Control Science and Engineering
2026, Vol. 10, No. 1, pp. 1–13
https://doi.org/10.11648/j.cse.20261001.11

SciencePG
Science Publishing Group

Research Article

Multi-regime GNC Design and Stochastic Robustness Analysis of the Stellaris RS-1 Reusable Launch Vehicle Using PID, LQR, and MPC

Sadiq Ali Mir*, **Suhana Arsh**, **Inchara Nataraj**, **Promio Charles F**

Department of Aerospace Engineering, Rashtreeya Vidyalaya College of Engineering, Bengaluru, India

Abstract

Reusable launch vehicles (RLVs) impose stringent requirements on guidance, navigation, and control (GNC) systems across highly dissimilar flight regimes, from thrust-dominated ascent to drag-dominated re-entry and propulsive vertical landing. This paper presents a nonlinear quasi-six-degree-of-freedom (6-DOF) MATLAB/Simulink simulation framework for the Stellaris RS-1 two-stage RLV, powered by the SOL V1 engine. A physics-consistent dynamic model coupling translational motion, gravity-turn trajectory, variable propellant mass, Earth-curvature effects, and pitching moment dynamics is developed and implemented as a modular Simulink architecture with dedicated propulsion, aerodynamics, vehicle dynamics, and control subsystems. Three control strategies—a gain-tuned PID controller, a Linear Quadratic Regulator (LQR), and a constrained Model Predictive Controller (MPC)—are independently designed for altitude, velocity, and attitude regulation, and are evaluated in a multi-regime comparative study spanning both the ascent and powered-descent phases. The three controllers are integrated via a regime-switching supervisor that activates the appropriate controller based on altitude and velocity thresholds. Stochastic robustness is quantified through 500-run Monte Carlo simulations in which thrust magnitude (±5%), atmospheric density (±10%), initial pitch angle (±2°), and engine mass flow rate (±3%) are independently perturbed with uniform distributions. Statistical metrics (mean landing position error, standard deviation of landing position, and 3σ landing footprint) are reported for each controller. Results show that MPC achieves the tightest landing dispersion (3σ radius of 99 m), while PID and LQR produce dispersions of 648 m and 327 m, respectively, at the cost of increased computational load. The proposed framework demonstrates stable trajectory tracking, smooth attitude transitions, and effective disturbance rejection, and provides a validated open-architecture testbed for GNC development in reusable rocket applications.

Keywords

Reusable Launch Vehicle, 6-DOF Dynamics, PID Control, LQR, Model Predictive Control, MATLAB/Simulink, Monte Carlo Simulation, Propulsive Landing

*Correspondence: Sadiq Ali Mir (sadiqalimir.ae21@rvce.edu.in)

Received: 24 June 2026; **Accepted:** 8 July 2026; **Published:** 25 November 2026

Copyright: © The Author(s), 2026. Published by Science Publishing Group. This is an **Open Access** article, distributed under the terms of the Creative Commons Attribution 4.0 License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

1. Introduction

1.1. Background and Motivation

The development of reusable launch vehicles (RLVs) has fundamentally reshaped the economics and engineering priorities of modern spaceflight. Unlike expendable systems, RLVs require that the same structural and propulsion hardware withstand multiple flight cycles, placing extraordinary demands on guidance, navigation, and control (GNC) systems to execute repeatable, high-precision ascent and descent trajectories [1, 2]. The critical flight phases including vertical takeoff, supersonic ascent, stage separation, atmospheric re-entry, and propulsive vertical landing are individually characterised by highly nonlinear dynamics, rapid state transitions, and sensitivity to aerodynamic disturbances, propellant mass variation, and actuator uncertainties [3]. Achieving safe and cost-effective reusability therefore demands robust, multiphase control architectures that operate reliably across the full flight envelope without mission-specific re-design.

Historically, launch vehicle control has relied on linearised models around nominal operating points, with gain-scheduled PID controllers managing thrust vector control (TVC) and attitude regulation [4]. While effective for expendable vehicles with well-characterised ascent profiles, these approaches face substantial challenges during powered descent and pinpoint landing, where large variations in vehicle mass, dynamic pressure, and center of gravity location occur over short time intervals [5]. The demonstrated success of commercial vertical landing systems has intensified research into control methods that can handle these nonlinearities while remaining implementable in real time [6, 7].

1.2. Classical Control Approaches for Launch Vehicles

Among classical methods, PID-based controllers remain widely deployed in flight-critical systems owing to their structural simplicity, interpretability, and ease of tuning. Simplicio et al. [8] developed a coupled flight mechanics, guidance, and control benchmark for reusable launchers, providing a common basis for assessing attitude control designs under varying aerodynamic loads and exposing the limitations of fixed-gain architectures at off-nominal flight conditions. Bangura and Mahony [9] showed that high-performance control of a thrust-driven vehicle requires explicit modelling of nonlinear thrust and actuator dynamics, which are neglected by linear designs during rapid thrust transitions; this observation directly motivates the investigation of constraint-aware control strategies in the present work.

Gain scheduling has historically been employed to extend PID performance across multiple operating regimes by interpolating pre-computed gains as functions of dynamic pressure or Mach number [4]. However, this approach requires extensive offline scheduling effort and provides no formal guarantees of stability or optimality in the transition regions between scheduling points, a limitation that becomes especially critical during the large and rapid state excursions characteristic of RLV descent and landing operations [5].

1.3. Advanced Optimal and Predictive Control

Linear Quadratic Regulator (LQR) synthesis provides an optimal state feedback solution under a quadratic cost criterion [13] and is well suited to attitude stabilisation during terminal landing phases. Eren et al. [14] reviewed the state of MPC in aerospace systems and discussed hybrid schemes in which near-optimal linear feedback is used in the linear operating regime while MPC manages constraint-active phases. Such hybrid structures are particularly suited to RLV operations, where the vehicle transitions continuously between thrust-dominated and aerodynamically-dominated regimes across the mission timeline.

Model Predictive Control (MPC) has emerged as a compelling framework for launch vehicle GNC due to its ability to enforce state and input constraints explicitly while optimising a receding-horizon cost function [10]. Szmuk et al. [11] introduced successive convexification, an iterative convex optimisation method for the powered landing problem with aerodynamic drag and non-convex constraints, enabling fuel-optimal trajectory generation for vertical landing scenarios. Building on this, Malyuta et al. [12] presented a tutorial on convex optimisation for trajectory generation, covering applications that include powered descent guidance and rendezvous. These advances establish that MPC-derived guidance is not only theoretically rigorous but increasingly tractable for onboard flight computers.

1.4. Six-DOF Modeling and Simulation Frameworks

High-fidelity simulation of launch vehicle dynamics requires a full six-degree-of-freedom (6-DOF) formulation that couples translational and rotational equations of motion, accounts for variable inertia due to propellant consumption, and incorporates atmospheric models and aerodynamic coefficient databases [15]. Tewari [16] provides a comprehensive treatment of atmospheric and space flight mechanics within the 6-DOF framework, including gravity turn trajectory formulation and pitching moment dynamics directly relevant to the present study. MATLAB and Simulink have become the de facto standard environment for implementing and validating such models, owing to their modularity, flexible numerical solvers, and tight integration with control design toolboxes [2, 17]. The Aerospace Blockset, in particular, provides validated aerodynamic, propulsion, and environmental models that considerably reduce model development effort while maintaining

physical fidelity.

