Brief Report

1 Introduction

Modern launch vehicles are critically constrained by trajectory design. Small improvements in ascent guidance can translate into significant gains in payload capacity, mission robustness, and operational safety. At the same time, the governing dynamics of a rocket in atmospheric flight are highly nonlinear and strongly coupled: translational and rotational motion interact through aerodynamic forces, thrust vectoring, and mass depletion, all under tight structural and environmental constraints.

This project develops a high-fidelity six-degree-of-freedom (6-DOF) dynamics model for a launch vehicle, together with a direct optimal control formulation of the ascent trajectory problem and data-driven surrogate models based on physics-informed neural networks (PINNs). The resulting framework can generate physically consistent optimal trajectories, enforce realistic path constraints such as limits on dynamic pressure and load factor, and provide differentiable approximations of the dynamics for downstream optimisation and analysis.

The present document introduces the underlying physical model, the numerical methods used to solve the optimal control problem, and the current state and planned evolution of the overall methodology.

1.1 Document Structure and Scope

This document is organized as follows. Section 2 defines the physical model and assumptions underlying the 6-DOF dynamics, including state and control representations, reference frames, and the forces and moments acting on the vehicle. Section 3 explains how optimal trajectories are generated using direct collocation methods and the CasADi/IPOPT solver framework. Section 4 describes how these optimal trajectories are used to train physics-informed neural network (PINN) surrogates, with the Direction AN architecture serving as the primary surrogate model studied in this work. The relationship between the OCP solver, truth integrator, and PINN surrogate is illustrated schematically in Figure 1 and discussed in detail in Section 4.

2 Physical Model

2.1 Notation

Table 1 summarizes the key notation used throughout this document, including state and control variables, physical parameters, and their typical ranges or units.

2.2 State, Control, and Reference Frames

The rocket is modelled as a rigid body with full 6-DOF dynamics. The state vector $x \in \mathbb{R}^{14}$ is

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

where

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

Table 1: Notation and typical parameter ranges

Symbol	Description	Units	Typical Range
$\mathbf{r}_{\mathrm{i}}$	Position (inertial)	m	–
$\mathbf{v}_{\mathrm{i}}$	Velocity (inertial)	m/s	0–500
$\mathbf{q}$	Quaternion (attitude)	–	unit norm
$\boldsymbol{\omega}_{\mathrm{b}}$	Angular velocity (body)	rad/s	–
m	Mass	kg	35–65
T	Thrust magnitude	N	0–5000
$\theta_{\mathrm{g}}, \phi_{\mathrm{g}}$	Gimbal angles	rad	± 0.1745
δ	Control surface deflection	rad	± 0.1745
m_0	Initial mass	kg	45–65
I_{sp}	Specific impulse	s	220–280
C_{D}	Drag coefficient	–	0.25–0.45
$C_{\mathrm{L}\alpha}$	Lift-curve slope	rad^{-1}	2.5–4.5
$C_{\mathrm{m}\alpha}$	Pitch-moment coefficient	rad^{-1}	-1.2–(-0.4)
$T_{\max}$	Maximum thrust	N	3000–5000
$q_{\mathrm{dyn,max}}$	Max dynamic pressure	Pa	40,000
$n_{\max}$	Max load factor	g	5.0

- $m \in \mathbb{R}$ is the mass of the vehicle.

The control vector $u \in \mathbb{R}^4$ is

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

where

- T is the commanded thrust magnitude,
- θ_{g} and ϕ_{g} are the thrust gimbal pitch and yaw angles, and
- δ is a representative control surface deflection.

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

Reference frames summary. The model uses two primary reference frames:

- **Inertial frame**: Earth-centred, non-rotating, with z-axis pointing upward (opposite to gravity).
- **Body frame**: Vehicle-fixed, with x-axis aligned with the nominal thrust direction.
- **Transformations**: Quaternions are used throughout to represent rotations between frames, with $R_{\mathrm{b}\to\mathrm{i}}(\mathbf{q})$ transforming vectors from body to inertial coordinates.

