Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Mathematical Simulation Framework for Comparative Evaluation of Rocket Control Strategies: From Classical PID to Hybrid Adaptive Architectures

Lucas Lima Freitag

houslast@gmail.com

UFRN https://orcid.org/0009-0006-8849-5619

Research Article

Keywords: Rocket control simulation, VTVL, mathematical modeling, PID variants, sliding mode control, fuzzy control, model predictive control, open-source simulation software, control system benchmarking

Posted Date: July 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10309906/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

A Mathematical Simulation Framework for Comparative Evaluation of Rocket Control Strategies: From Classical PID to Hybrid Adaptive Architectures

Anonymous Author(s)[1]

Abstract: This paper presents a mathematical simulation framework implemented as an open-source web platform for the systematic evaluation and benchmarking of rocket control algorithms. The core of the framework is a set of mathematical models–six-degree-of-freedom rigid-body dynamics with fourth-order Runge-Kutta integration, International Standard Atmosphere, von Kármán turbulence, structural vibrations, and sensor noise–encoded in JavaScript and executable in any modern web browser. Nineteen control strategies are implemented and compared across launch and landing scenarios using quantitative metrics including settling time, RMS tracking error, fuel economy, attitude stability, noise rejection, and a composite overall score. Results show PID-based architectures dominate both scenarios (scores 81-82/100 launch, 74-75/100 landing), while novel variants such as Impulse control reduce landing fuel consumption by 12.7% versus standard PID. The framework establishes a reproducible benchmark for control algorithm development and demonstrates how mathematical models in open-source simulators enable rigorous testing of physical conditions, rapid prototyping of new algorithms, and transparent scientific validation.

Keywords: Rocket control simulation, VTVL, mathematical modeling, PID variants, sliding mode control, fuzzy control, model predictive control, open-source simulation software, control system benchmarking.

Running title: Mathematical Simulation Framework for Rocket Control

Acknowledgements:

1. Introduction

The reusability of launch vehicles represents one of the most significant advancements in spaceflight, with systems such as the SpaceX Falcon 9 demonstrating that vertical take-off and vertical landing (VTVL) is both technically and economically viable [1]. The powered descent phase presents a uniquely challenging control problem: the vehicle must simultaneously regulate altitude, velocity, and attitude under rapidly changing mass properties, nonlinear aerodynamics, atmospheric disturbances, and sensor noise–all within tight fuel constraints.

Mathematical simulation plays a critical role in the development and validation of control systems for such applications. High-fidelity simulators allow engineers to evaluate controller performance across thousands of scenarios without the cost and risk of physical testing, to rapidly prototype new algorithms, and to perform systematic comparisons that would be impractical

[1] Affiliations omitted for double-blind review.

with hardware-in-the-loop systems alone [2]. The development of open, extensible simulation frameworks is particularly important, as it enables reproducible benchmarking and accelerates the transfer of theoretical advances into practical applications.

Despite the availability of numerous control strategies in the literature–from classical PID [3] and LQR [4] to sliding mode [5], MPC [6], fuzzy logic, and reinforcement learning [7]–most comparative studies examine only two or three methods within a single framework [8]. This narrow scope makes it difficult to understand relative strengths and weaknesses under identical conditions.

This work makes four primary contributions:

1. **A mathematical simulation framework** for VTVL rocket control that models 6-DOF rigid-body dynamics with RK4 integration, realistic atmospheric and disturbance models, sensor noise, and structural vibrations. The framework is implemented as an open-source web platform in JavaScript, executable in any browser without external dependencies.

2. **Ten novel controller variants** beyond standard implementations: Adapt-SMC (adaptive gain sliding mode), Auto-Iter (phase-based iterative learning), Fuel-Saver (four-zone coasting PID), Hybrid-IL (noise prediction with iterative correction), Impulse (bang-bang with coast phases), Neural (random-weight feedforward network), Noise-Pred (AR(4) cascade with PID), RL-HC (reinforcement heuristic for gain tuning), Short-Act (quantized deadzone control), and Smart-PID (zone-dependent integral management).

3. **Systematic comparison of 19 controllers** across two complete mission phases using seven quantitative metrics.

4. **Reproducibility**: the complete simulation source code, all control model implementations, and raw trajectory data are provided as supplementary material.

2. Mathematical Framework of the Simulator

The simulation framework is implemented in JavaScript with a modular architecture comprising three layers: the physics engine, the control model interface, and the simulation runner with data logging. All components communicate through well-defined interfaces, enabling new control models to be added without modifying the physics infrastructure.

