Brief Report

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.

1 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 [**?**, **?**, **?**].

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. [**?**], 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 [**?**, **?**].

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.

1.2 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 [**?**, **?**]. Analytical studies have further examined PINN training dynamics and failure modes, emphasizing the importance of architectural design and loss construction for practical success [**?**, **?**].

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.

1.3 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.

1.4 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.

1.5 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.

1.6 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.

1.7 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.

1.8 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 reviews the theoretical background and related work in physics-informed neural networks and optimal control. Chapter 3 presents the rocket dynamics model and problem formulation. Chapter 4 describes the PINN architecture, training methodology, and surrogate evaluation. Chapter 5 reports the numerical results. Chapter 6 discusses the findings and limitations of the study. Chapter 7 concludes the thesis and outlines directions for future research.

2 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 **??**.

2.2 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:

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

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

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

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

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

Drag and lift forces in the body frame took the form

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

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

The thrust vector in the body frame was

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

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

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

2.3 Numerical Simulation and Trajectory Generation

2.3.1 Optimal Control Problem Formulation

The ascent trajectory design problem was posed as a continuous-time optimal control problem (OCP) over a finite horizon $t \in [0, t_f]$, with the 6-DOF dynamics as constraints. The objective function was formulated to minimize 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.

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 to improve 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 was implemented in 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 differentiation.

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 to improve numerical conditioning and enable scale-invariant learning. The reference scales were chosen to yield quantities of order unity, as summarized in Table 1.

Table 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.

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 **??**). A shared stem first embedded the normalised time and context: time was expanded by Fourier features 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 $[\mathbf{r}_{\text{i}}, \mathbf{v}_{\text{i}}]$. A rotation branch with hidden dimensions $[256, 256]$ mapped z to quaternion and angular velocity components $[\mathbf{q}, \boldsymbol{\omega}_{\text{b}}]$ 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

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.

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.

2.6 Training Procedure

2.6.1 Loss Function Components

The training objective for Direction AN followed a physics-informed pattern. Given a reference trajectory $x^\star(t_k)$ on a uniform grid $\{t_k\}_{k=0}^{N-1}$, the total loss consisted of four groups: (1) data fidelity losses, (2) physics consistency losses, (3) structural constraints, and (4) smoothing and trajectory regularization.

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 regularization terms were included:

- $L_{\text{mass,res}}$: 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

$$\begin{aligned}
\mathcal{L} = {} & \lambda_{\text{data}} L_{\text{data}} + \lambda_{\text{phys}} L_{\text{phys}} + \lambda_{\text{bc}} L_{\text{bc}} \\
& + \lambda_{\text{mass,res}} L_{\text{mass,res}} + \lambda_{\text{vz,res}} L_{\text{vz,res}} + \lambda_{\text{vxy,res}} L_{\text{vxy,res}} \\
& + \lambda_{\text{smooth,z}} L_{\text{smooth,z}} + \lambda_{\text{smooth,vz}} L_{\text{smooth,vz}} + \lambda_{\text{pos,vel}} L_{\text{pos,vel}} \\
& + \lambda_{\text{smooth,pos}} L_{\text{smooth,pos}} + \lambda_{\text{zero,vxy}} L_{\text{zero,vxy}} + \lambda_{\text{zero,axy}} L_{\text{zero,axy}} \\
& + \lambda_{\text{hacc}} L_{\text{hacc}} + \lambda_{\text{xy,zero}} L_{\text{xy,zero}},
\end{aligned} \tag{1}$$

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 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 was used with $T_{\max} = 160$ and minimum learning rate $\eta_{\min} = 1.0 \times 10^{-6}$.

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, allowing the model to 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.

2.7 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

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

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.

2.8 Implementation Details

The neural network models were implemented in PyTorch. The optimal control solver was implemented using CasADi for symbolic computation and IPOPT for nonlinear optimization. The truth integrator was implemented in C++. Data processing and training scripts were written in Python, using NumPy for numerical operations and HDF5 for data storage. Training was conducted on computational hardware with automatic device selection (CPU or GPU) depending on availability.

2.9 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.