2.3 Physical Scales and Typical Values

The model is parameterised for small sounding rockets with initial masses ranging from 45 to 65 kg, maximum thrusts of 3000–5000 N, and specific impulses of 220–280 s. Typical trajectories reach apogees of 30–42 km over 30-second flight durations, with maximum velocities of approximately 300–500 m/s. The aerodynamic coefficients vary within realistic ranges: drag coefficient $C_{\mathrm{D}} \in [0.25, 0.45]$, lift-curve slope $C_{\mathrm{L}\alpha} \in [2.5, 4.5]$ rad^{-1}, and pitch-moment coefficient $C_{\mathrm{m}\alpha} \in [-1.2, -0.4]$ rad^{-1}. Path constraints enforce maximum dynamic pressure of 40,000 Pa and maximum load factor of 5 g, consistent with structural limits for small launch vehicles. Typical propellant consumption ranges from 10 to 30 kg over the flight duration (corresponding to initial masses of 45–65 kg and a dry mass of 35 kg), with gimbal angles remaining small (less than 10°) for near-vertical ascent trajectories.

Control representation across the pipeline. Different parts of the implementation use different control representations for numerical and storage efficiency. The CasADi-based optimal control solver and PyTorch PINN dynamics use the gimbal-angle parameterisation $u = [T, \theta_{\mathrm{g}}, \phi_{\mathrm{g}}, \delta]^\top$ described above. The C++ truth integrator uses a unit-vector representation $\tilde{u} = [T, u_{\mathrm{T},x}, u_{\mathrm{T},y}, u_{\mathrm{T},z}, \delta]^\top$, where the thrust direction components are related to the gimbal angles by

$$u_{\mathrm{T},x} = \cos\theta_{\mathrm{g}} \cos\phi_{\mathrm{g}}, \qquad u_{\mathrm{T},y} = \sin\phi_{\mathrm{g}}, \qquad u_{\mathrm{T},z} = \sin\theta_{\mathrm{g}} \cos\phi_{\mathrm{g}}.$$

Processed datasets store control as $[T, u_{\mathrm{T},x}, u_{\mathrm{T},y}, u_{\mathrm{T},z}]$ (4-D format, omitting δ which is typically zero). The control surface deflection δ is optimised in the CasADi formulation and included in the C++ dynamics when non-zero, but is not used in PINN training, where a zero-control assumption is employed (see Section 4).

2.4 Forces, Moments, and Mass Flow

The translational motion is 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 $\mathbf{F}_{\mathrm{i}}$ is the total non-gravitational force in the inertial frame and $\mathbf{g}_{\mathrm{i}}$ is the gravitational acceleration. In the high-fidelity C++ "truth" model, gravity is modelled as an inverse-square law,

$$\mathbf{g}_{\mathrm{i}}(\mathbf{r}_{\mathrm{i}}) = -g_0 \left(\frac{R_{\mathrm{E}}}{\|\mathbf{r}_{\mathrm{i}}\|}\right)^2 \frac{\mathbf{r}_{\mathrm{i}}}{\|\mathbf{r}_{\mathrm{i}}\|},$$

with Earth radius R_{E} and nominal surface gravity g_0. In the CasADi- and PyTorch-based models used for optimal control and PINN training, a constant gravitational field $\mathbf{g}_{\mathrm{i}} = [0, 0, -g_0]^\top$ is employed for simplicity and numerical robustness.

Aerodynamic and thrust forces are naturally expressed in the body frame. The relative wind velocity is

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

where $\mathbf{v}_{\mathrm{wind,i}}$ is the wind velocity in the inertial frame. The C++ dynamics support a wind callback; in the current optimal control and PINN implementations, wind is neglected and $\mathbf{v}_{\mathrm{wind,i}} = \mathbf{0}$.

The atmosphere is modelled by an exponential density profile,

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

where h is altitude above sea level, ρ_0 is the sea-level density, and h_{scale} is a scale height. The dynamic pressure is

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

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

The thrust vector in the body frame is

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

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

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

Rotational dynamics are 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 $\mathbf{I}_\mathrm{b}$ is the inertia tensor in the body frame and $\mathbf{M}_\mathrm{b}$ is the sum of aerodynamic and thrust moments, including gimbal and control-surface effects via pitch-moment and control derivatives.

The mass dynamics follow 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 is enforced in the implementation to prevent unphysical mass depletion.