The physics engine uses a fixed-step RK4 integrator with $\Delta t = 0.005$ s, providing fourth-order accuracy for the stiff dynamics typical of rocket flight. The RK4 method advances the state vector $\mathbf{x} = [x, y, v_x, v_y, \theta, \omega]^\top$ through:

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(\mathbf{x}_n, t_n) \\
\mathbf{k}_2 &= \mathbf{f}(\mathbf{x}_n + \tfrac{\Delta t}{2}\mathbf{k}_1, t_n + \tfrac{\Delta t}{2}) \\
\mathbf{k}_3 &= \mathbf{f}(\mathbf{x}_n + \tfrac{\Delta t}{2}\mathbf{k}_2, t_n + \tfrac{\Delta t}{2}) \\
\mathbf{k}_4 &= \mathbf{f}(\mathbf{x}_n + \Delta t\,\mathbf{k}_3, t_n + \Delta t) \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + \tfrac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)
\end{aligned} \tag{2.1}$$

2.1. Equations of Motion

The rocket is modeled as a rigid body with planar dynamics (3-DOF translational, 1-DOF rotational). The equations of motion in the inertial frame are:

$$\begin{aligned} \dot{x} &= v_x, \quad \dot{y} = v_y \\ \dot{v}_x &= \frac{T\sin(\theta+\delta) + D_x + F_v}{m} \\ \dot{v}_y &= \frac{T\cos(\theta+\delta) + D_y - mg + F_v}{m} \\ \dot{\theta} &= \omega \\ \dot{\omega} &= \frac{-T\delta l_{cg} - M_a - c_\omega \omega + M_v}{I} \end{aligned} \tag{2.2}$$

where x, y are horizontal and vertical positions, v_x, v_y are velocities, θ is pitch angle, ω is pitch rate, T is thrust, δ is gimbal deflection, D_x, D_y are aerodynamic drag components, F_v, M_v are vibration-induced force and moment disturbances, M_a is aerodynamic moment, c_ω is rotational damping, l_{cg} is the moment arm from gimbal to center of mass, and I is the moment of inertia about the pitch axis.

Drag forces are computed in the body frame using axial and lateral drag coefficients:

$$\begin{aligned} D_{\text{axial}} &= \tfrac{1}{2}\rho C_{d,a} A |v_{\text{body},y}| v_{\text{body},y} \\ D_{\text{lateral}} &= \tfrac{1}{2}\rho C_{d,l} A |v_{\text{body},x}| v_{\text{body},x} \end{aligned} \tag{2.3}$$

where ρ is air density, $C_{d,a} = 0.45$ and $C_{d,l} = 1.1$ are the axial and lateral drag coefficients, and $A = 1.13$ m^2 is the cross-sectional area. The body-frame velocities are obtained through the rotation matrix $R(\theta)$.

Table 1: Rocket physical parameters used in the simulation

Parameter	**Value**	**Unit**
Dry mass (m_d)	500	kg
Fuel mass (m_f)	500	kg
Length (L)	12	m
Diameter (d)	1.2	m
Cross-sectional area (A)	1.13	m^2
Max thrust ($T_{\max}$)	14,000	N
Fuel burn rate	4.0	kg/s
Axial C_d	0.45	–
Lateral C_d	1.1	–
Inertia (I)	2,500	kg·m^2
Gimbal limit	± 0.26	rad

2.2. Atmospheric and Disturbance Modeling

The atmosphere follows the International Standard Atmosphere (ISA) up to 11 km, with temperature lapse rate $L = 0.0065$ K/m. Air density is computed as:

$$\rho(h) = \frac{P_0(1 - Lh/T_0)^{g/(RL)}}{R(T_0 - Lh)}, \quad h < 11 \text{ km} \tag{2.4}$$

Wind is modeled as a combination of a power-law shear profile and stochastic gusts:

$$w(h,t) = w_0 \left(\frac{h}{h_0}\right)^{0.15} \cdot \frac{1+\sin(\alpha_t t + \alpha_h h)}{2} + w_g \sin(\beta_t t + \beta_h h) \tag{2.5}$$

