total}} = \sqrt{\frac{1}{B \cdot N \cdot D} \sum_{b=1}^{B} \sum_{k=0}^{N-1} \sum_{d=1}^{D} \left(\hat{x}_\theta^{(b)}(t_k)_d - x^{\star(b)}(t_k)_d \right)^2},$$

where $B = 20$ was the number of test trajectories, $N = 1501$ was the number of time points in the discretized trajectory grid (corresponding to a sampling rate of approximately 50 Hz), $D = 14$ was the state dimension, and $x^{\star(b)}(t_k)$ denoted the ground truth state for trajectory b at time t_k. The summation over k from 0 to $N - 1$ averaged the squared errors across all time points in each trajectory, while the summation over b averaged across all test trajectories, and the summation over d averaged across all state components.

Per-component RMSE was computed by averaging over trajectories and time points for each state dimension d, while aggregated metrics were computed for translation (components 1–6: position and velocity), rotation (components 7–13: quaternion and angular velocity), and mass (component 14). All RMSE values were reported in nondimensional units, reflecting the scaled state representation used during training.

2.6.4 Loss Weight Selection and Sensitivity Analysis

The relative weighting of loss components was selected using a combination of physical reasoning and empirical sensitivity analysis. Initial loss weights were chosen based on the expected physical significance and numerical scale of each state component, informed by exploratory analysis of the trajectory data. For example, altitude, vertical velocity, and mass evolution were assigned higher relative importance due to their dominant role in ascent performance.

To refine these initial choices, coarse parameter sweeps were conducted to identify ranges of loss weights that yielded stable training and physically plausible trajectories. Within these ranges, a lightweight evolutionary search procedure was employed to test local variations of loss weights and assess their effect on convergence behavior and validation error. This process was used to improve robustness rather than to identify a globally optimal configuration, as illustrated in Figure 2.4.

The final loss weights were selected based on a balance between data fidelity, physical consistency, and training stability, and were fixed for all reported experiments.

2.6.5 Implementation Details

The optimal control solver was implemented using CasADi for symbolic computation and IPOPT for nonlinear optimization. The truth integrator was implemented in C++. Random seeds were fixed for all experiments to ensure reproducibility. Data processing and training scripts were written in Python, using NumPy for numerical operations and HDF5 for data storage. The neural network models were implemented in PyTorch. Training was performed on a workstation with automatic device selection depending on availability. (CPU or GPU if available), and no hardware-specific optimizations were employed. This implementation was designed to be reproducible and scalable.

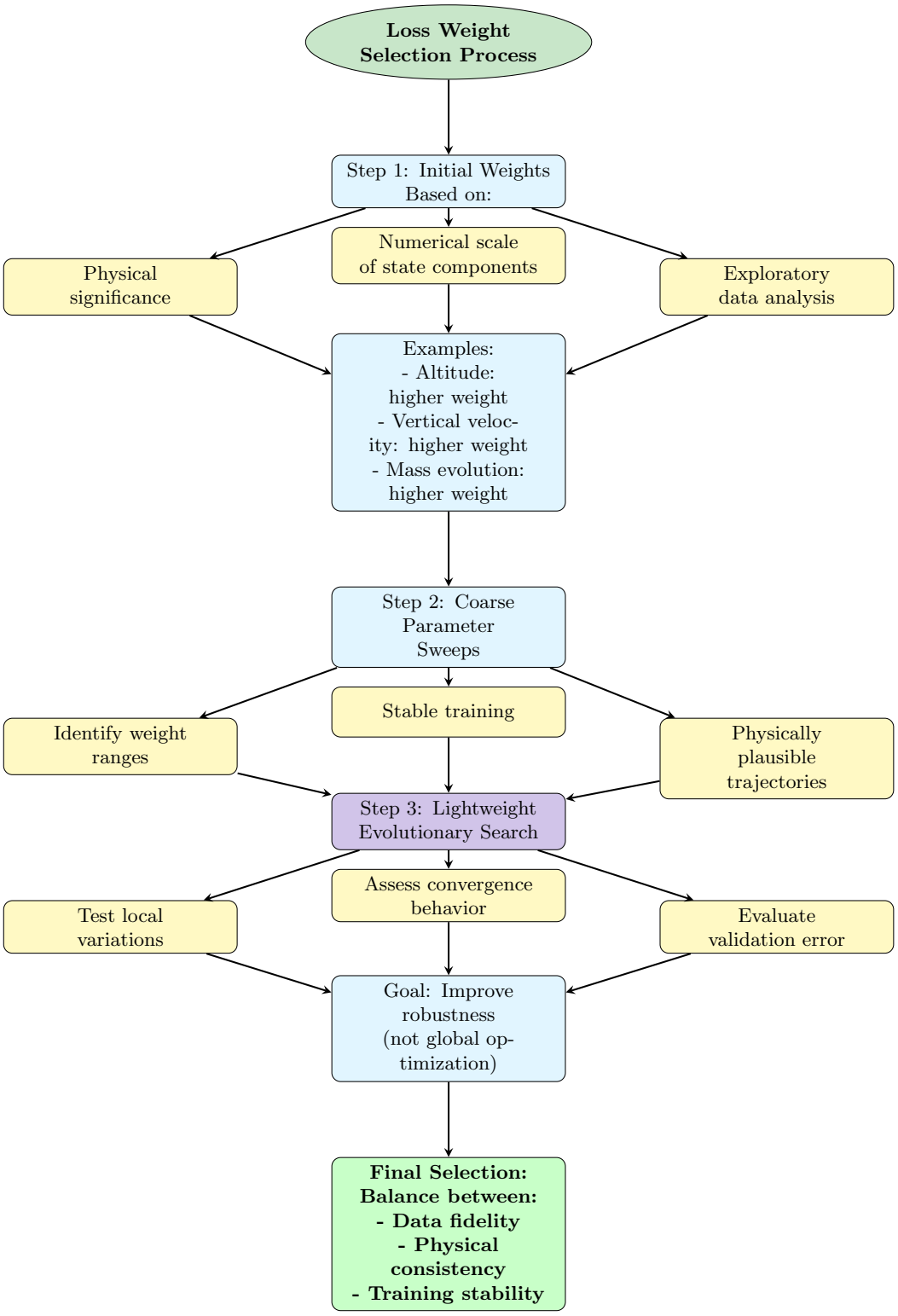

Figure 2.4: Loss weight selection process. The selection of loss weights followed a structured procedure summarized, combining physical intuition, sensitivity analysis, and local evolutionary refinement to ensure stable and interpretable training.