Model implementation hierarchy. The same physical model is implemented at three fidelity levels to serve different purposes in the pipeline: (i) a high-fidelity C++ truth integrator used for validation and data generation, (ii) a symbolic CasADi version for optimal control problem formulation and solution, and (iii) a differentiable PyTorch implementation for PINN training and inference. Differences between implementations are limited to gravity and wind modelling choices for numerical robustness: the C++ integrator supports inverse-square gravity, while the CasADi and PyTorch versions use constant gravity for computational efficiency. Wind is configurable in the C++ integrator but set to zero in the optimization and PINN configurations. This hierarchy enables the framework to generate high-fidelity reference trajectories while maintaining computational tractability for large-scale optimization and neural network training.

2.5 Assumptions and Limitations

The current model assumes a spherical, non-rotating Earth, neglects higher gravitational harmonics and planetary rotation, and uses a single-vehicle rigid body without staging or flexible modes. The atmosphere is represented by a single-parameter exponential model; detailed thermodynamic and compositional effects are neglected. Wind is configurable in the C++ truth integrator but is set to zero in the optimisation and PINN configurations.

Motivation for simplifications. Each simplification is chosen to balance fidelity with computational tractability while maintaining accuracy for the target application. The spherical Earth assumption is acceptable for altitudes below 100 km (typical trajectories reach 30–42 km), where gravitational variations are less than 1%. Constant gravity in the optimization formulation reduces computational cost while maintaining accuracy for short-duration flights; the error introduced is negligible compared to other uncertainties. The exponential atmosphere model captures over 90% of the density variation relevant to ascent trajectories; more complex models (e.g., US Standard Atmosphere) add minimal value for the ascent phase while significantly increasing computational complexity. The single-stage rigid-body assumption focuses the model on the ascent phase, where staging events are not present; flexible modes are negligible for small rockets with high structural stiffness (typical natural frequencies exceed 100 Hz, well above the trajectory dynamics). Wind effects are neglected in optimization to reduce parameter space dimensionality, but the framework supports wind models for future extensions. These simplifications yield a model that is rich enough to capture the key couplings between translation, rotation, aerodynamics, and thrust, while remaining tractable for large-scale optimal control and data generation.

3 Numerical Solution and Optimal Control Formulation

3.1 Direct Optimal Control and Transcription

The ascent trajectory design problem is posed as a continuous-time optimal control problem (OCP) over a finite horizon $t \in [0, t_f]$, with the 6-DOF dynamics from Section 2 as constraints. Typical objectives include minimising fuel consumption or maximising delivered payload, subject to path and terminal constraints on altitude, velocity, attitude, and structural loads. The optimal control solver is not the final product of this project, but a data-generation mechanism for training and validating physics-informed surrogate models. The relationship between the OCP solver, truth integrator, and PINN surrogate is illustrated schematically in Figure 1 and discussed in detail in Section 4.

In this project the OCP is discretised by a direct collocation method. The continuous state and control trajectories are 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 $u_k \approx u(t_k)$ become decision variables in a finite-dimensional nonlinear programme (NLP). The dynamics are enforced via Hermite–Simpson collocation, which provides 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 are enforced as equality constraints in the NLP.

3.2 CasADi-Based Dynamics and IPOPT Solver

The 6-DOF dynamics, including aerodynamics, thrust, and mass depletion, are implemented symbolically in CasADi. This provides exact Jacobians and Hessians to the NLP solver, which is crucial for robustness and performance in the presence of stiff dynamics and tight path constraints. Special care is 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 is solved using IPOPT, a large-scale interior-point nonlinear optimiser. State and control variables are scaled using reference length, velocity, time, mass, and force scales to improve conditioning, and linear solver backends (such as MUMPS or HSL variants) are selected automatically depending on availability. The framework supports 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 are enforced directly in the NLP using auxiliary CasADi functions for these quantities. Operational limits on gimbal angles, control-surface deflections, and angle of attack are also encoded in the state and control bounds. The solver returns optimal knot sequences $\{x_k^\star\}_{k=0}^{N}$, $\{u_k^\star\}_{k=0}^{N-1}$, an optimal final time $t_f^\star$, and detailed convergence statistics.