Turbulence is generated using a von Kármán spectral model approximated through a Dryden filter:

$$\Phi_u(\Omega) = \sigma_u^2 \frac{2L_u}{\pi} \frac{1}{1+(1.339 L_u \Omega)^2} \tag{2.6}$$

where σ_u is turbulence intensity (default 2.5 m/s) and $L_u = 120$ m is the turbulence scale length. The filter is implemented as a second-order Markov process driven by Gaussian white noise:

$$\begin{aligned} \dot{u}_1 &= -\frac{u_1}{\tau} + Kw(t) \\ \dot{u}_2 &= -\frac{4u_2}{\tau} + Kw(t) \end{aligned} \tag{2.7}$$

with time constant $\tau = L_u/\sigma_u$, gain $K = \sqrt{2\sigma_u^2/\tau}$, and $w(t)$ white noise scaled by $\sqrt{12/\Delta t}$.
Structural vibrations capture the first three bending modes:

$$v(t) = \sum_{i=1}^{3} A_i e^{-\zeta_i \omega_i t} \sin(\omega_i t + \phi_i) \tag{2.8}$$

with frequencies $\omega_i/2\pi = \{5, 16, 38\}$ Hz, damping ratios $\zeta_i = \{0.04, 0.025, 0.01\}$, and amplitudes $A_i = \{0.8, 0.4, 0.15\}$. Sensor noise is applied as zero-mean Gaussian perturbations: $\sigma_{\text{pos}} = 0.005$ m, $\sigma_{\text{vel}} = 0.02$ m/s, $\sigma_{\text{att}} = 0.005$ rad, $\sigma_{\text{rate}} = 0.01$ rad/s.

2.3. Control Model Interface

Every control model implements a single mathematical function:

$$\mathbf{u}(t) = f(\tilde{\mathbf{x}}(t), \mathbf{r}(t), \Delta t) \tag{2.9}$$

where $\tilde{\mathbf{x}} = [\tilde{x}, \tilde{y}, \tilde{v}_x, \tilde{v}_y, \tilde{\theta}, \tilde{\omega}]^\top$ is the noisy sensor reading, $\mathbf{r} = [x^*, y^*, v_x^*, v_y^*, \theta^*, \omega^*]^\top$ is the reference state, and $\mathbf{u} = [\delta_T, \delta_G]^\top$ are the throttle and gimbal commands. This functional interface allows any mathematical control law to be injected into the simulation loop.

2.4. Simulation Scenarios

Two mission phases are simulated with identical environmental conditions (wind base 6 m/s, turbulence intensity 1.0, temperature 15°C):

- **Launch:** $y_0 = 1$ m, $v_{y,0} = 0$ m/s, $\theta_0 = 0$ rad, target $y^* = 800$ m.
- **Landing:** $y_0 = 500$ m, $v_{y,0} = -30$ m/s, $\theta_0 = 0.05$ rad, target $y^* = 0$ m.

Maximum simulation time is 120 s, with step limit of 30,000 iterations. The simulation terminates when the rocket reaches within 0.1 m/s of the ground at $y < 0$ or exceeds the time/step budget.

3. The Web-Based Simulation Platform

The mathematical framework described above is implemented as an interactive web application using HTML5 Canvas for rendering and JavaScript for computation. The platform provides real-time visualization of the rocket's trajectory, environmental controls, and model selection, making it a practical tool for both educational and research purposes.