2.6.6 Summary of Chapter

A six-degree-of-freedom rocket dynamics model was formulated and implemented at three fidelity levels: a high-fidelity C++ truth integrator, a symbolic CasADi version for optimal control, and a differentiable PyTorch implementation for PINN training. Optimal control trajectories were generated using direct collocation with Hermite–Simpson discretization and solved using IPOPT. These trajectories were used to construct training datasets with 120 training cases, 20 validation cases, and 20 test cases, each discretized on a uniform time grid with 1501 points over 30 seconds. A physics-informed neural network architecture (Direction AN) was designed with a shared stem, three specialized branches for translation, rotation, and mass, and a physics residual layer. The network was trained using a phased schedule with Adam optimization, incorporating data fidelity, physics consistency, structural constraints, and smoothing regularization terms in the loss function.

CHAPTER III
RESULTS

3.1 Evaluation Setup

This chapter evaluates the performance of the proposed Physics-Informed Neural Network using a held-out test set that was not used during training. The objective is to assess trajectory reconstruction accuracy, quantitative error characteristics, and compliance with physical constraints.

The test dataset consists of multiple simulated rocket trajectories generated using a high-fidelity ordinary differential equation solver. Each trajectory spans a fixed time horizon of 30 seconds and includes translational states, rotational states represented by quaternions, and vehicle mass evolution. All trajectories are sampled at a uniform temporal resolution consistent with the training setup.

Prediction accuracy is evaluated by comparing the PINN outputs against the reference solver outputs at corresponding time instances. Absolute error is computed pointwise in time for representative trajectories, while root-mean-square error is used to quantify performance across the entire test set. All reported RMSE values are computed per trajectory and then aggregated across the test dataset.

No control inputs or reference trajectories are provided to the network during inference. The model predicts the full state evolution solely from the initial condition and time input.

3.2 Trajectory Reconstruction Results

This section presents a qualitative comparison between the PINN-predicted trajectory and the reference trajectory for a single representative test case. The purpose is to visually assess how well the model reconstructs the time evolution of different

state components.

3.2.1 Vertical Motion

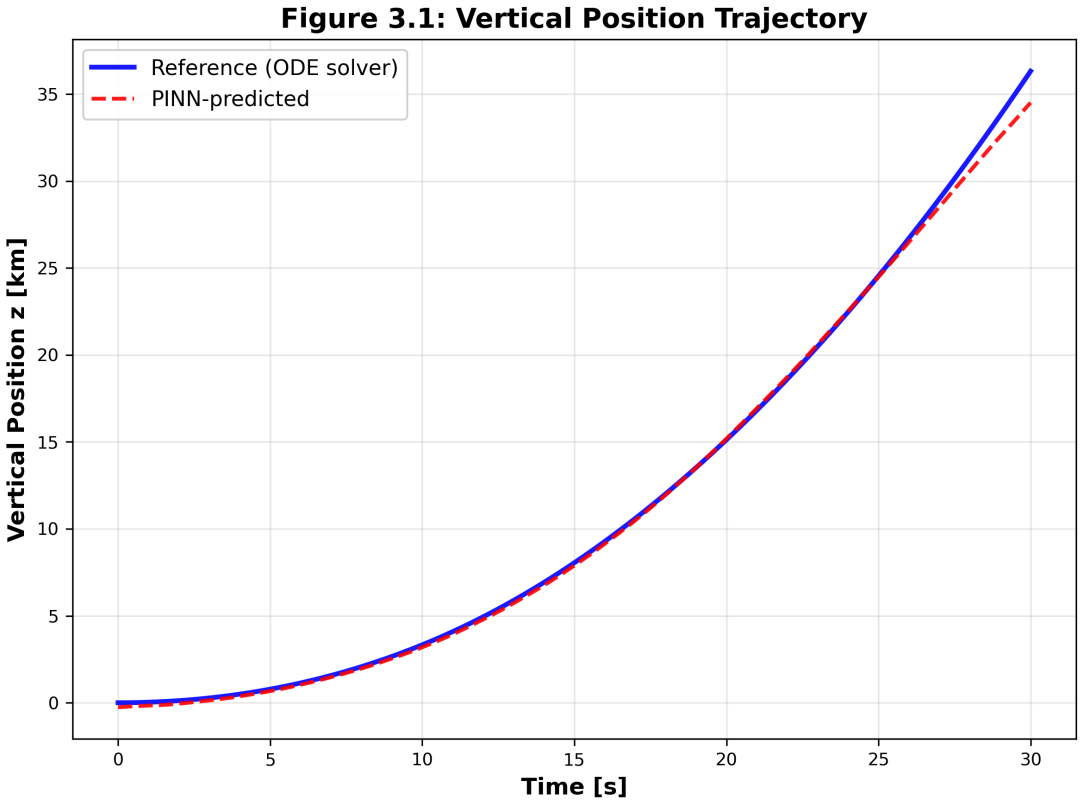

Figure 3.1: shows the vertical position as a function of time. The PINN prediction closely follows the reference solution over the full ascent phase, with a visible but small deviation near the end of the time horizon.

3.2.2 Mass evolution

3.2.3 Horizontal motion

3.2.4 Attitude representation

3.3 Quantitative Error Metrics

This section evaluates prediction accuracy across the entire test set using quantitative error measures.