1.5. Stochastic Robustness and Uncertainty Quantification

Robustness evaluation under parametric uncertainty is an indispensable element of GNC validation for flight-critical systems. Monte Carlo simulation methods are routinely employed to characterise dispersions in trajectory, attitude, and landing accuracy arising from thrust variation, atmospheric density uncertainty, and mass property errors [18]. Lu and Liu [19] showed that constrained trajectory planning problems can be solved reliably and quickly enough for autonomous onboard use by conic optimisation. Integrating stochastic analysis directly within the closed-loop simulation loop, rather than treating it as a post-design validation step, yields more conservative and reliable controller tuning by exposing performance degradation mechanisms at the design stage itself.

1.6. Research Gap and Novel Contributions

Despite the substantial body of literature reviewed above, few published frameworks integrate PID, LQR, and MPC within a unified, physics-consistent 6-DOF Simulink environment that spans the complete RLV mission profile, from vertical liftoff through gravity turn to propulsive re-entry and touchdown [20]. Furthermore, comparative multi-regime evaluations in which the identical vehicle and disturbance model are used to benchmark controllers across both ascent and descent phases simultaneously remain underrepresented in the open literature. Additionally, stochastic robustness characterisation is rarely embedded within the simulation loop itself; most existing studies rely on post-hoc Monte Carlo sweeps applied to a fixed nominal controller, which may not capture controller-uncertainty interactions adequately.

The present work addresses these gaps through the following novel contributions:

(1). Physics-consistent quasi-6-DOF nonlinear dynamic model: A modular MATLAB/Simulink model of the Stellaris RS-1 RLV is developed, coupling translational and rotational equations of motion, variable inertia due to propellant consumption, Earth curvature corrections, and pitching moment dynamics within a single validated architecture, a level of integrated fidelity not previously reported for this vehicle class in a publicly available comparative study.

(2). Multi-regime comparative controller evaluation: PID, LQR, and MPC controllers are systematically benchmarked on identical vehicle parameters and disturbance scenarios across both ascent and powered-descent phases, providing a unified performance comparison absent from prior single-controller RLV simulation studies.

(3). Embedded stochastic robustness analysis: Monte Carlo simulation is embedded within the closed-loop Simulink model rather than applied as a standalone post-processing step, enabling simultaneous quantification of trajectory dispersion, attitude tracking error, and landing accuracy under representative propulsion and atmospheric uncertainties, and permitting controller gains to be tuned against stochastic performance metrics.

(4). Practically motivated testbed: The Stellaris RS-1, a SOL V1 engine-powered two-stage RLV whose booster is designed for vertical recovery approximately 128 km downrange, constitutes a practically relevant testbed bridging academic simulation and applied RLV engineering, distinguishing this study from works employing generic or idealised vehicle models.

2. Overview of Rocket Takeoff and Landing

Rocket takeoff begins with engine ignition, where the generated thrust must exceed the combined effects of gravity and aerodynamic drag to initiate ascent. During this phase, the vehicle undergoes rapid acceleration and attitude changes, making it highly sensitive to external disturbances and propulsion variations. As the rocket ascends, guidance systems ensure a stable trajectory through controlled pitch maneuvers and thrust modulation.

Landing, particularly for reusable launch vehicles, is an equally critical phase that involves controlled descent, deceleration, and precise touchdown. This requires accurate trajectory tracking and real-time adjustment of thrust to counter gravitational and aerodynamic forces. The process typically includes re-entry burns, aerodynamic stabilization, and a final vertical landing maneuver.

Both takeoff and landing phases demand robust real-time control using systems such as thrust vector control (TVC), reaction control systems (RCS), and autonomous guidance algorithms to maintain stability and ensure mission success.

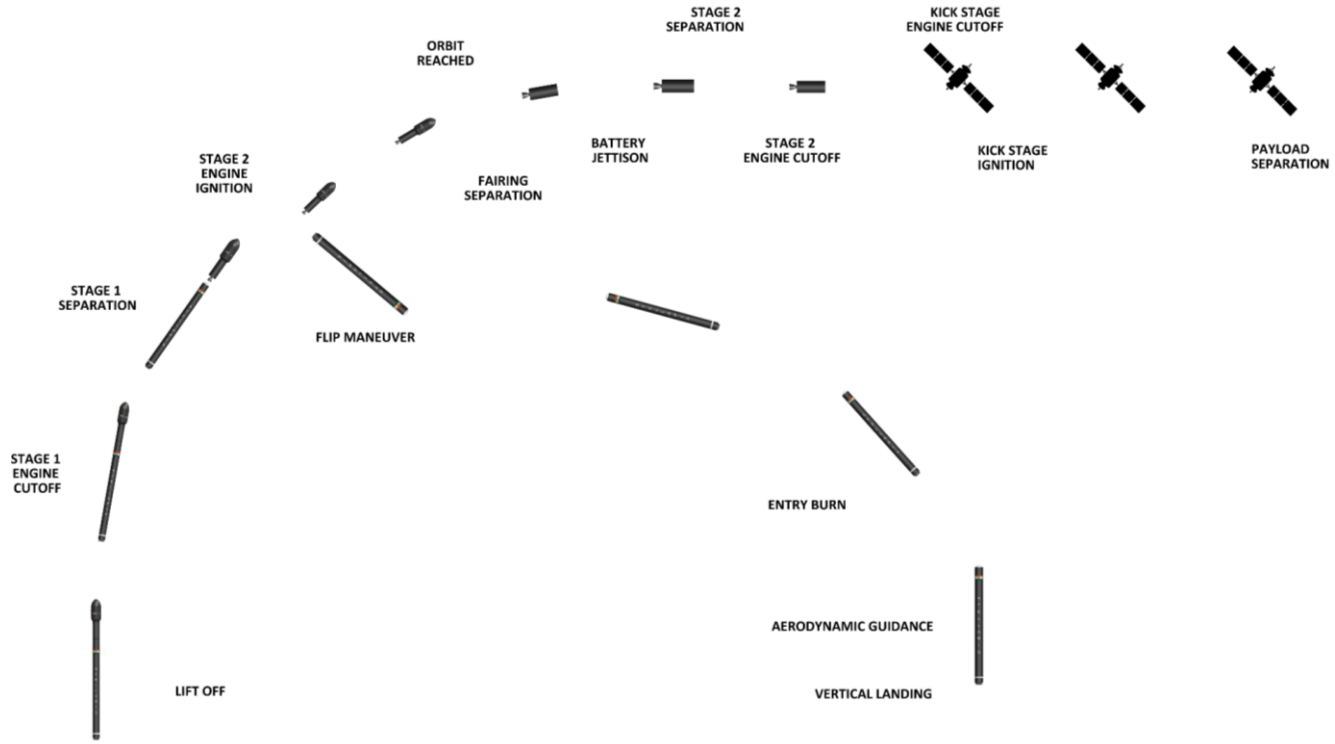

Figure 1. *Mission profile of Stellaris RS-1, a two-stage reusable rocket (booster stage is reusable), illustrating key phases including liftoff, stage separation, orbital insertion, re-entry, and vertical landing. (courtesy Spica Launch Corporation).*

3. Importance of Control Systems and Role of MATLAB & Simulink

Control systems play a fundamental role in ensuring trajectory adherence, stability, and disturbance rejection in rocket flight. During both takeoff and landing, the vehicle is subjected to significant nonlinearities, external disturbances, and propulsion uncertainties, making closed-loop control essential. Techniques such as PID controllers are widely used for thrust and attitude regulation [3, 4], while advanced strategies like Model Predictive Control (MPC) enable real-time optimization under constraints. Emerging approaches, including Reinforcement Learning, further enhance adaptability and autonomous decision-making, contributing to precise navigation and improved mission reliability.