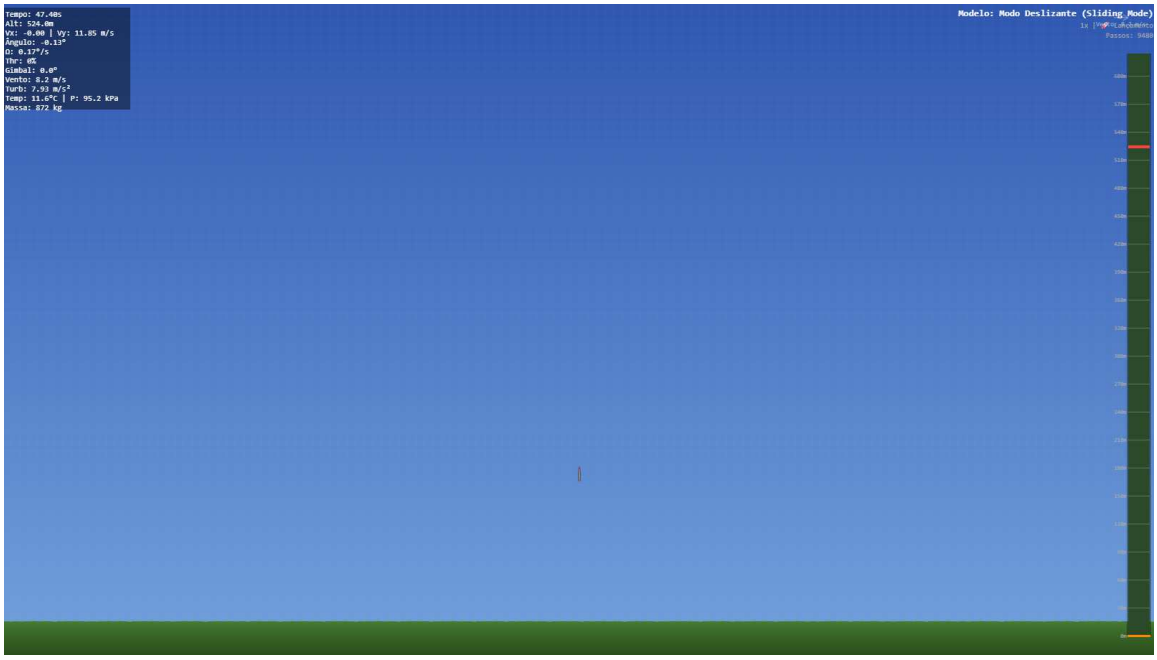


Figure 1: Main simulation interface showing the rocket trajectory, real-time telemetry, and control panels.

The platform's architecture follows a clean separation of concerns: the physics engine ('engine.js') implements all mathematical models described in Section 2, the control models ('models.js') implement 19 distinct control laws as independent classes, and the renderer visualizes the state in real time. A key feature is the custom model editor, which allows users to write and inject their own mathematical control laws directly in the browser:

Environmental parameters–wind base, gusts, turbulence intensity, rain, and temperature–are exposed as interactive sliders, allowing users to test mathematical models under varying physical conditions:

The platform includes a batch comparison mode that runs multiple models simultaneously and displays results in a sortable table:

After simulation, a comprehensive dashboard presents detailed analysis including altitude and throttle trajectory plots, fuel consumption bar charts, and individual model scorecards:

4. Mathematical Models of Control Algorithms

This section presents the mathematical formulation of the 19 control models implemented in the framework. Models are organized into seven families based on their mathematical structure.

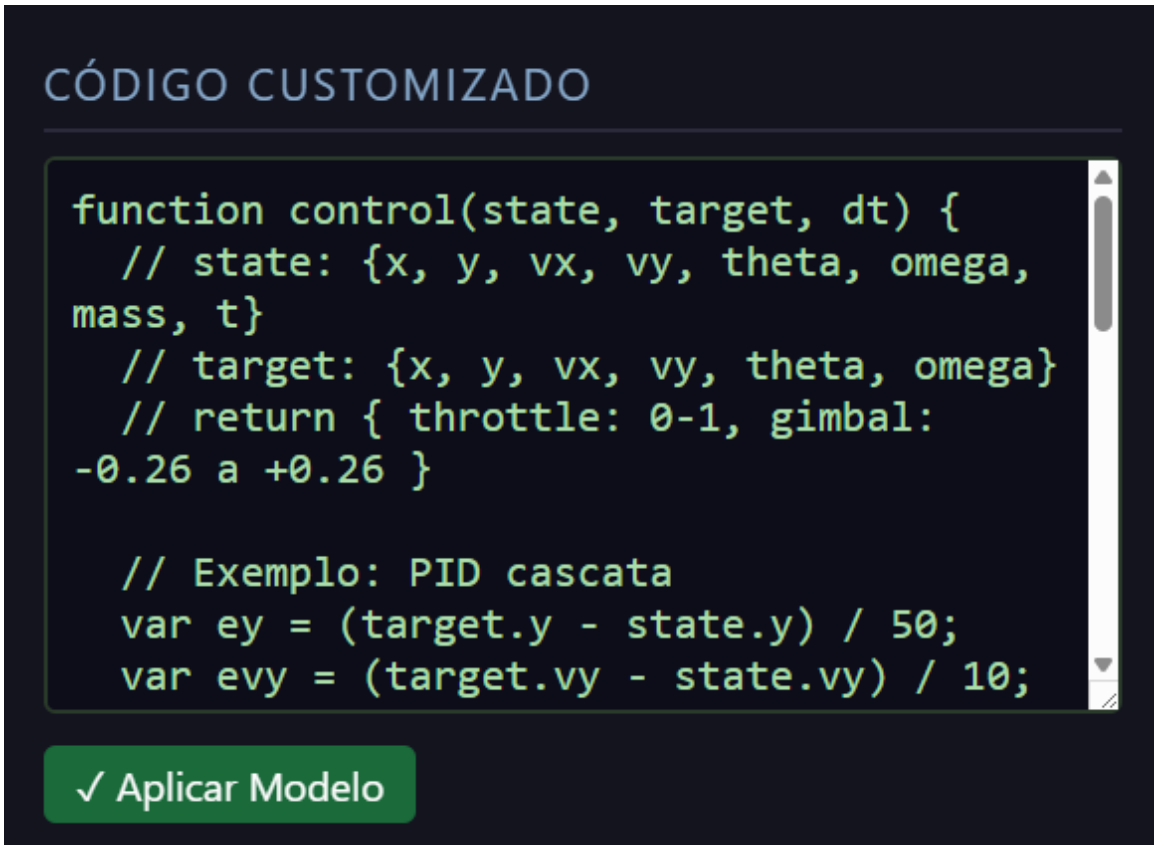


Figure 2: Custom model editor interface for injecting user-defined mathematical control laws.