3.3 Truth Integration and PINN Dynamics

In addition to the collocation-based OCP solver, the project includes a high-fidelity C++ integrator for the same 6-DOF dynamics, which is used as a "truth" model for validation and data generation. This integrator supports inverse-square gravity, exponential atmosphere, aerodynamic forces and moments, wind callbacks, and detailed diagnostic outputs such as dynamic pressure, load factor, and constraint violations. A Python wrapper interface is planned to expose this integrator as a function on uniform time grids; at present, the wrapper is defined but not yet fully implemented.

For data-driven approximation, a differentiable version of the dynamics is implemented in PyTorch. This module mirrors the CasADi model, including the state and control definitions, aerodynamic coefficients, and mass dynamics, but uses tensor operations to enable automatic differentiation. It serves as the physics backbone for physics-informed neural networks and hybrid PINN architectures, which learn to reproduce or augment the 6-DOF dynamics from simulated optimal trajectories.

4 Methodology, Current Status, and Planned Extensions

4.1 Overall Methodology

The project methodology is organised around three tightly coupled components:

1. a high-fidelity 6-DOF dynamics truth integrator,
2. a CasADi- and IPOPT-based optimal control solver using direct collocation, and
3. data-driven surrogate models based on PINNs and related architectures.

The workflow proceeds as follows. First, the physical model is specified and implemented consistently across C++, CasADi, and PyTorch, including a common set of physical parameters and operational limits. Second, the direct collocation solver is configured and tuned to produce dynamically feasible, constraint-satisfying ascent trajectories across a range of mission scenarios. Third, these optimal trajectories are used to construct datasets (stored in HDF5 format) for training and validating PINN-based surrogate models of the dynamics and, where appropriate, of the optimal control policy.

4.2 Current State of Data and Models

On the dynamics and optimisation side, the 6-DOF model is fully implemented in C++ and in symbolic CasADi form, including aerodynamic forces and moments, thrust vector control, and mass depletion. The direct collocation transcription, Hermite–Simpson defects, state and control bounds, and path constraints on dynamic pressure, load factor, and mass are implemented and interfaced with IPOPT. Solver configuration includes robust scaling, automatic linear-solver selection, and diagnostics for constraint violation and iteration statistics.

4.3 Dataset Structure and Storage

The data pipeline produces a hierarchy of datasets. At the lowest level, raw cases are 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 are then aggregated into processed split files (train/validation/test) in nondimensional form. The processed datasets contain approximately 120 training cases, 20 validation cases, and 20 test cases. Each trajectory is discretised on a uniform time grid with $N = 1501$ points, corresponding to a 30-second flight duration sampled at 50 Hz. Each processed dataset exposes a time grid, a context vector of dimension 7 encoding key physical and environmental parameters (initial mass m_0, specific impulse I_{sp}, drag coefficient C_{D}, lift-curve slope $C_{\mathrm{L}\alpha}$, pitch-moment coefficient $C_{\mathrm{m}\alpha}$, maximum thrust $T_{\max}$, and wind magnitude), and normalised state targets; an extended "v2" format additionally provides time series of thrust magnitude and dynamic pressure as explicit input features.

4.4 Nondimensionalization and Scaling

All state and control variables are nondimensionalized using physics-aware reference scales to improve numerical conditioning and enable scale-invariant learning. The reference scales are chosen to yield quantities of order unity, as summarized in Table 2.

Table 2: Reference scales for nondimensionalization

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

This nondimensionalization ensures that all state components are typically in the range $[0.1, 10]$, which improves gradient flow during neural network training and reduces numerical errors in the optimization solver. The scaling also makes the model scale-invariant: trajectories for different-sized rockets can be learned using the same architecture, with physical scale encoded in the context vector.

On the modelling side, several families of PINN and hybrid architectures have been implemented in PyTorch. Common building blocks include Fourier feature embeddings of time, shallow and deep encoders for the context vector, and MLP or Transformer-based temporal encoders. Sequence PINNs treat the entire time grid as a sequence and apply Transformer encoders to predict the 14-D state at all nodes. Hybrid latent-ODE models combine an encoder that infers a latent initial condition z_0 from early-time behaviour and context with a learned latent dynamics model that evolves $z(t)$ over time, followed by decoders or dedicated branches that reconstruct translation, rotation (with explicit quaternion normalisation or minimal parametrisations), and mass trajectories. Several designs enforce structural properties such as monotonically decreasing mass and unit-norm quaternions by construction, and all models can use the differentiable 6-DOF dynamics module as a physics regulariser in the training loss to promote physically consistent predictions.