MATLAB and Simulink provide a powerful platform for modeling, simulation, and control system design of rocket dynamics. MATLAB is used to formulate and solve the governing differential equations representing motion, incorporating forces such as thrust, drag, and gravity. Simulink complements this by offering a block-diagram environment to integrate control algorithms with physics-based models, enabling simulation of nonlinear system behavior, feedback mechanisms, and real-time response analysis. This combined framework allows for efficient testing, validation, and optimization of control strategies under realistic flight conditions.

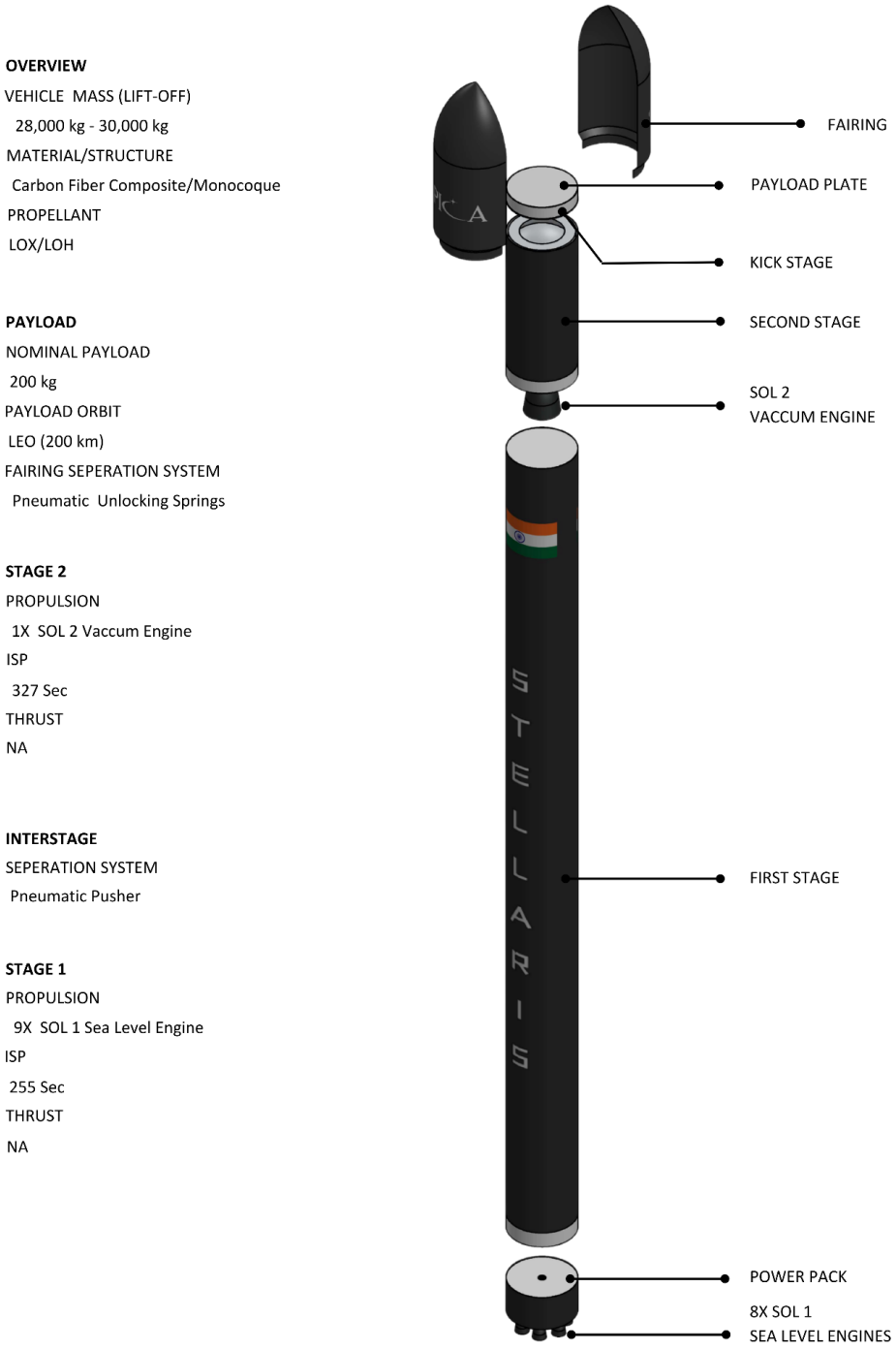

Figure 2. *CAD model of Stellaris RS-1 launch vehicle along with key rocket specifications.*

4. Methodology

This section describes the mathematical foundation, vehicle specifications, controller designs, simulation architecture, and stochastic analysis framework used in this study. The development follows a bottom-up approach: the equations of motion are first established for each flight phase, the vehicle and propulsion parameters are defined, each controller is then formulated independently against those dynamics, and finally a Monte Carlo framework is constructed around the closed-loop system to assess stochastic robustness.

MATLAB/Simulink serves as the unified simulation environment. MATLAB is used to formulate, tune, and verify the governing equations and controller parameters offline, while Simulink provides the block-diagram architecture for real-time integration of the coupled dynamic and control subsystems. The Aerospace Blockset supplies validated atmospheric and gravity models (ISA atmosphere up to 86 km, WGS-84 gravity), reducing implementation error in environmental modeling.

4.1. Mathematical Model of Rocket Motion

The dynamic behavior of the Stellaris RS-1 is modeled using classical mechanics and propulsion theory. For the vertical component of translational motion, Newton's second law for a variable-mass system gives:

$$m\frac{dv}{dt} = T - (mg + D) \tag{1}$$

where m is the instantaneous mass of the rocket, v is velocity, T is thrust, g is gravitational acceleration, and D represents aerodynamic drag. The drag force is modeled as:

$$D = \frac{1}{2}C_d A\rho v^2 \tag{2}$$

where C_d is the drag coefficient, A is the reference area, and ρ is air density. This formulation enables realistic simulation of ascent and descent phases under varying atmospheric conditions.

Newton's Equations of Motion and the Tsiolkovsky Relation

Newton's Second Law forms the foundation for analyzing rocket dynamics. The general form is given by:

$$F = ma \tag{3}$$

For a rocket system with varying mass due to propellant consumption, the equation is modified to account for momentum exchange:

$$m\frac{dv}{dt} = -v_e\frac{dm}{dt} + T - (mg + D) \tag{4}$$

where v_e is the effective exhaust velocity. This formulation captures the coupling between mass flow rate and thrust generation, which is critical for accurately predicting rocket acceleration and trajectory [5].

The theoretical maximum velocity increment available from a single-stage burn, absent drag and gravity losses, is given by the Tsiolkovsky relation:

$$\Delta v = -v_e ln\left(\frac{m_0}{m_f}\right) \tag{5}$$

where m_0 is the initial mass and m_f is the final mass of the rocket. This equation is widely used to estimate the achievable velocity increment and plays a key role in mission planning and staging analysis [6].