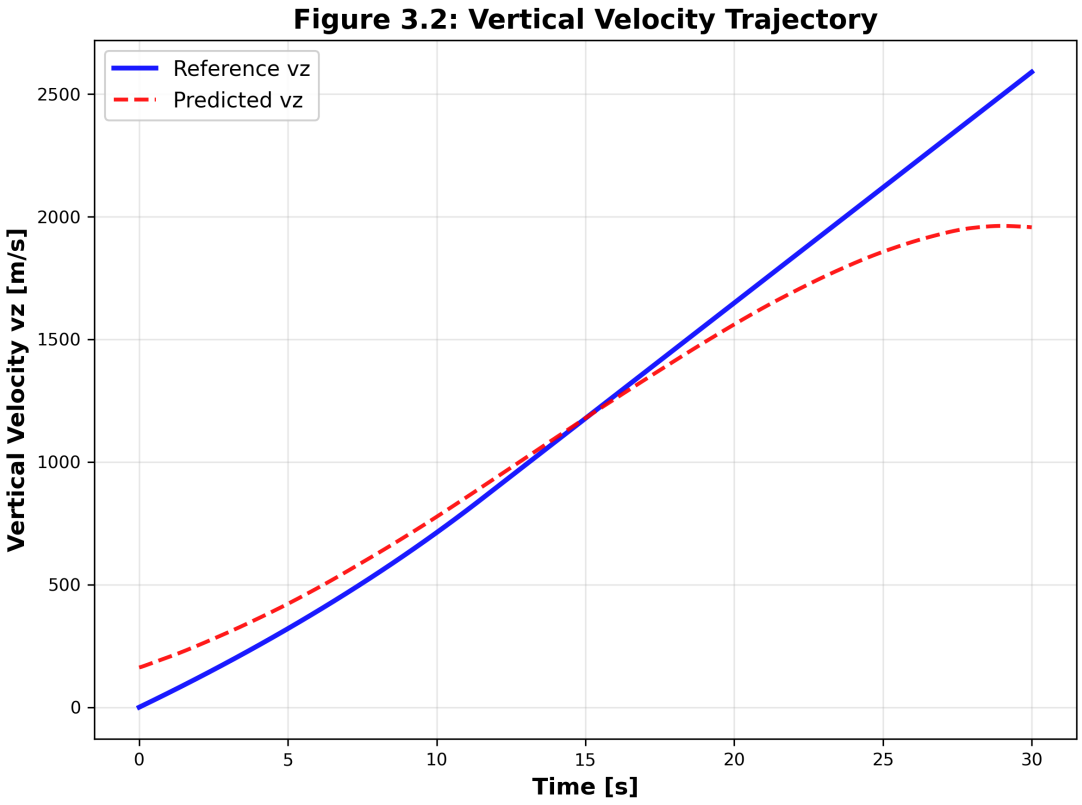

Figure 3.2: presents the corresponding vertical velocity profile. The predicted velocity captures the overall trend and curvature of the reference trajectory, with noticeable divergence during the later phase of motion.

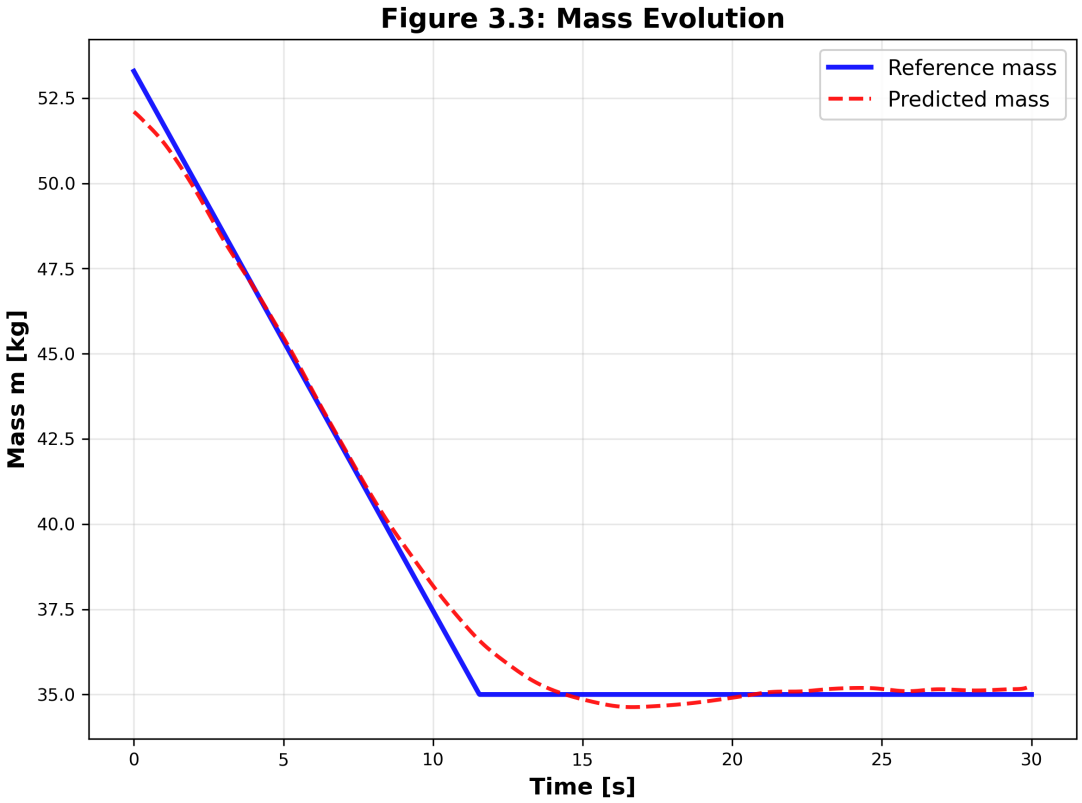

Figure 3.3: compares the predicted mass evolution against the reference solution. During the propulsive phase, the predicted mass follows the general decreasing trend of the reference. After propellant depletion, the reference mass remains constant, while the predicted mass exhibits small fluctuations around the terminal value.