4.5 Direction AN Model and Training Objective

Among the available architectures, the "Direction AN" model is a canonical example of how the project combines learned representations with physics-based regularisation. This section describes its architecture, training objective, and design rationale.

4.5.1 Architecture of Direction AN

The Direction AN structure is organised into three stages (see Figure 1). A shared stem first embeds the normalised time and context: time is expanded by Fourier features, while the context vector is passed through a small encoder and then concatenated with the time embedding. This combined input is processed by a residual multilayer perceptron with skip connections and optional layer normalisation, yielding a latent feature sequence $z(t, c)$ of fixed dimension along the trajectory.

On top of this shared latent representation, three "mission branches" specialise to different parts of the 14-D state. A translation branch maps z to position and velocity components $[\mathbf{r}_\mathrm{i}, \mathbf{v}_\mathrm{i}]$, a rotation branch maps z to quaternion and angular velocity components $[\mathbf{q}, \boldsymbol{\omega}_\mathrm{b}]$ with explicit quaternion renormalisation, and a mass branch produces the mass trajectory $m(t)$. The concatenation of these branch outputs forms the predicted state $x_\theta(t)$. A variant of the architecture (AN1) replaces the plain stem input with a richer block that also ingests time series of thrust magnitude and dynamic pressure, fusing $(t, c, T_\mathrm{mag}(t), q_\mathrm{dyn}(t))$ into the shared latent representation.

Architecture design rationale. The Direction AN architecture is designed to leverage the physical structure of the 6-DOF dynamics while enabling efficient learning. The shared stem captures common temporal and physical dependencies that affect all state components (e.g., altitude-dependent atmospheric effects, time-varying mass). Separate mission branches decouple the learning of translation, rotation, and mass dynamics, which have different physical characteristics: translation follows Newton's laws with aerodynamic coupling, rotation involves quaternion kinematics and moment dynamics, and mass follows a simple depletion law. This decoupling allows each branch to specialize while sharing common features through the stem. The explicit quaternion renormalisation in the rotation branch ensures valid rotations (unit quaternions) by construction, avoiding the need for post-processing or soft constraints. The physics residual layer directly enforces the 6-DOF dynamics equations, ensuring that predictions satisfy the underlying physics even when data is sparse or noisy. This hybrid approach combines the expressiveness of neural networks with the rigor of physics-based constraints.

4.5.2 Training Objective and Loss Design

The training objective for Direction AN follows 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 consists of four groups:

1. **Data fidelity losses**: Component-weighted mean-squared error between predicted and true states, with separate group weights for translation, rotation, and mass.
2. **Physics consistency losses**: Penalties for violations of the continuous dynamics, comparing finite-difference estimates of time derivatives to the right-hand side of the 6-DOF model.
3. **Structural constraints**: Quaternion norm penalties, mass monotonicity constraints, and boundary condition enforcement.
4. **Smoothing and trajectory regularisation**: Terms that encourage smooth trajectories, position–velocity consistency, and suppression of unphysical horizontal motion.

The physics loss L_phys penalises 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 $f(x_\theta(t_k), u_k; p)$ of the 6-DOF model implemented in PyTorch:

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

In the PINN setting, explicit control inputs are omitted and set to zero in the physics residual, reflecting the design choice that the network learns state evolution implicitly from trajectories rather than from control commands. The control u_k in the physics loss is therefore set to zero ($u_k = [0, 0, 0, 0]^\top$). This zero-control assumption is consistent with the processed dataset format, which does not include control trajectories, and enables the model to learn the mapping from initial conditions and physical parameters directly to state evolution. The control is implicitly encoded in the state trajectory itself, as the trajectories are generated by an optimal control solver and therefore already reflect optimal control effects.

A boundary term L_bc enforces agreement between predicted and true initial conditions, while regularisers such as the quaternion norm penalty L_quat and a mass-flow consistency term L_mass encourage unit quaternions and non-increasing mass along the trajectory.