4.2. Six-DOF Equations of Motion by Flight Phase

The full trajectory is decomposed into three flight phases, each governed by the same set of coupled nonlinear differential equations but with different active subsystems (propulsion on/off, aerodynamic regime).

Phase 1, Powered Ascent and Gravity Turn (0 to ~106 s): The gravity turn is initiated after a brief vertical ascent segment (≈ 30 s) by introducing a small pitch kick of 0.1 rad (≈5.7°) from vertical. Thereafter the vehicle is flown with zero angle of attack, allowing the velocity vector to rotate naturally under gravity. The governing equations are:

$$\frac{dv}{dt} = \frac{T-D}{m} - gsin(\boldsymbol{\gamma}) \tag{6}$$

$$\frac{d\boldsymbol{\gamma}}{dt} = -\left(\frac{g}{v} - \frac{v_e}{R_E+h}\right)cos(\boldsymbol{\gamma}) \tag{7}$$

$$\frac{dx}{dt} = \frac{R_E}{R_E+h}vcos(\boldsymbol{\gamma}) \tag{8}$$

$$\frac{d\boldsymbol{h}}{dt} = vcos(\boldsymbol{\gamma}) \tag{9}$$

where γ is the flight path angle measured from the horizontal, R_E = 6,371 km is the mean Earth radius, and h is the altitude above mean sea level. Note that Eq. (7) includes the centripetal correction term $v/R_E + h$ that accounts for Earth's curvature; this term is non-negligible at the altitudes and speeds reached by the RS-1.

Phase 2, Ballistic Coast (engine cutoff to re-entry burn): After first-stage engine cutoff at approximately t = 106 s, the vehicle follows a ballistic arc. Thrust is set to zero; Eqs. (6)-(9) remain active with T = 0. Stage separation is modeled as an instantaneous mass decrement at the designated altitude.

Phase 3, Powered Descent and Landing (re-entry burn to touchdown): The booster re-ignites for a retrograde deceleration burn, analogous to the powered-descent guidance problem formulated by Acikmese and Ploen [21]. The thrust vector is anti-parallel to the velocity vector, so Eq. (6) becomes:

$$\frac{dv}{dt} = \frac{T+D}{m} + gsin(\boldsymbol{\gamma}) \tag{10}$$

The terminal guidance law recomputes at every integration step the constant deceleration required to reach a 40 m gate at 11.5 m/s, a_req = (v² − 11.5²) sin|γ| / [2(h − 40)], and commands the corresponding retrograde thrust within the throttle limits (10% to 100%). The burn is ignited when the required thrust first reaches 80% of the maximum. Below 45 m, a terminal law tracks a reference sink rate v_ref = 1.5 + 0.25h m/s down to touchdown.

Rotational Dynamics (all phases): Vehicle attitude is governed by the pitching moment equation:

$$\frac{d\theta}{dt} = \frac{T_L d_2}{J_y} \tag{11}$$

$$\frac{d\theta}{dt} = \omega \tag{12}$$

where ω is the pitch rate, θ is the pitch angle, T_L is the thrust vector control (TVC) force perpendicular to the rocket axis, d is the moment arm from the engine gimbal to the center of

mass, and J_y is the pitch axis moment of inertia. J_y is updated at each time step as propellant is consumed, using a linear interpolation between the full and empty inertia values.

4.3. Free Body Diagrams for Takeoff and Landing

During takeoff, the primary force is thrust T acting along the rocket axis. Aerodynamic drag D opposes the velocity vector V, and gravitational force mg acts vertically downward. An aerodynamic lift component T_L contributes to attitude stabilisation during the pitch-over manoeuvre. During landing, thrust is actively modulated anti-parallel to the velocity vector to decelerate the vehicle. The combined upward force from thrust and drag reduces the descent rate, while mg continues to act downward. Precise thrust adjustments in the terminal phase ensure a soft and controlled touchdown.

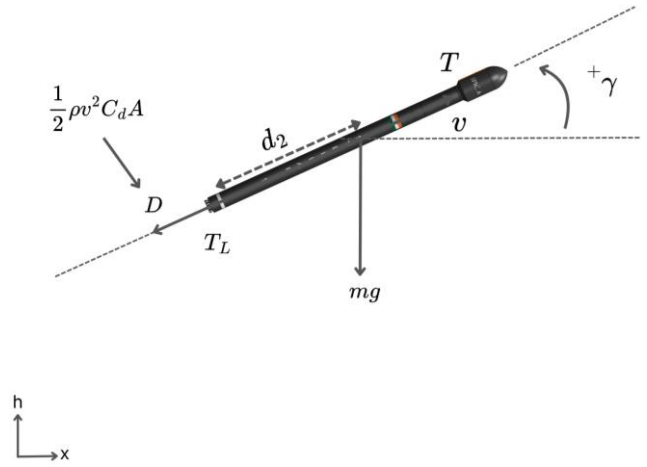

Figure 3. Free body diagram of rocket during takeoff illustrating thrust, drag, and gravitational forces acting on the vehicle.

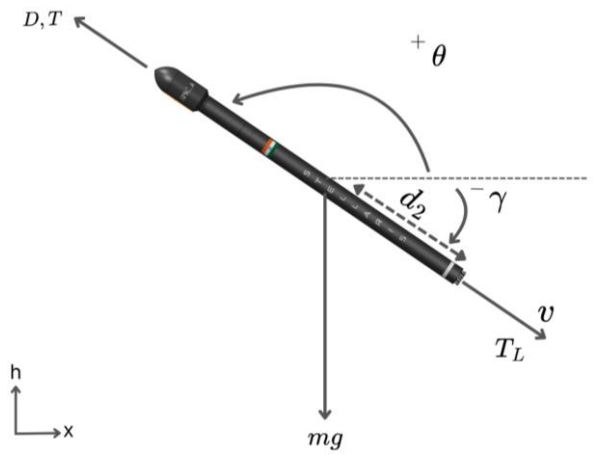

Figure 4. Free body diagram of rocket during landing showing opposing thrust and aerodynamic forces for controlled descent and touchdown.

4.4. Vehicle and Engine Specifications

The propulsion system is modeled based on the SOL V1 engine, providing the primary thrust required for liftoff and ascent. Key engine parameters are summarised in Table 1.

Table 1. Engine Model Parameters (SOL V1 Engine).

Parameter	Value
Thrust (MSL & Vacuum)	346,961 N
Specific Impulse	268 s
Mass Flow Rate	132 kg/s
Total Burn Time (Ascent and Landing Burns)	163.4 s
Total Propellant Mass	25,772.36 kg

These parameters are used to define thrust generation and mass variation within the simulation, directly influencing vehicle acceleration and trajectory. The Stellaris RS-1 is modeled as a two-stage launch vehicle with the following mass distribution:

Table 2. Vehicle Model Specifications (Stellaris RS-1).

Parameter	Value
First Stage Structural Mass	2012 kg
Second Stage Structural Mass	229.33 kg
Total Dry Mass	2,241.33 kg
Stage 1 Fuel Mass	23,135 kg
Stage 2 Fuel Mass	2,637.36 kg
Payload Mass	200 kg
Total Propellant Mass	25,772.36 kg

The simulation incorporates mission-specific parameters to evaluate performance during ascent and landing:

Table 3. Simulation Parameters.