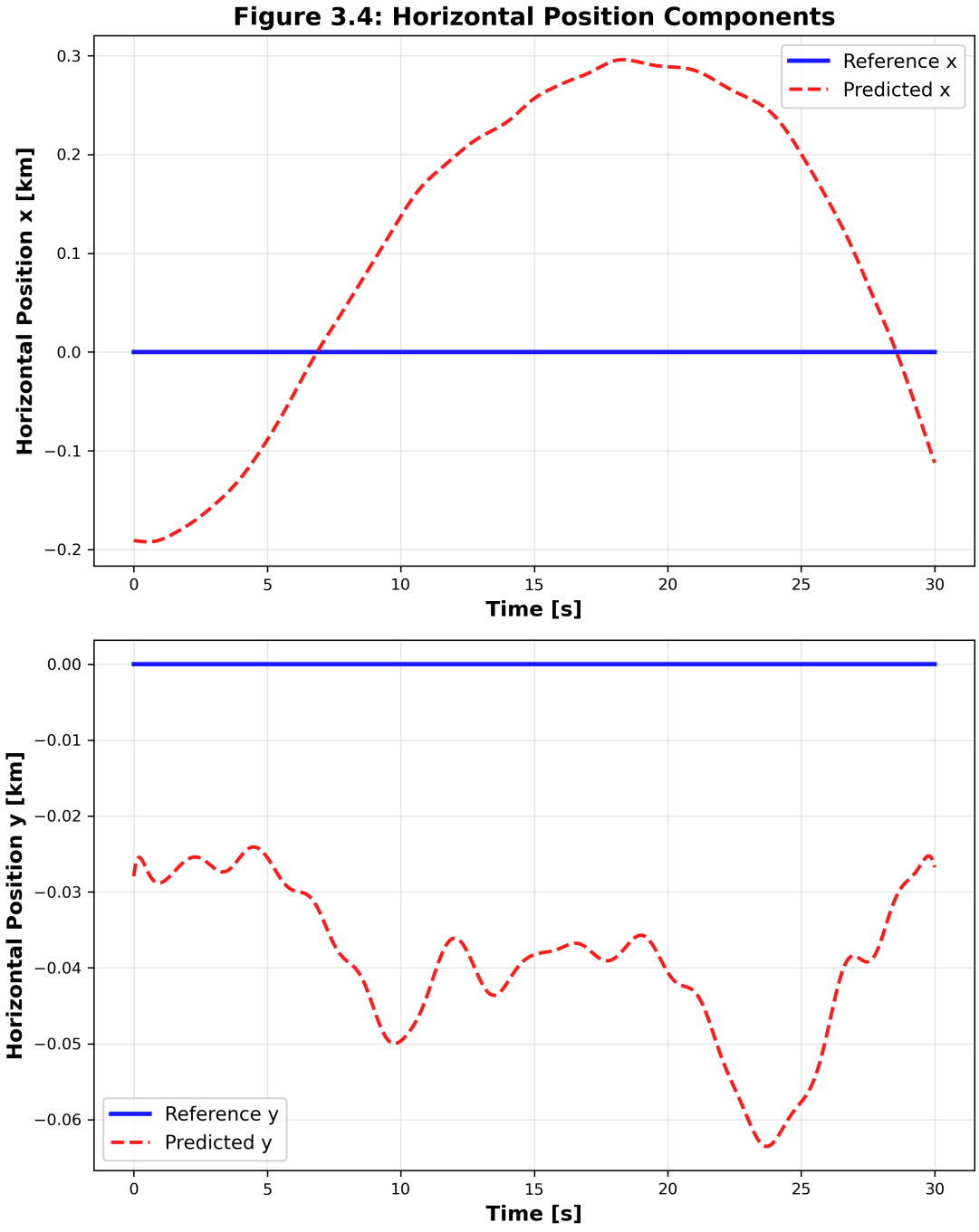

Figure 3.4: illustrates the horizontal position components. The reference trajectory remains at zero in both horizontal directions, while the predicted trajectory exhibits small deviations, indicating drift in lateral motion.

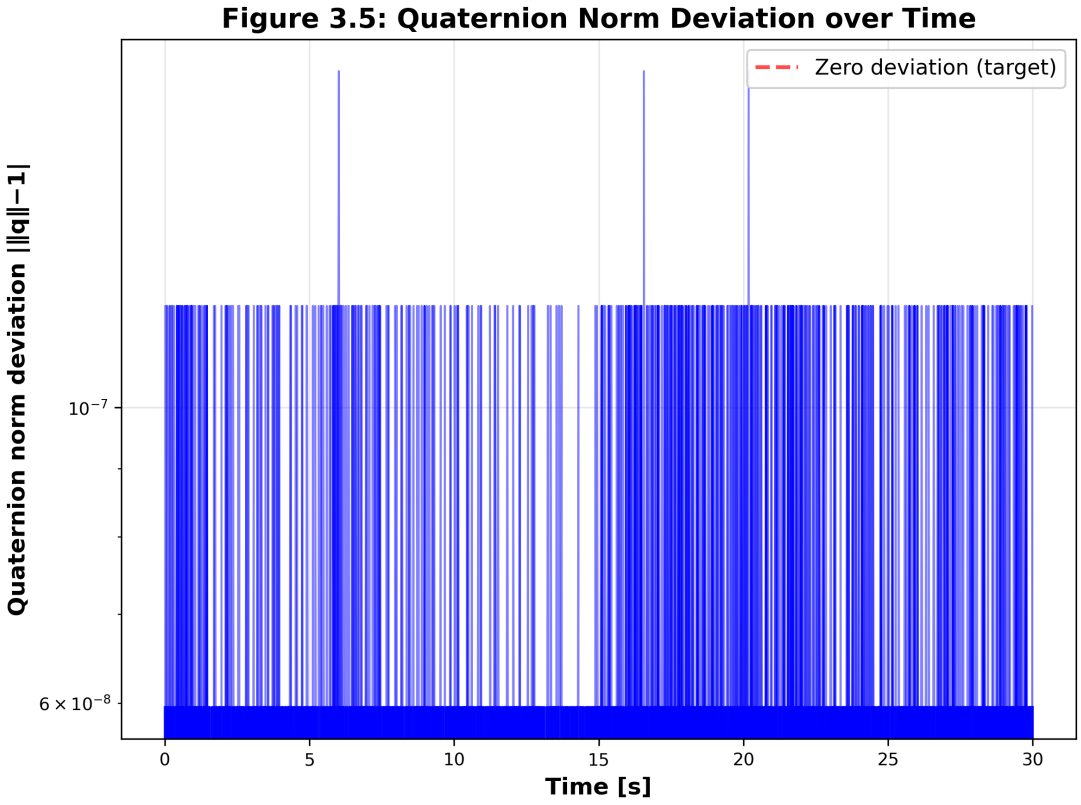

Figure 3.5: reports the deviation of the predicted quaternion norm from unity over time. The deviation remains extremely small throughout the trajectory, indicating that quaternion normalization is largely preserved.