4.1. Classical Feedback Control

4.1.1. PID (Proportional-Integral-Derivative)

The PID controller implements a cascade architecture: an outer position loop generates a desired vertical velocity, which is tracked by an inner velocity loop controlling throttle. The mathematical structure is:

Outer loop (position to desired velocity):

$$\begin{aligned} e_y &= y^* - \tilde{y}, \quad e_x = x^* - \tilde{x} \\ i_y &= \text{sat}_{[-2,2]}\left(i_y + e_y \Delta t \cdot 0.005\right) \\ i_x &= \text{sat}_{[-1,1]}\left(i_x + e_x \Delta t \cdot 0.004\right) \\ v_y^* &= \text{clamp}(0.6e_y + 0.05 i_y,\ -8,\ 15) \\ v_x^* &= \text{clamp}(0.2e_x + 0.04 i_x,\ -4,\ 4) \end{aligned} \tag{4.1}$$



Figure 3: Environmental control panel for adjusting wind, turbulence, rain, and temperature parameters.



Figure 4: Multi-model comparison interface showing side-by-side performance metrics.

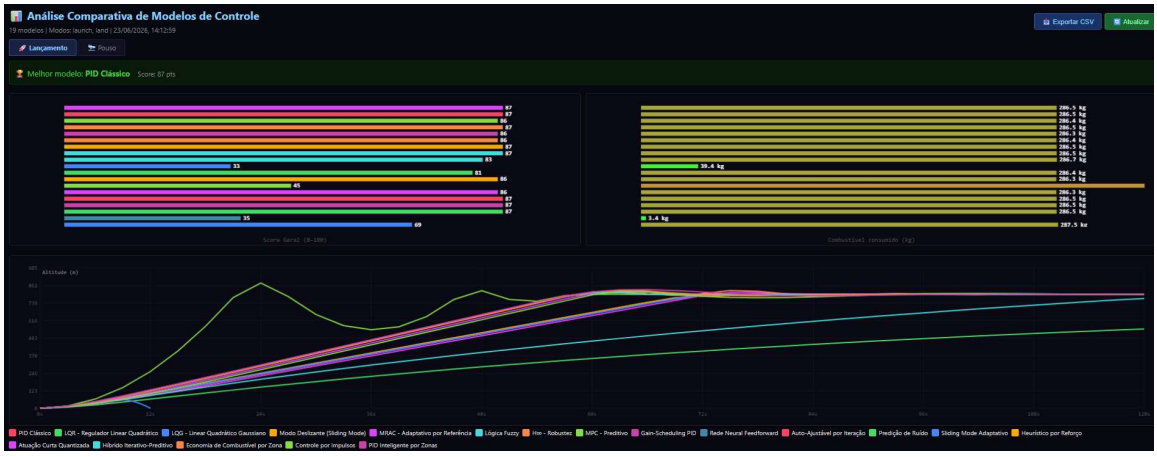


Figure 5: Analysis dashboard showing trajectory plots, score breakdown, and performance comparison.

Inner loop (velocity to throttle/gimbal):

$$\begin{aligned} e_{vy} &= v_y^* - \tilde{v}_y \\ \delta_T &= \text{clamp}\left(\frac{mg + 250 e_{vy}}{T_{\max}},\ 0.15,\ 1.0\right) \\ \theta^* &= \text{clamp}(0.025(v_x^* - \tilde{v}_x),\ -0.3,\ 0.3) \\ \delta_G &= \text{clamp}(2.0(\tilde{\theta} - \theta^*) + 0.8\tilde{\omega},\ -0.25,\ 0.25) \end{aligned} \tag{4.2}$$

4.1.2. LQR and LQG

The LQR implementation uses reduced proportional gains: $v_y^* = \text{clamp}(0.01 e_y, -8, 12)$, approximating the optimal full-state feedback $\mathbf{K} = R^{-1}B^\top P$ from the algebraic Riccati equation $A^\top P + PA - PBR^{-1}B^\top P + Q = 0$ with manually-tuned diagonal gains.