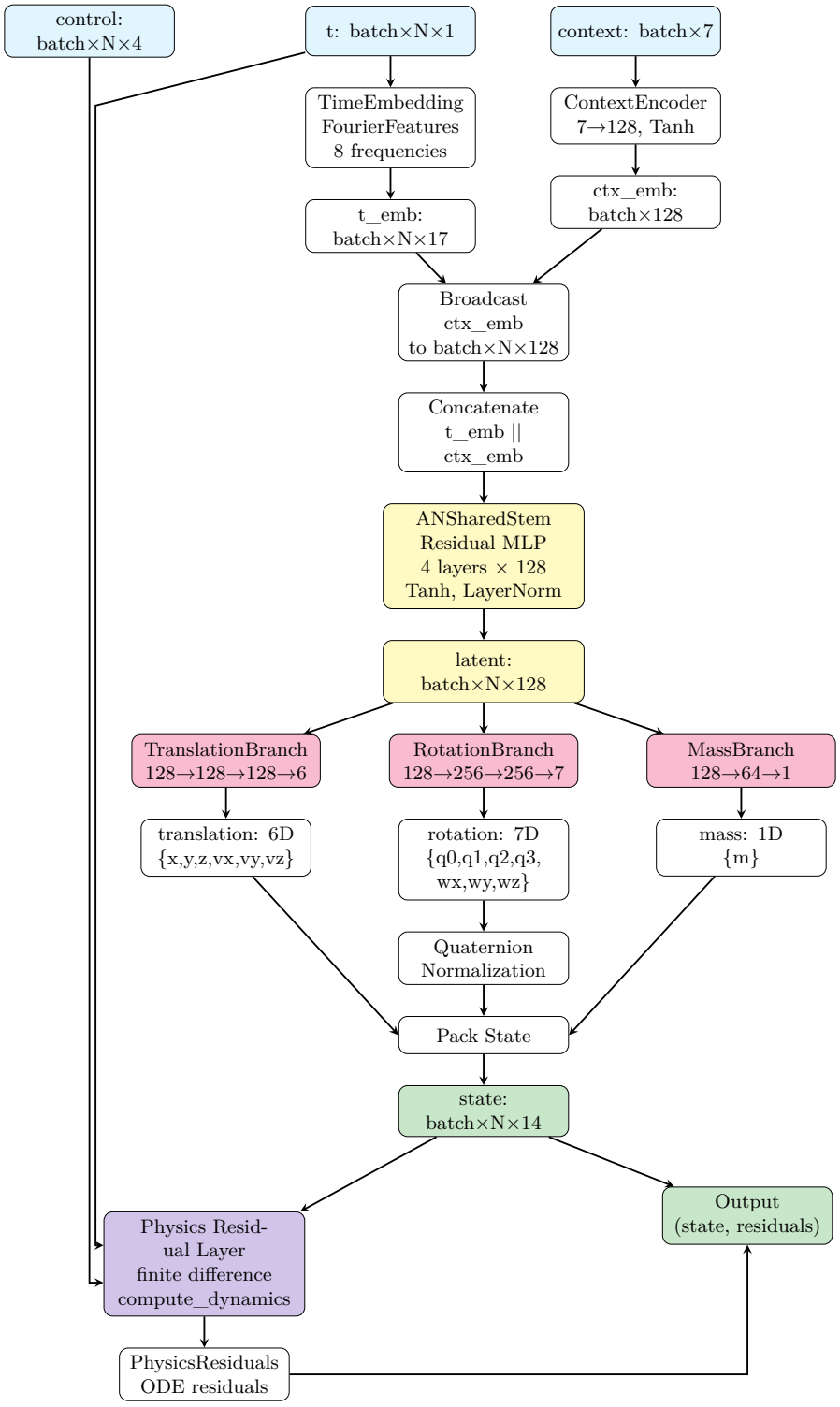

Figure 1: Architecture of the Direction AN model. The shared stem processes time and context inputs to produce a latent representation $z(t, c)$. Three specialized branches (translation, rotation, mass) map this latent space to the corresponding state components. The physics residual layer computes dynamics residuals to enforce physical consistency.

Training employs a phased schedule where loss weights are adjusted over time. In the initial phase (approximately the first 55% of training epochs), the model focuses on fitting the data and coarse physics constraints, with lower weights on higher-order regularisation terms. In the second phase, these regularisation weights are gradually increased using a cosine ramp schedule, allowing the model to refine its predictions while maintaining physical consistency. This phased approach helps prevent the model from overfitting to the regularisation terms early in training while ensuring that physical constraints are properly enforced in the final model.