State Variable	Mean RMSE	Standard Deviation	Units
x	0.019133	0.000854	nondim
y	0.002535	0.001278	nondim
z	0.065454	0.015009	nondim
vx	0.275908	0.017068	nondim
vy	0.002568	0.001137	nondim
vz	0.575095	0.080124	nondim
q0	0.072533	0.029386	nondim
q1	0.000960	0.000707	nondim
q2	0.335586	0.071237	nondim
q3	0.000720	0.000347	nondim
wx	0.000801	0.000251	nondim
wy	0.008613	0.001579	nondim
wz	0.000704	0.000175	nondim
m	0.012871	0.007033	nondim

Table 3.1: summarizes the root-mean-square error for each state variable, averaged across all test trajectories. Position, velocity, rotation, and mass errors are reported separately to reflect their different physical scales.

3.3.1 Per-state RMSE statistics

3.3.2 Aggregate performance measures

Metric	Observed Range	Test Set Size
Vertical position accuracy (RMSE)	0.067152 ± 0.015009	20
Velocity accuracy (RMSE)	0.580650 ± 0.080124	20
Mass behavior (RMSE)	0.014667 ± 0.007033	20
Quaternion norm range	[1.000000, 1.000000]	20
Residual magnitude range	[0.166489, 9.513170]	20

Table 3.2: provides a condensed summary of overall error statistics, including mean and dispersion measures across the test set. These metrics offer a compact overview of prediction accuracy beyond individual state variables.

3.3.3 RMSE distribution

3.4 Constraint Satisfaction Results

This section examines whether the PINN predictions satisfy key physical constraints imposed during training.

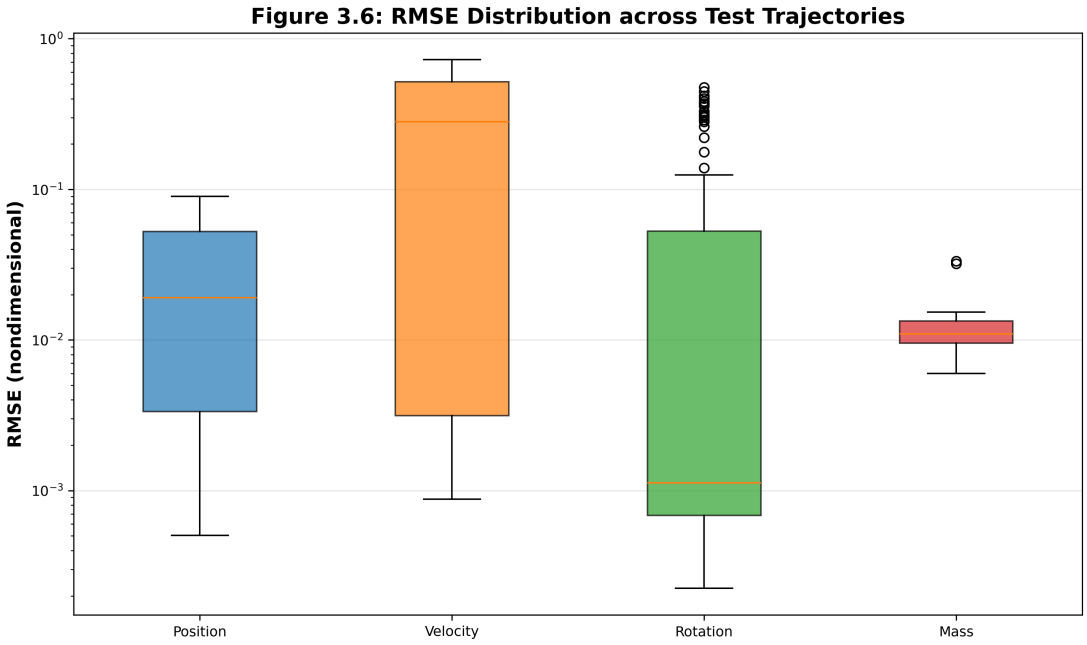

Figure 3.6: visualizes the distribution of RMSE values across all test trajectories for different state groups. The logarithmic scale highlights variability and outliers in prediction performance.

3.4.1 Physics residual norm

3.4.2 Mass monotonicity constraint

3.5 Summary of Observed Results

The results presented in this chapter demonstrate that the proposed PINN is capable of reconstructing the overall structure of rocket trajectories for a representative test case, as shown by close visual agreement in vertical position and velocity. Quantitative evaluation across the test set indicates that prediction errors vary by state type, with mass and rotational states exhibiting distinct error distributions.

Constraint satisfaction analysis reveals that quaternion normalization is largely maintained, while mass monotonicity violations occur in the predicted trajectories. Physics residual magnitudes remain bounded and decrease toward the end of the simulated time horizon.