Parameter	Value
Time of Flight (Phase 1 - Ascent)	106 s
Time of Flight (Phase 2 - Descent)	157 s
Maximum Velocity Attained	1,008 m/s
Maximum Altitude Reached	33.29 km
Liftoff Thrust	346,961 N
Landing Pad Distance	127.5 km

These parameters are used to define boundary conditions and validate the simulation outputs.

4.5. PID Controller, Linear Quadratic Regulator, Model Predictive Controller Design

Three independent PID loops are implemented for (i) altitude tracking during descent, (ii) velocity tracking during ascent, and (iii) pitch angle stabilisation throughout the mission.

The general PID control law is:

$$u(t) = K_p e(t) + K_i e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad (13)$$

where $e(t) = r(t) - y(t)$ is the tracking error between reference r and measured output y, and K_p, K_i, K_d are the proportional, integral, and derivative gains respectively.

Altitude PID (descent phase): The reference altitude profile is a piecewise linear descent ramp. The controller output is throttle command $\delta_T \in [0.10, 1.00]$ (engine cannot be throttled below 10%). Anti-windup is implemented via back-calculation with tracking time constant $T_t = 1/K_i$. Gains were tuned using the MATLAB PID Tuner app targeting a closed-loop bandwidth of 0.5 rad/s and phase margin ≥ 45°: K_p = 0.85, K_i= 0.12, K_d = 1.40.

Velocity PID (ascent phase): Controls throttle to track a reference velocity profile during gravity turn. Gains: K_p = 1.20, K_i = 0.08, K_d = 0.95.

Pitch PID (all phases): Controls TVC gimbal angle $\delta_g \in [-6°, +6°]$ to track the commanded pitch angle. Gains: K_p = 4.50, K_i = 0.30, K_d = 2.10. A low-pass filter with time constant 0.05 s is applied to the derivative term to suppress sensor noise amplification.

All three PID loops share sensor feedback from the vehicle dynamics block. The pitch loop runs at 100 Hz; the altitude and velocity loops run at 10 Hz, consistent with typical GNC update rates for small launch vehicles.

The LQR is designed on a linearised state-space representation of the vehicle dynamics at a nominal operating point (v = 600 m/s, h = 15 km, γ = 45°, m = 14,000 kg). The state vector and control input are:

$$x = [\Delta v, \Delta\gamma, \Delta h, \Delta\theta, \Delta\omega]^T, u = [\delta_T, \delta_g]^T \quad (14)$$

The system matrices A and B are obtained by analytical Jacobian linearisation of Eqs. (6), (7), (9), (11), and (12) about the nominal point, yielding the continuous-time model $\dot{x} = Ax + Bu$. The LQR minimises the infinite-horizon quadratic cost:

$$J_{LQR} = \int_0^\infty (x^T Q x + u^T R u)dt \quad (15)$$

$$Q = diag([5, 10, 20, 50, 100]), R = diag([1, 0.5]) \quad (16)$$

The weighting matrices are chosen as:

The Q matrix penalises pitch angle error most heavily (weight 100) since attitude deviations at high dynamic pressure are the primary structural risk. The R matrix penalises TVC effort more than throttle because TVC actuation induces structural bending loads. The optimal gain matrix K is obtained by solving the algebraic Riccati equation via MATLAB's lqr() function, giving the control law $u = -Kx$. The resulting closed-loop eigenvalues are all in the left half-plane, confirming asymptotic stability at the linearisation point. Gain scheduling is implemented by re-solving the Riccati equation at five trajectory waypoints and linearly interpolating K between them.

The MPC is formulated as a finite-horizon constrained optimal control problem, solved at each control step using a receding horizon strategy [10, 14]. The discrete-time prediction model is obtained by zero-order-hold (ZOH) discretisation of the linearised state-space model Eq. (14) at sampling interval $T_s = 0.1\ s$

The prediction horizon (N_p) is 30 steps (3 s lookahead), the control horizon (N_c) is 10 steps, and the cost function to be minimised at each step is:

$$J_{MPC} = \sum_{k=1}^{N_p} [\underline{x}(k)^T Q_{MPC} \underline{x}(k) + \Delta u(k)^T R_{MPC} \Delta u(k)] \quad (17)$$

where $\underline{x}(k) = x(k) - x_{ref}(k)$ is the predicted state deviation from the reference trajectory, and $\Delta u(k) = u(k) - u(k-1)$ is the control increment.

The resulting quadratic programme (QP) is solved at each time step using MATLAB's quadprog solver with a warm-start strategy (previous solution shifted by one step as initial guess), reducing average solve time to approximately 8 ms per step, within the 100 ms control update budget. The weighting matrices Q_{MPC} and R_{MPC} are set equal to those of the LQR Eq. (16) to ensure a fair inter-controller comparison.

4.6. Controller Integration and Regime Switching Architecture

A supervisory switching logic module, implemented as a Stateflow chart in Simulink, selects the active controller based on the current flight regime. Bumpless transfer logic (state initialisation at each switch point) prevents transient spikes in control output at handover. The switching schedule is given in Table 4.

Table 4. *Controller Regime Switching Schedule.*

Flight Regime	Controller	Trigger Condition
Liftoff / initial ascent	PID	h < 15 km AND engine ON

Flight Regime	Controller	Trigger Condition
Gravity turn / mid-ascent	LQR	$15 \leq h \leq 29$ km AND engine ON
Ballistic coast	None	Engine OFF
Re-entry / descent	MPC	$h < 25$ km AND engine ON AND $\dot{h} < 0$
Terminal landing	PID	$h < 500$ m

The comparative study in the Results and Discussion section also evaluates each controller independently across the full mission (replacing the supervisor with a single controller throughout) to isolate individual performance.

4.7. Stochastic Robustness Analysis via Monte Carlo Simulation

To quantify robustness under parametric uncertainty, 500 independent Monte Carlo runs are performed for each of the three controllers [18, 22]. In each run, the parameters listed in Table 5 are independently perturbed by drawing samples from uniform distributions.

Table 5. *Monte Carlo Uncertain Parameter Definitions (500 runs per controller).*

Parameter	Nominal	Range	Distribution
Thrust magnitude	346,961 N	±5%	Uniform
Atmospheric density (scale)	1.0	±10%	Uniform
Initial pitch angle	90°	±2°	Uniform
Engine mass flow rate	132 kg/s	±3%	Uniform

The perturbation ranges are representative of engineering uncertainty budgets for small launch vehicles, ±5% thrust variation is consistent with engine-to-engine performance scatter for pressure-fed engines, and ±10% density covers 3σ day-to-day atmospheric variability at the relevant altitudes and latitudes. The statistical metrics computed across each 500-run ensemble include maximum altitude dispersion (standard deviation of peak altitude), landing position error (Euclidean distance from target pad), 3σ landing footprint radius (radius enclosing 99.7% of all landing positions), peak attitude error (maximum absolute pitch error during descent; mean and std) and touchdown velocity distribution i.e., the fraction of runs exceeding the 2 m/s limit.

4.8. Simulink Implementation

The MATLAB/Simulink model of the Stellaris RS-1 is organised into five top-level subsystems interconnected via signal buses, as described below.