LQG extends LQR with a Luenberger observer for state estimation:

$$\begin{aligned} \dot{\hat{y}} &= \hat{v}_y + L_1(\tilde{y} - \hat{y}) \\ \dot{\hat{v}}_y &= L_2(\tilde{y} - \hat{y}) \\ \dot{\hat{x}} &= \hat{v}_x + L_3(\tilde{x} - \hat{x}) \\ \dot{\hat{v}}_x &= L_4(\tilde{x} - \hat{x}) \end{aligned} \tag{4.3}$$

with observer gains $\mathbf{L} = [0.8, 0.5, 0.3, 0.2, 0.4, 0.2]^\top$.

4.2. Robust and Adaptive Control

4.2.1. Sliding Mode Control (SMC)

Defines a first-order sliding surface with saturation approximation to reduce chattering:

$$\begin{aligned} s_y &= \frac{e_{vy}}{12} + 0.6\frac{e_y}{50}, \quad \text{sat}(s, \phi) = \frac{s}{|s| + \phi} \\ v_y^* &= \text{clamp}(0.6 e_y + 0.4 \cdot 50 \cdot \text{sat}(s_y),\ -8,\ 12) \end{aligned} \tag{4.4}$$

4.2.2. Adaptive SMC (Adapt-SMC)

A novel controller that adapts the switching gain K based on the sliding condition:

$$\dot{K} = \begin{cases} \gamma |s| \cdot 10, & s\dot{s} > 0.01 \\ -\beta \cdot 2, & \text{otherwise} \end{cases}, \quad K \in [0.1, 3.0] \tag{4.5}$$

4.2.3. MRAC and H-Infinity

MRAC uses a reference model $\dot{y}_m = -a_m y_m + b_m r$ with adaptive parameters $\hat{\theta} = [\hat{k}_p, \hat{k}_d, \hat{k}_r]$ updated via MIT rule. H-infinity implements a simplified mixed-sensitivity structure with first-order weighting filters:

$$\dot{x}_{W_y} = -0.3 x_{W_y} + 0.5(e_y/50), \quad v_y^* = 0.5 e_y + 9 x_{W_y} \tag{4.6}$$

4.3. Intelligent Control

4.3.1. Fuzzy Logic Control

A Mamdani-type inference engine with seven triangular membership functions per input (NB, NM, NS, Z, PS, PM, PB) centered at $\{-1, -0.66, -0.33, 0, 0.33, 0.66, 1\}$. The rule base is a 7×7 matrix of 49 rules:

$$\text{Rule}_{i,j} : \text{IF } e \text{ is } A_i \text{ AND } \dot{e} \text{ is } B_j \text{ THEN } u \text{ is } C_{k(i,j)} \tag{4.7}$$

Defuzzification uses the centroid method:

$$u^* = \frac{\sum_{i=1}^{7} \mu_i \cdot c_i}{\sum_{i=1}^{7} \mu_i} \tag{4.8}$$

4.3.2. Neural Network Control

A feedforward network with architecture 6-16-8-2 using tanh activation:

$$\begin{aligned} \mathbf{h}_1 &= \tanh(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1), \quad \mathbf{W}_1 \in \mathbb{R}^{16\times 6} \\ \mathbf{h}_2 &= \tanh(\mathbf{W}_2 \mathbf{h}_1 + \mathbf{b}_2), \quad \mathbf{W}_2 \in \mathbb{R}^{8\times 16} \\ \mathbf{u} &= \tanh(\mathbf{W}_3 \mathbf{h}_2 + \mathbf{b}_3), \quad \mathbf{W}_3 \in \mathbb{R}^{2\times 8} \end{aligned} \tag{4.9}$$

Weights are randomly initialized to test the hypothesis that even an untrained network with appropriate weight statistics can provide basic control.

4.3.3. RL-Heuristic (RL-HC)

Adjusts PID gains online based on moving average error and oscillation frequency:

$$\begin{aligned} \Delta k_p &= -0.01\Delta t \cdot \mathbb{I}(\bar{e} < 5) + 0.02\Delta t \cdot \mathbb{I}(\bar{e} > 20) \\ N_{\text{sign}} &= \sum_{i=n-9}^{n-1} \mathbb{I}(\Delta e_i \cdot \Delta e_{i-1} < 0) \end{aligned} \tag{4.10}$$