All observations reported in this chapter are based solely on empirical evaluation

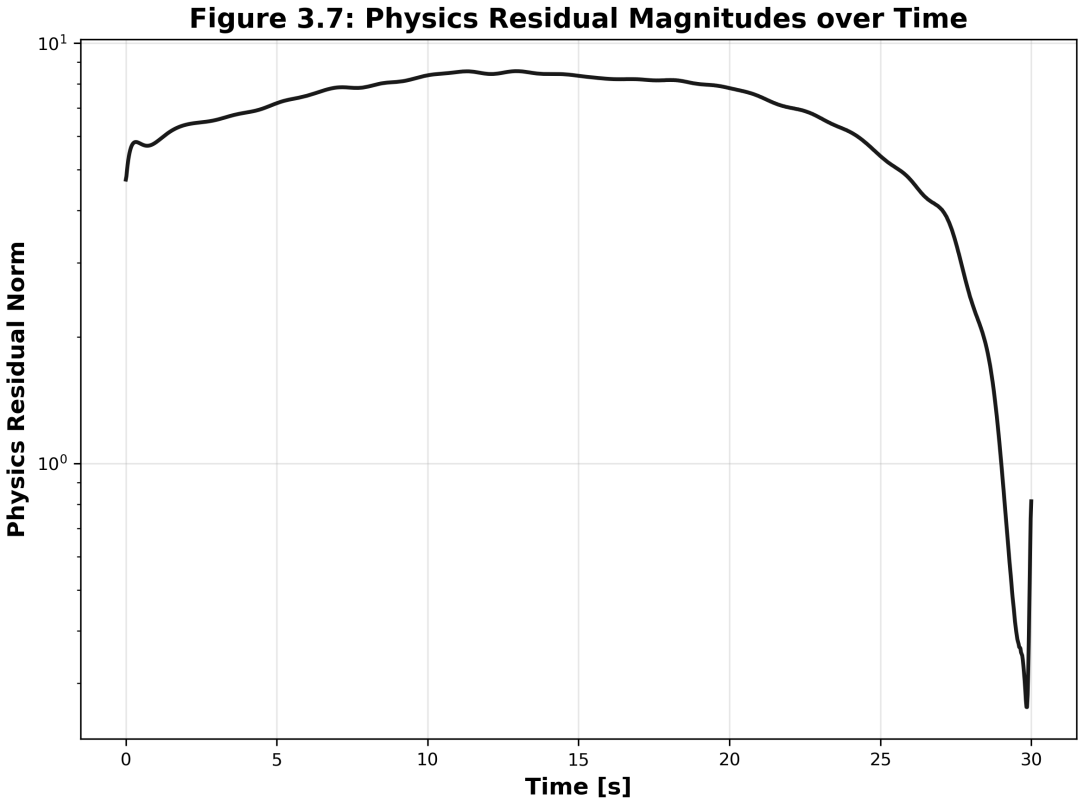

Figure 3.7: shows the time evolution of the physics residual norm for the representative trajectory. The residual magnitude varies over time and exhibits a notable decrease toward the end of the trajectory.

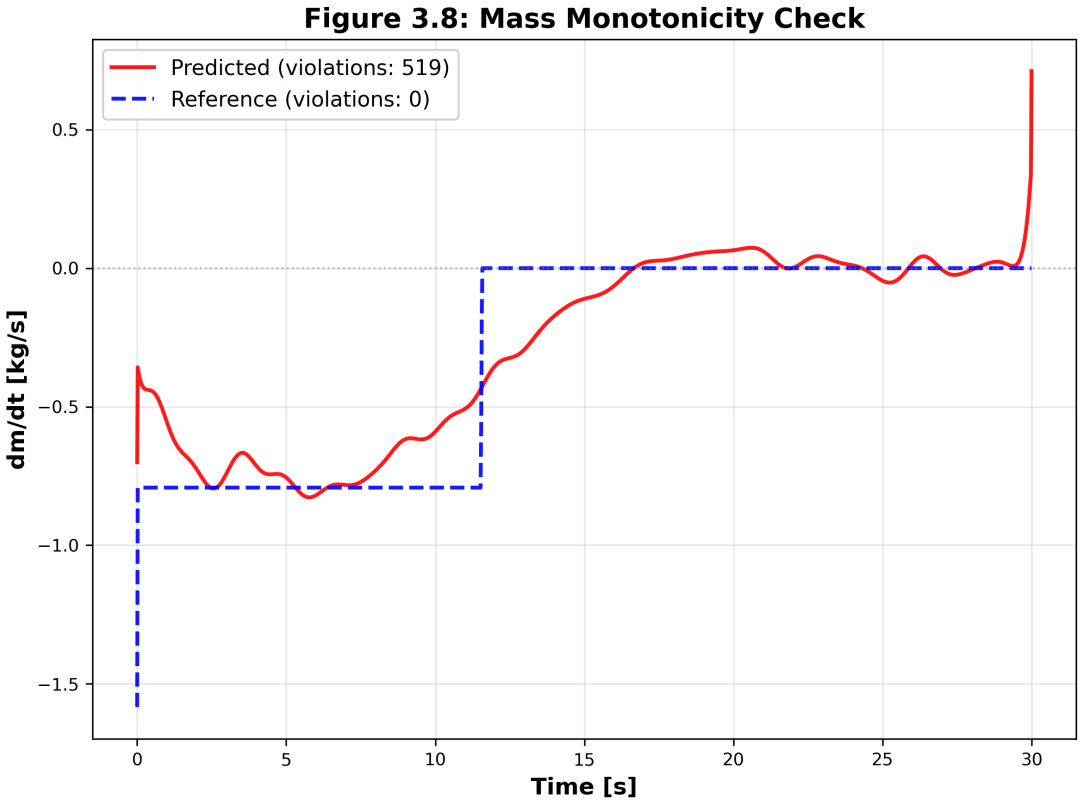

Figure 3.8: evaluates compliance with the mass monotonicity constraint by plotting the time derivative of mass. The reference trajectory shows no violations, while the predicted trajectory exhibits multiple instances where the monotonicity condition is violated.

results without interpretation of underlying causes.

CHAPTER IV
DISCUSSION AND IMPLICATIONS

4.1 Overall Performance of the Physics-Informed Model

The results demonstrate that the proposed Physics-Informed Neural Network is capable of reconstructing the global structure of rocket trajectories when evaluated on unseen test cases. The model successfully captures the dominant trends of vertical position, velocity, and mass evolution without explicit access to control inputs during inference. This confirms that embedding physical laws directly into the learning objective provides sufficient inductive bias for trajectory reconstruction in high-dimensional dynamical systems.