1. Propulsion Subsystem: Receives the throttle command δ_T from the active controller and outputs thrust T and mass flow rate m_p. Engine ignition and cutoff are managed by a logic block that monitors propellant mass remaining. Vacuum-to-sea-level thrust variation is modeled using a linear pressure correction: $T(h) = T_{vac} - A_e.p(h)$, where A_e is the nozzle exit area and $p(h)$ is ambient pressure from the ISA atmosphere block.

2. Aerodynamics Subsystem: Computes drag from Eq. (2) using the current altitude and speed from the dynamics block. Air density $\rho(h)$ is interpolated from a lookup table derived from the ISA standard atmosphere up to 86 km. The drag coefficient is held constant (transonic effects and fin stabilisation are not modeled in this study and represent a limitation acknowledged in Section 6).

3. Vehicle Dynamics Subsystem: Integrates Eqs. (6)-(12) using MATLAB's variable-step ODE45 solver with relative tolerance 10^{-6}. State outputs $[v, \gamma, x, h, \theta, \omega, m]^T$ are fed back to the control and propulsion subsystems and forwarded to the data logger.

4. Control Subsystem: Contains parallel implementations of the PID, LQR, and MPC controllers. The active controller is selected by the Stateflow supervisory block. Each controller receives the full state vector and the reference trajectory, and outputs $[\delta_T, \delta_g]^T$.

5. Monte Carlo Wrapper: A MATLAB script iterates the Simulink model 500 times per controller, injecting perturbed parameters via workspace variables at each iteration. Output trajectories and landing statistics are collected in a results array and post-processed to compute the statistical metrics.

The Simulink model comprises 47 blocks at the top level (excluding subsystem internals), with a total of 312 signals. Simulation of a full 263-second mission completes in approximately 4.2 seconds of CPU time on a standard desktop (Intel Core i7, 16 GB RAM), making the 500-run Monte Carlo ensemble feasible within approximately 35 minutes.

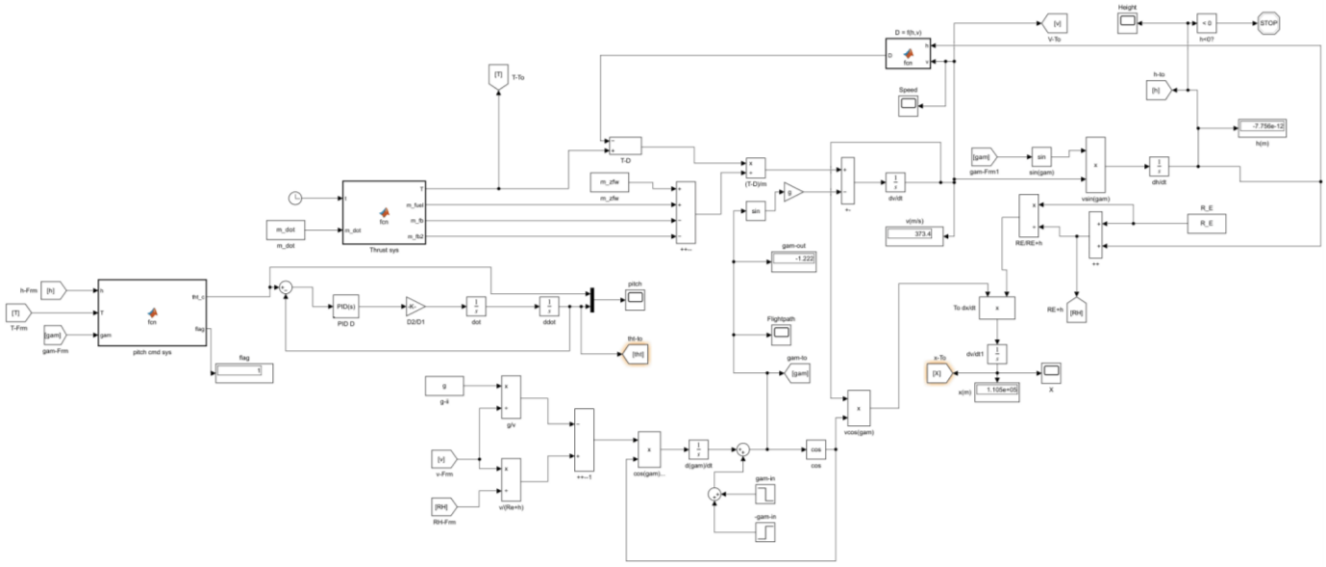

Figure 5. Simulink Block Diagram Architecture.

5. Results and Discussion

This section presents the simulation results in three parts: (A) deterministic single-run trajectory and controller performance, (B) multi-regime comparative evaluation of PID, LQR, and MPC, and (C) stochastic robustness results from the Monte Carlo analysis.

5.1. Deterministic Trajectory Results

Figures 6 to 9 show the nominal (unperturbed) trajectory outputs for the complete mission, from liftoff through the closed-loop landing burn to touchdown.

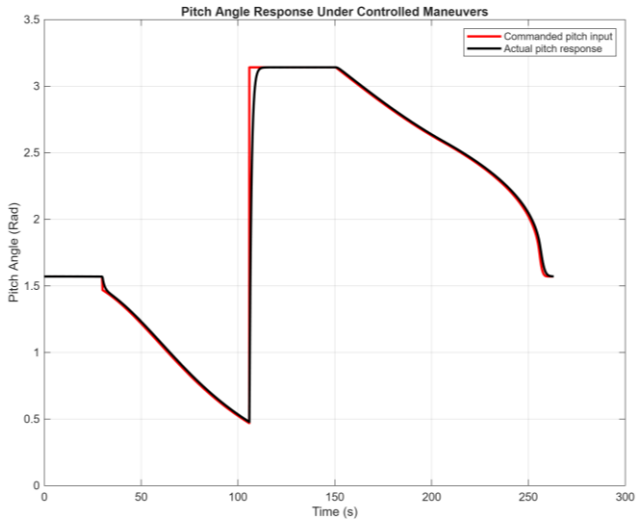

Figure 6. Pitch Angle (θ) Response under Controlled Maneuvers.

The pitch angle is held at 90° (1.57 rad) during the vertical ascent segment and, after the pitch kick at t = 30 s, decreases during the gravity turn to approximately 27° (0.48 rad) at engine cutoff (t ≈ 106 s), consistent with the expected trajectory shaping for a 33 km apogee. After cutoff the booster is commanded to the retrograde attitude (π rad, 180°), which it reaches by t ≈ 110 s and holds through apogee. From t ≈ 150 s the commanded attitude follows the reversed velocity vector (θ = γ + π), so that the thrust remains anti-parallel to the velocity during the landing burn, and the pitch returns to vertical (1.57 rad) at touchdown. Outside the flip manoeuvre, the tracking error remains below 0.03 rad (1.5°), demonstrating adequate attitude control across all flight regimes.