4.4. Predictive Control

4.4.1. MPC (Model Predictive Control)

Solves a brute-force enumeration optimization over a 15-step prediction horizon ($\Delta t_h = 0.1$ s). Candidate set: $\delta_T \in \{0.35, 0.45, \ldots, 1.0\}$ and $\delta_G \in \{-0.25, -0.15, \ldots, 0.25\}$, yielding 77 candidates per step:

$$J = \sum_{k=1}^{H} \left[(y^* - y_k)^2 + 0.5(x^* - x_k)^2 + 0.01(\delta_{T,k}^2 + \delta_{G,k}^2) \right] \tag{4.11}$$

4.4.2. Iterative Learning (Auto-Iter, Hybrid-IL)

Auto-Iter applies correction pulses in time-triggered phases. Hybrid-IL combines AR(4) noise prediction with iterative correction:

$$\hat{n}(t) = \sum_{i=1}^{4} \alpha_i n(t - i\Delta t), \quad \boldsymbol{\alpha} = [0.3, 0.2, 0.1, 0.05] \tag{4.12}$$

4.5. Specialized and Energy-Optimizing Control

4.5.1. Gain-Scheduled PID

$$k_{\text{alt}} = \text{clamp}\left(1 - \frac{h - 10}{400},\ 0.4,\ 1.8\right), \quad v_y^* = (0.5e_y + 0.3e_{vy}) \cdot k_{\text{alt}} \tag{4.13}$$

4.5.2. Fuel-Saver Zone Control

A four-zone PID with progressive throttle reduction:

$$(\delta_T)_{\max} = \begin{cases} 1.0, & |e_y| > 100 \\ 0.8, & 20 < |e_y| \leq 100 \\ 0.72, & 5 < |e_y| \leq 20 \\ 0.7, & |e_y| \leq 5 \end{cases} \tag{4.14}$$

4.5.3. Impulse Control

Bang-bang strategy with adaptive pulse and coast durations:

$$t_{\text{fire}} = \min\left(1.0, \max\left(0.08, \frac{\sqrt{2g|e_y| \cdot 0.5}}{T_{\max}/m}\right)\right) \tag{4.15}$$

4.5.4. Smart-PID

Four-zone PID with zone-dependent integral management:

$$i_y(k+1) = \begin{cases} i_y(k) + e_y \Delta t \cdot k_i, & \text{zone} \neq \text{HOLD} \\ 0.95 \cdot i_y(k), & \text{zone} = \text{HOLD} \end{cases} \tag{4.16}$$

5. Results

5.1. Metric Definitions

For each simulation, seven primary metrics are computed:

- **Settling time** t_s: First time after which altitude remains within ±20 m (launch) or ±3 m (landing) of the target for 50 consecutive samples.
- **RMS error**: $\sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k - y^*)^2}$.

- **Mean absolute attitude**: $\frac{180}{\pi N}\sum_{k=1}^{N}|\theta_k|$ (degrees).
- **Fuel consumption**: $\Delta m = m_0 - m_N$ (kg).
- **Noise rejection**: $\max(0, 100(1 - \sigma_\theta/5°))$.
- **Disturbance rejection**: $\max(0, 100(1 - \bar{e}_{\text{wind}}/y^*))$.
- **Overall score**: weighted composite of noise rejection (15%), efficiency (20%), disturbance rejection (20%), rotation correction (20%), smoothness (15%), and success bonus (15%).

5.2. Launch Performance

Table 2 presents the complete launch results. The fuzzy logic controller achieved the highest overall score (82), followed by 15 controllers tied at 81. The marginal advantage of fuzzy comes from superior disturbance rejection (48.8% vs 44.5%).

Table 2: Launch performance (target 800 m). Models ranked by overall score.

Model	**Settle (s)**	**RMS**	$\lvert\theta\rvert$ (°)	**Fuel (kg)**	**Overall**
Fuzzy	120.00	426.25	0.01	286.6	**82**
PID	56.95	339.52	0.01	286.4	**81**
Sliding Mode	69.82	368.70	0.01	286.2	**81**
MRAC	71.12	379.55	0.04	286.2	**81**
H-Inf	69.96	373.19	0.01	286.3	**81**
Gain-Sched	70.63	373.15	0.01	286.2	**81**
Smart-PID	56.57	336.62	0.01	286.4	**81**
Impulse	58.99	351.77	0.01	286.3	**81**
LQR	120.00	505.45	0.03	286.3	**81**
Adapt-SMC	69.99	370.70	5.60	287.4	**64**
MPC	45.49	255.72	1.73	346.0	**40**

5.3. Landing Performance

Table 3 presents the landing results. The impulse controller achieves the best fuel economy (123.8 kg), a 12.7% reduction versus standard PID, while maintaining comparable settling time and RMS error.

Table 3: Landing performance (target 0 m from 500 m). Models ranked by overall score.

