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.

2   Physical Model

2.1   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

- $m \in \mathbb{R}$ 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 $R_{b\to i}(\mathbf{q})$ is obtained from the quaternion $\mathbf{q}$. Its transpose $R_{i\to b} = R_{b\to i}^{\top}$ transforms vectors from the inertial frame to the body frame.

2.2 Forces, Moments, and Mass Flow

The translational motion is governed by Newton's second law,

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

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

$$\mathbf{g}_i(\mathbf{r}_i) = -g_0\left(\frac{R_E}{\|\mathbf{r}_i\|}\right)^2 \frac{\mathbf{r}_i}{\|\mathbf{r}_i\|},$$

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 $\mathbf{g}_i = [0, 0, -g_0]^{\top}$ is employed for simplicity and numerical robustness.

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

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

where $\mathbf{v}_{wind,i}$ 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 $\mathbf{v}_{wind,i} = \mathbf{0}$.

The atmosphere is modelled by an exponential density profile,

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

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

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

Drag and lift forces in the body frame take the form

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

where S_{ref} is a reference area, C_D is the drag coefficient, $C_{L\alpha}$ is a lift-curve slope, α is the angle of attack computed from $\mathbf{v}_{rel,b}$, and $\hat{\mathbf{e}}_L$ 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.

2.3 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.

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.

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 model (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.

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. Each processed dataset

exposes a time grid, a context vector encoding key physical and environmental parameters (masses, thrust limits, aerodynamic coefficients, inertia properties, atmospheric scales, wind settings, and constraint limits), and normalised state targets; an extended "v2" format additionally provides time series of thrust magnitude and dynamic pressure as explicit input features.

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.

4.3 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. Its structure is organised into three stages. 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.

The training objective for Direction AN follows a physics-informed pattern. Given a reference trajectory $x^\star(t_k)$ on a uniform grid $\{t_k\}_{k=0}^{N}$, the total loss is a weighted sum

$$L = \lambda_\mathrm{data} L_\mathrm{data} + \lambda_\mathrm{phys} L_\mathrm{phys} + \lambda_\mathrm{bc} L_\mathrm{bc} + \lambda_\mathrm{quat} L_\mathrm{quat} + \lambda_\mathrm{mass} L_\mathrm{mass} + \cdots,$$

where the ellipsis denotes optional soft constraints and smoothing terms. The data term L_data is a component-weighted mean-squared error between predicted

and true states, with separate group weights for translation, rotation, and mass so that altitude, vertical velocity, and attitude can be emphasised. The physics term 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,

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

using either known controls or, in the current dataset, a zero-control surrogate consistent with the reference trajectories.

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. Additional soft losses are available to enforce vertical motion structure, position–velocity consistency, temporal smoothing of altitude and velocity, and suppression of spurious lateral velocities and accelerations during near-vertical flight. Their weights can be scheduled in time so that the model first focuses on fitting the data and coarse physics, with higher-order regularisation activated gradually later in training. Together, these elements yield a training objective that balances data fidelity with dynamical consistency and structural robustness.

4.4 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.

5 Document Structure

The remainder of the thesis expands on the elements introduced here. A dedicated chapter presents the detailed derivation and validation of the 6-DOF

dynamics model. Subsequent chapters describe the optimal control formulation and numerical methods, the data generation process and learned surrogate models, and finally the numerical results, validation studies, and conclusions.