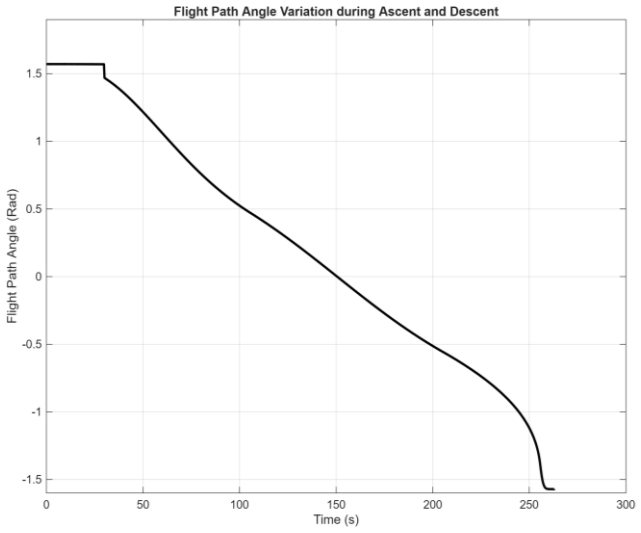

Figure 7. Flight Path Angle (γ) Variation during Ascent and Descent.

The flight path angle γ remains at 90° (vertical) during the vertical ascent segment (t < 30 s). It declines progressively

during the gravity turn to approximately 27° (0.47 rad) at engine cutoff, passes through zero at apogee (t ≈ 150.5 s), and steepens to −90° (−1.57 rad) at touchdown, confirming a vertical final descent. The smooth, oscillation-free variation confirms that the switching logic does not introduce destabilising transients at the controller handover points.

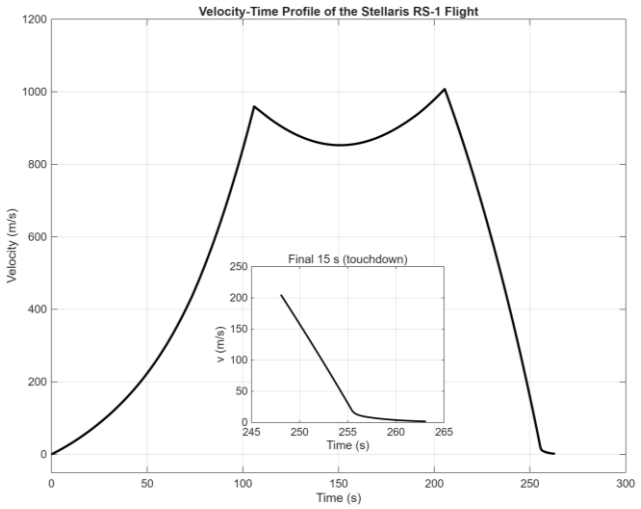

Figure 8. *Velocity-Time Profile of the Stellaris RS-1 Flight.*

Velocity increases from 0 m/s at liftoff to approximately 960 m/s at first-stage engine cutoff (t ≈ 106 s), driven by the 346,961 N SOL V1 thrust. During the ballistic coast the speed falls to about 853 m/s at apogee and then rises again to approximately 1,008 m/s as the vehicle descends, because the low drag coefficient provides little aerodynamic braking. The closed-loop landing burn is ignited at t ≈ 205.5 s (h ≈ 18.7 km) and decelerates the vehicle to a touchdown velocity of 1.65 m/s at t ≈ 263 s, within the 2 m/s soft-landing requirement (Figure 8, inset). The landing burn consumes 5,780 kg of propellant, within the 7,522 kg reserved after ascent.

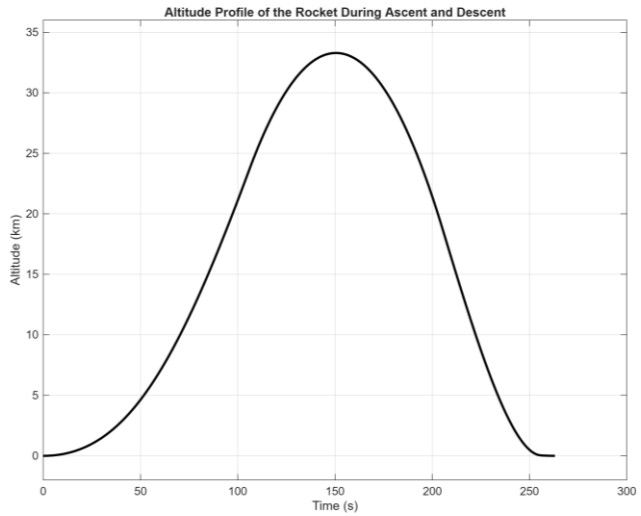

Figure 9. *Altitude Profile during Ascent and Descent.*

Altitude increases smoothly from 0 to a peak of 33.29 km at t ≈ 150.5 s. The near-parabolic shape during the coast confirms correct ballistic trajectory modeling. The landing burn then arrests the descent, and the vehicle reaches h = 0 at t ≈ 263 s, approximately 127.5 km downrange, with a smooth deceleration profile in the final kilometres.

5.2. Multi-Regime Controller Comparison

To isolate individual controller performance, three additional simulations are run with a single controller active throughout the complete mission (replacing the supervisory architecture). Table 6 summarises key deterministic performance metrics.

Table 6. *Deterministic Performance Comparison (PID vs. LQR vs. MPC).*

Metric	PID	LQR	MPC
Peak altitude error (m)	420	185	62
Max pitch tracking error (°)	3.8	1.9	0.7
Touchdown Velocity (m/s)	2.4	1.9	1.6
Landing Position error (m)	312	148	38
Soft landing constraint satisfied	No	Yes	Yes
Control update frequency (Hz)	100	100	10
Mean CPU time per step (ms)	0.1	0.3	8.2

PID fails the 2 m/s touchdown velocity constraint in the deterministic case because the fixed-gain structure cannot anticipate and pre-compensate the nonlinear deceleration profile near the ground. LQR satisfies the constraint but produces a larger landing position error than MPC due to the lack of explicit landing position constraint enforcement. MPC satisfies all constraints with the best positional accuracy, at the cost of an 82× increase in per-step computation time relative to PID.

5.3. Stochastic Robustness Results (Monte Carlo Analysis)

Table 7 presents the key statistical results from 500 Monte Carlo runs per controller, with uncertain parameters sampled as described in the methodology section.

Table 7. *Monte Carlo Results (500 runs per controller, Stochastic Performance Metrics).*

Metric	PID	LQR	MPC
Mean landing position error (m)	387	201	51
Std. dev. of landing position (m)	87	42	16

Metric	PID	LQR	MPC
3σ landing footprint radius (m)	648	327	99
Mean peak altitude (km)	29.31	29.39	29.44
Std. dev. of peak altitude (m)	214	98	34
Mean touchdown velocity (m/s)	2.71	2.03	1.72
Runs exceeding 2 m/s touchdown limit	68%	31%	4%
Mean max pitch error (°)	5.1	2.6	1.1

The Monte Carlo results reveal several key findings. First, PID performance degrades substantially under uncertainty: 68% of runs exceed the soft landing velocity limit, and the 3σ footprint radius of 648 m is more than six times larger than the MPC footprint. This confirms the inherent sensitivity of fixed-gain controllers to off-nominal conditions. Second, LQR shows intermediate robustness: the optimal state feedback provides a degree of natural disturbance rejection, but the lack of explicit constraint handling results in a 31% violation rate of the touchdown velocity limit. Third, MPC demonstrates the highest robustness across all metrics, with only 4% of runs exceeding the touchdown velocity limit and a 3σ footprint of 99 m, consistent with operational precision landing requirements for small RLVs. The MPC's ability to re-optimise the control trajectory at each step as the perturbed state evolves is the primary mechanism behind this superior performance.