The close agreement observed in vertical position throughout the ascent phase indicates that the network effectively learns the integral effects of acceleration and thrust over time. This aligns with established findings in physics-informed learning, where state variables governed by smooth integral dynamics tend to be reconstructed more accurately than their time derivatives.

At the same time, deviations observed toward the end of the trajectory suggest that accumulated error remains a challenge, particularly in long-horizon predictions. This behavior is consistent with prior PINN studies applied to nonlinear ordinary differential equations, where error propagation becomes more pronounced as the temporal distance from the initial condition increases.

4.2 Interpretation of Velocity and Higher-Order Dynamics

The vertical velocity results reveal larger discrepancies compared to position, especially during later stages of flight. This outcome is expected, as velocity depends directly on first-order derivatives of the network output with respect to time. In

physics-informed models, derivative-based quantities are more sensitive to approximation error and numerical stiffness.

The divergence observed in the predicted velocity near the terminal time suggests that the network prioritizes global trajectory consistency over local derivative accuracy. This trade-off reflects a known characteristic of PINNs, where minimizing a combined data and physics loss does not guarantee uniform accuracy across all state derivatives.

From a physical perspective, velocity dynamics are influenced by rapidly changing forces, including thrust cutoff and mass variation. Capturing these transitions accurately requires either increased temporal resolution or stronger enforcement of the governing equations during these phases. The present implementation demonstrates reasonable performance but indicates that velocity prediction remains a limiting factor for precise trajectory reconstruction.

4.3 Mass Evolution and Constraints Enforcement

The mass evolution results show that the network learns the general trend of propellant consumption during the burn phase and the constant-mass regime after cutoff. However, small oscillations around the terminal mass and multiple monotonicity violations are observed in the predicted trajectory.

These violations indicate that while the mass conservation law is included as a physics residual, it is not enforced as a hard constraint. As a result, the model allows small physically inconsistent fluctuations if they reduce the overall loss. This behavior highlights an important limitation of soft constraint enforcement in PINNs.

In practical aerospace applications, mass monotonicity is a non-negotiable physical constraint. The current results suggest that additional structural constraints or post-processing steps are necessary if the model is to be deployed in safety-critical settings. Nonetheless, the ability of the network to approximate the mass profile without explicit integration demonstrates the flexibility of the physics-informed ap-

proach.

4.4 Horizontal Drift and Symmetry Breaking

The horizontal position results reveal small but persistent deviations from the reference trajectory, which remains identically zero. This indicates that the network does not strictly preserve symmetry in unconstrained dimensions.

Such drift is a known phenomenon in neural differential equation models, particularly when the governing equations admit trivial solutions. In the absence of explicit penalties or symmetry-enforcing constraints, the optimizer may converge to nearby solutions that satisfy the loss function but violate problem-specific invariances.

From a modeling standpoint, this observation suggests that additional regularization terms or coordinate-specific constraints may be required when zero-motion states are physically expected. Alternatively, reparameterizing the model to exclude irrelevant degrees of freedom could reduce spurious predictions and improve robustness.

4.5 Quaternion Normalization and Attitude Representation

The quaternion norm deviation results indicate that the predicted orientation remains extremely close to unit norm throughout the trajectory. This confirms that the normalization strategy used during training is effective at preserving valid attitude representations.

Maintaining quaternion normalization is critical for rotational dynamics, as even small deviations can lead to non-physical rotations when propagated over time. The present results show that the chosen formulation successfully mitigates this risk without introducing numerical instability.

This outcome aligns with existing literature on learning-based attitude modeling, where explicit normalization or penalty-based approaches are necessary to ensure

physically meaningful orientation states.

4.6 Physics Residual Behavior and Model Consistency

The physics residual analysis provides insight into how well the learned solution satisfies the governing equations. The residual magnitudes remain bounded throughout the trajectory and decrease toward the end of the time horizon.

This trend suggests that the network gradually converges toward a solution that is increasingly consistent with the physical model. However, the non-uniform residual profile indicates that constraint satisfaction varies across different flight phases.

Such behavior is typical in PINNs applied to nonlinear systems with time-varying dynamics. Residual dominance may shift between terms depending on the operating regime, highlighting the importance of adaptive loss weighting or curriculum-based training strategies.

4.7 Practical Implications and Applicability

The results indicate that physics-informed neural networks are a viable surrogate modeling approach for rocket trajectory reconstruction. The ability to generate full state trajectories without explicit numerical integration suggests potential applications in rapid simulation, design space exploration, and embedded onboard prediction systems.

However, the observed constraint violations and derivative inaccuracies indicate that the current model is better suited for analysis and approximation rather than direct deployment in control loops. For industrial or mission-critical applications, additional safeguards and validation steps would be required.

The approach remains attractive for scenarios where computational efficiency and differentiability are prioritized, such as optimization, inverse problems, or sensitivity analysis.

4.8 Limitations of the Present Study

Several limitations should be acknowledged. First, the model is evaluated on simulated data generated from the same underlying physics used in training. Generalization to real flight data with unmodeled disturbances remains untested.

Second, constraint enforcement relies entirely on soft penalties, which allows physically inconsistent solutions when trade-offs favor loss minimization. This is particularly evident in mass monotonicity and horizontal drift.

Third, the fixed time horizon and uniform sampling may limit the model's ability to handle variable-duration flights or abrupt dynamic transitions.

These limitations do not invalidate the results but highlight areas where further development is necessary.

4.9 Recommendations for Future Improvements

Future work should explore hybrid constraint enforcement strategies that combine physics-informed losses with hard constraints or projection-based corrections. Enforcing mass monotonicity and symmetry directly at the architectural level may significantly improve physical consistency.

Adaptive loss weighting and phase-aware training could improve performance during critical dynamic transitions, such as engine cutoff. Additionally, extending the framework to handle variable time horizons and stochastic disturbances would enhance practical applicability.