4.5.3 Rationale for Loss Components

The additional soft losses serve specific physical and numerical purposes:

- $L_{\text{mass,res}}$: Ensures mass decreases monotonically, preventing unphysical mass gain that could arise from data fitting alone.
- $L_{\text{vz,res}}$: Encourages realistic vertical velocity evolution, ensuring the model captures the acceleration–deceleration profile of ascent trajectories.
- $L_{\text{vxy,res}}$: Suppresses spurious horizontal velocity components, as sounding rockets should follow near-vertical trajectories.
- $L_{\text{smooth,z}}$, $L_{\text{smooth,vz}}$: Temporal smoothing terms that prevent oscillatory predictions in altitude and vertical velocity, which can arise from noisy data or insufficient physics constraints.
- $L_{\text{smooth,pos}}$: Smooths position predictions to ensure physically plausible trajectories without sudden jumps.
- $L_{\text{pos,vel}}$: Enforces kinematic consistency between position and velocity (i.e., $\dot{\mathbf{r}} = \mathbf{v}$).
- $L_{\text{zero,vxy}}$, $L_{\text{zero,axy}}$: Suppress horizontal velocities and accelerations during near-vertical flight, as lateral motion should be minimal for sounding rockets.
- L_{hacc}, $L_{\text{xy,zero}}$: Additional constraints on horizontal acceleration and position to ensure vertical ascent behavior.

These regularization terms work together to ensure that the learned model produces physically plausible trajectories that respect the expected behavior of sounding rocket ascent, even when training data is limited or contains outliers. Together, these elements yield a training objective that balances data fidelity with dynamical consistency and structural robustness.

RMSE calculation. Model performance is evaluated using root mean square error (RMSE) computed on the test set. For a trained Direction AN model, predictions $\hat{x}_\theta(t_k)$ are generated for all test trajectories on a uniform time grid $\{t_k\}_{k=0}^{N-1}$ spanning the trajectory duration (typically 30 seconds). The total RMSE is 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 is the number of test trajectories (typically 20), $N = 1501$ is the number of time points in the discretized trajectory grid (corresponding to a sampling rate of approximately 50 Hz), $D = 14$ is the state dimension, and $x^\star(b)(t_k)$ denotes the ground truth state for trajectory b at time t_k. The summation over k from 0 to $N-1$ averages the squared errors across all time points in each trajectory, while the summation over b averages across all test trajectories, and the summation over d averages across all state components. Per-component RMSE is computed by averaging over trajectories and time points for each state dimension d, while aggregated metrics are computed for translation (components 1–6: position and velocity), rotation (components 7–13: quaternion and angular velocity), and mass (component 14). All RMSE values are reported in nondimensional units, reflecting the scaled state representation used during training. The current Direction AN model achieves a total RMSE of approximately 0.197 on the test set, with translation RMSE of 0.264, rotation RMSE of 0.133, and mass RMSE of 0.015.

RMSE interpretation. The translation RMSE (0.264) is higher than rotation RMSE (0.133) because vertical position and velocity are the most dynamically active components, experiencing large changes over the 30-second ascent. The rotation RMSE is relatively low because attitude changes are constrained by the near-vertical trajectory and the explicit quaternion normalization in the architecture. The mass RMSE (0.015) is very low, reflecting the simple depletion dynamics and the structural monotonicity constraint. The total RMSE of 0.197 represents a moderate relative error when compared to the typical nondimensional state ranges (0.1–10), which is acceptable for surrogate modeling applications where the primary goal is capturing trajectory structure and physics rather than achieving perfect point-wise accuracy.

4.6 Planned Extensions

Several extensions are planned to increase fidelity and robustness:

- incorporating more detailed atmospheric and wind models in both the truth integrator and the OCP formulation;
- modelling staging events, thrust profiles, and mass properties that vary with propellant depletion and configuration changes;
- enriching the objective functions to include measures of robustness, controllability, or dispersion sensitivity;
- extending the PINN and latent-dynamics models to capture a wider range of operating conditions and to support uncertainty quantification.

Further work will also focus on integrating the C++ truth integrator more tightly with the Python/ML stack, enabling closed-loop validation of learned controllers and surrogates against the highest-fidelity model.