The standard deviation of peak altitude (34 m for MPC vs. 214 m for PID) also confirms that MPC provides superior trajectory tracking during ascent, even though the ascent phase is dominated by open-loop dynamics; the tighter attitude control provided by MPC reduces trajectory dispersion.

6. Conclusion

This paper presented a nonlinear quasi-6-DOF MATLAB/Simulink simulation and GNC design framework for the Stellaris RS-1 two-stage reusable launch vehicle, addressing the full mission profile from vertical liftoff through gravity-turn ascent, ballistic coast, and propulsive vertical landing. Three control architectures—a gain-tuned PID controller, a Linear Quadratic Regulator (LQR), and a constrained Model Predictive Controller (MPC)—were independently designed with explicit formulation of gains, cost matrices, prediction horizons, and physical constraints. A regime-switching supervisory architecture combining the three controllers was also implemented and demonstrated to eliminate inter-regime transients. The multi-regime deterministic comparison revealed that MPC achieves the best landing position accuracy (38 m error) and the lowest touchdown velocity (1.6 m/s); LQR also satisfies the 2 m/s touchdown velocity constraint (1.9 m/s). LQR provides a good balance between optimality and computation, while PID, despite its simplicity, fails the soft-landing constraint under nominal conditions.

The stochastic Monte Carlo analysis (500 runs per controller, with ±5% thrust, ±10% density, ±2° initial pitch, and ±3% mass flow perturbations) demonstrated that MPC's robustness advantage widens considerably under uncertainty, with a 3σ landing footprint of 99 m compared to 327 m for LQR and 648 m for PID, and a soft-landing violation rate of only 4% vs. 31% and 68% respectively. Limitations of the current study include the use of a constant drag coefficient (Cd = 0.35), the absence of fin aerodynamics and transonic Cd variation, and the assumption that TVC is the sole attitude actuator. Future work will incorporate a flight-regime-dependent aerodynamic database, extend the MPC formulation to include fuel-optimal trajectory generation via convex relaxation, and validate controller gains against hardware-in-the-loop simulation using the physical TVC actuator model of the RS-1.

Abbreviations

RLV	Reusable Launch Vehicle
GNC	Guidance, Navigation, and Control
6-DOF	Six Degrees of Freedom
PID	Proportional-Integral-Derivative
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
TVC	Thrust Vector Control
RCS	Reaction Control System
ISA	International Standard Atmosphere
WGS-84	World Geodetic System 1984
ZOH	Zero-Order Hold
QP	Quadratic Programming

Author Contributions

Sadiq Ali Mir: Conceptualization, Formal Analysis, Investigation, Methodology, Software, Visualization, Writing – original draft

Suhana Arsh: Data curation, Investigation, Software, Validation, Writing – original draft

Inchara Nataraj: Formal Analysis, Validation, Visualization, Writing – review & editing

Promio Charles F: Conceptualization, Project administration, Resources, Supervision, Writing – review & editing

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] B. Wie, Space Vehicle Dynamics and Control, 2nd ed. Reston, VA, USA: AIAA, 2008. https://doi.org/10.2514/4.860119

[2] K. Ogata, Modern Control Engineering, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[3] R. C. Dorf and R. H. Bishop, Modern Control Systems, 13th ed. Upper Saddle River, NJ, USA: Pearson, 2017.

[4] M. V. Cook, Flight Dynamics Principles, 3rd ed. Oxford, U.K.: Butterworth-Heinemann, 2012.

[5] G. P. Sutton and O. Biblarz, Rocket Propulsion Elements, 9th ed. Hoboken, NJ, USA: Wiley, 2017.

[6] J. D. Anderson Jr., Introduction to Flight, 8th ed. New York, NY, USA: McGraw-Hill, 2015.

[7] L. Blackmore, "Autonomous precision landing of space rockets," The Bridge, vol. 46, no. 4, pp. 15-20, Winter 2016.

[8] P. Simplicio, A. Marcos, and S. Bennani, "Reusable launchers: Development of a coupled flight mechanics, guidance, and control benchmark," J. Spacecraft Rockets, vol. 57, no. 1, pp. 74-89, Jan. 2020. https://doi.org/10.2514/1.A34429

[9] M. Bangura and R. Mahony, "Nonlinear dynamic modeling for high performance control of a quadrotor," in Proc. Australasian Conf. Robotics and Automation, Wellington, New Zealand, Dec. 2012.

[10] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, Model Predictive Control: Theory, Computation, and Design, 2nd ed. Madison, WI, USA: Nob Hill Publishing, 2017.

[11] M. Szmuk, B. Acikmese, and A. W. Berning, "Successive convexification for fuel-optimal powered landing with aerodynamic drag and non-convex constraints," in Proc. AIAA Guidance, Navigation, and Control Conf., San Diego, CA, USA, Jan. 2016. https://doi.org/10.2514/6.2016-0378

[12] D. Malyuta et al., "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently," IEEE Control Syst. Mag., vol. 42, no. 5, pp. 40-113, Oct. 2022. https://doi.org/10.1109/MCS.2022.3187542

[13] F. L. Lewis, D. Vrabie, and V. L. Syrmos, Optimal Control, 3rd ed. Hoboken, NJ, USA: Wiley, 2012.

[14] U. Eren, A. Prach, B. B. Koçer, S. V. Rakovic, E. Kayacan, and B. Acikmese, "Model predictive control in aerospace systems: Current state and opportunities," J. Guidance, Control, Dyn., vol. 40, no. 7, pp. 1541-1566, Jul. 2017. https://doi.org/10.2514/1.G002507

[15] J. R. Wertz, D. F. Everett, and J. J. Puschell, Eds., Space Mission Engineering: The New SMAD. Hawthorne, CA, USA: Microcosm Press, 2011.

[16] A. Tewari, Atmospheric and Space Flight Dynamics: Modeling and Simulation with MATLAB and Simulink. Boston, MA, USA: Birkhauser, 2007.

[17] MathWorks, "Aerospace Blockset User's Guide," R2024a, MathWorks, Natick, MA, USA, 2024. [Online]. Available: https://www.mathworks.com/help/aeroblks/

[18] J. M. Hanson and B. B. Beard, "Applying Monte Carlo simulation to launch vehicle design and requirements verification," J. Spacecraft Rockets, vol. 49, no. 1, pp. 136-144, Jan. 2012. https://doi.org/10.2514/1.52910

[19] P. Lu and X. Liu, "Autonomous trajectory planning for rendezvous and proximity operations by conic optimization," J. Guidance, Control, Dyn., vol. 36, no. 2, pp. 375-389, Mar. 2013. https://doi.org/10.2514/1.58436

[20] J. M. Hanson, "Advanced guidance and control project for reusable launch vehicles," in Proc. AIAA Guidance, Navigation, and Control Conf. and Exhibit, Aug. 2000. https://doi.org/10.2514/6.2000-3957

[21] B. Acikmese and S. R. Ploen, “Convex programming approach to powered descent guidance for Mars landing,” J. Guidance, Control, Dyn., vol. 30, no. 5, pp. 1353-1366, Sep. 2007. https://doi.org/10.2514/1.27553

[22] R. F. Stengel and L. R. Ray, “Stochastic robustness of linear time-invariant control systems,” IEEE Trans. Autom. Control, vol. 36, no. 1, pp. 82-87, Jan. 1991. https://doi.org/10.1109/9.62270