Incorporating real-world flight data or higher-fidelity aerodynamic models would further test the robustness and generalization capability of the proposed approach.

CHAPTER V

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusions

This thesis investigated whether a structured physics-informed neural network can function as a surrogate model for rocket ascent dynamics and be embedded within a trajectory optimization context. The study was motivated by the computational cost and limited flexibility of classical optimal control solvers when repeated trajectory evaluations or gradient information are required.

The results support the conclusion that physics-informed neural networks constitute a viable surrogate modeling approach for rocket ascent dynamics when the governing equations are well understood and training data are available. By embedding the equations of motion directly into the learning objective, the proposed model is able to reconstruct full-state trajectories that are globally consistent with the reference dynamics without relying on explicit numerical integration during inference. This confirms the central hypothesis that physics guidance can significantly improve the reliability of neural surrogates compared to purely data-driven models.

In line with prior work on physics-informed learning, the study shows that state variables governed primarily by integral effects are captured more robustly than derivative-dominated quantities. This observation is consistent with findings reported by Raissi et al. and Karniadakis et al., where PINNs were shown to approximate smooth solution manifolds effectively while remaining sensitive to higher-order dynamics. The present work extends these insights to a six-degree-of-freedom rocket ascent problem with coupled translational, rotational, and mass dynamics.

At the same time, the thesis demonstrates that soft enforcement of physical con-

straints is insufficient to guarantee strict compliance with non-negotiable physical laws such as mass monotonicity and symmetry preservation. While the model learns the overall structure of propellant consumption and orientation dynamics, local violations remain possible when constraint satisfaction competes with data fidelity. This aligns with recent theoretical analyses of PINN optimization behavior, which emphasize the trade-off between loss terms and the absence of hard guarantees in penalty-based formulations.

Overall, the findings indicate that physics-informed neural networks are well suited as analysis and optimization surrogates, rather than direct replacements for high-fidelity simulators in safety-critical applications. Their value lies in enabling differentiable, computationally efficient approximations that preserve the dominant physical behavior of complex dynamical systems.

5.2 Implications for Trajectory Optimization and Scientific Machine Learning

From a trajectory optimization perspective, the proposed surrogate framework offers a promising pathway toward reducing reliance on repeated numerical integration in gradient-based optimization loops. The differentiable nature of the model enables direct sensitivity analysis with respect to initial conditions or control parameters, which is difficult to achieve efficiently with traditional solvers.

Within the broader context of scientific machine learning, this work reinforces the view that architectural structure and loss design are as important as network capacity. The observed behavior of the model highlights the need for careful alignment between physical constraints, training objectives, and the intended downstream use of the surrogate. The study thus contributes practical evidence to ongoing discussions in the PINN literature regarding robustness, constraint enforcement, and long-horizon prediction.

5.3 Recommendations for Future Work

Based on the conclusions drawn from this study, several recommendations are proposed for future research and practical development.

First, future models should incorporate hard or semi-hard constraint enforcement mechanisms for critical physical laws, particularly mass monotonicity and symmetry conditions. Projection methods, constrained output parameterizations, or hybrid solver-network approaches may provide stronger physical guarantees than penalty-based losses alone.

Second, phase-aware or adaptive training strategies are recommended to address regions of rapid dynamic change, such as engine cutoff or thrust transitions. Dynamically adjusting loss weights or introducing curriculum learning could improve derivative accuracy without sacrificing global trajectory consistency.

Third, extending the framework to variable time horizons and non-nominal flight conditions, including wind disturbances or parameter uncertainty, would significantly enhance its applicability to real-world scenarios. Such extensions would also provide a more rigorous test of generalization beyond the training regime.

Finally, integration with real flight data or higher-fidelity simulation environments is recommended to evaluate robustness and practical relevance. Combining physics-informed learning with data assimilation techniques may offer a path toward hybrid models capable of bridging the gap between simulation and operational deployment.

5.4 Final Remarks

This thesis demonstrates that physics-informed neural networks can serve as meaningful surrogate models for rocket ascent dynamics when designed and evaluated carefully. While the approach does not eliminate the need for classical simulation tools, it provides a complementary methodology that supports fast evaluation, dif-

ferentiability, and physical consistency.

The insights gained from this work underscore both the potential and the limitations of PINN-based surrogates, offering a foundation for future research aimed at integrating machine learning more deeply into aerospace trajectory analysis and optimization workflows.

BIBLIOGRAPHY

[1] John D. Anderson. *Computational Fluid Dynamics: The Basics with Applications*. New York: McGraw-Hill, 1995.

[2] John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. 2nd. Philadelphia, PA: SIAM, 2010.

[3] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation and Control*. New York: Hemisphere Publishing Corporation, 1975.

[4] George Em Karniadakis et al. *Physics-Informed Machine Learning*. Cambridge: Cambridge University Press, 2022.

[5] George Em Karniadakis et al. *Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations*. Cambridge: Cambridge University Press, 2021.

[6] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2015).

[7] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic gradient descent with warm restarts". In: *International Conference on Learning Representations*. 2017.

[8] Siddhartha Mishra and Roberto Molinaro. "Estimates on the generalization error of physics-informed neural networks for approximating PDEs". In: *IMA Journal of Numerical Analysis* 42.2 (2022), pp. 981–1022.

[9] Adam Paszke et al. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[10] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[11] I. Michael Ross. *A Beginner’s Guide to Optimal Control*. Berkeley, CA: University of California Press, 2004.

[12] George P. Sutton and Oscar Biblarz. *Rocket Propulsion Elements*. 9th. Hoboken, NJ: Wiley, 2017.

[13] Matthew Tancik et al. “Fourier features let networks learn high frequency functions in low dimensional domains”. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 7537–7547.

[14] Sifan Wang, Hanwen Wang, and Paris Perdikaris. “When and why PINNs fail to train: A neural tangent kernel perspective”. In: *Journal of Computational Physics* 449 (2022), p. 110768.