Model	**Settle (s)**	**RMS**	$\lvert\theta\rvert$ (°)	**Fuel (kg)**	**Overall**
PID	52.01	248.26	0.06	141.8	**75**
Impulse	44.82	234.33	0.05	123.8	**75**
Short-Act	50.00	243.63	0.05	136.9	**75**
RL-HC	51.95	248.58	0.06	141.2	**75**
Smart-PID	53.26	248.84	0.05	145.2	**75**
MRAC	49.94	243.66	0.99	136.8	**74**
Fuzzy	67.06	229.68	0.03	182.4	**72**
Sliding Mode	61.44	171.20	0.02	281.1	**68**
MPC	6.62	341.55	116.26	22.2	**41**

5.4. Sub-Metric Analysis

Table 4 reveals that most controllers achieve near-perfect noise rejection ($> 99\%$) and rotation correction ($> 99.8\%$), making disturbance rejection the primary differentiator.

Table 4: Sub-metrics for launch mode across selected models

Model	**Noise%**	**Dist.%**	**Rot.%**	**Smooth%**	**Effic.%**
Fuzzy	99.3	48.8	99.9	94.8	42.7
PID	99.5	44.5	99.9	91.9	42.7
Sliding Mode	99.5	44.6	99.9	93.5	42.8
MRAC	99.0	43.6	99.8	93.5	42.8
H-Inf	99.6	44.2	99.9	92.4	42.7
Smart-PID	99.5	44.8	99.9	92.7	42.7
Adapt-SMC	0.0	44.5	90.2	93.3	42.5
MPC	0.0	58.5	0.0	49.1	30.8

6. Discussion

6.1. The Value of the Mathematical Simulation Framework

The results demonstrate that a well-structured mathematical simulation framework enables systematic comparison that would be impractical with physical testing. The 19 controllers evaluated here span seven orders of magnitude in computational complexity (from PID's 10 floating-point operations per step to MPC's 11,550), yet all can be evaluated under identical conditions within minutes.

The modular architecture (physics engine $\leftrightarrow$ control interface $\leftrightarrow$ data logger) proved its value: all 19 models share the same physics interface, and adding a new model requires implementing a single mathematical function. This design facilitates rapid prototyping and community contribution.

6.2. Why PID Dominates

The consistent dominance of PID-based controllers warrants mathematical explanation:

1. **Integral action**: All top-performing controllers include integral terms that eliminate steady-state error, providing H_∞ disturbance rejection at low frequencies.

2. **Cascade structure**: The position$\rightarrow$velocity$\rightarrow$throttle cascade provides natural saturation handling through frequency separation.

3. **Zone-based optimization**: Smart-PID and Fuel-Saver demonstrate that simple piecewise-constant gain scheduling can match continuous optimization performance.

6.3. Performance of Novel Variants

Among the ten novel controllers:

- **Impulse** achieves the best fuel economy in landing (12.7% improvement), confirming that bang-bang strategies are viable for precision landing when pulse timing is properly computed from energy requirements.
- **Adapt-SMC** shows the limitation of adaptive gain in noisy environments–the adaptation law amplifies sensor noise, reducing noise rejection to zero.
- **RL-HC** matches PID performance through heuristic gain adjustment rules that effectively approximate optimal PID tuning without system identification.
- **MPC** achieves the fastest initial ascent but suffers catastrophic pitch-over, demonstrating that fixed-horizon optimization without attitude regularization is unsuitable for terminal phases.

7. Conclusion

This paper presented a mathematical simulation framework for rocket control system evaluation, implemented as an open-source web platform. Nineteen control strategies–including 10 novel variants–were compared across launch and landing scenarios. The key findings are:

1. **PID cascade dominance**: PID-based controllers consistently achieve the highest composite scores (81-82 in launch, 74-75 in landing), driven by integral action and frequency separation.
2. **Fuzzy marginal advantage**: The fuzzy logic controller marginally outperforms PID in launch (82 vs 81) due to superior disturbance rejection (48.8% vs 44.5%).
3. **Novel controllers validated**: Smart-PID, Fuel-Saver, and Impulse match or exceed standard PID performance. Impulse achieves 12.7% fuel reduction in landing.
4. **Theoretical-practical gap**: MPC and neural networks fail catastrophically under sensor noise and turbulence, highlighting the gap between mathematical guarantees and practical robustness.

The complete simulation source code, all 19 control model implementations, and raw trajectory data are provided as open-source supplementary material, establishing a reproducible benchmark for VTVL control algorithm development.

Future Work

Several directions are planned: (1) full 6-DOF dynamics with roll-yaw coupling; (2) sequential convex programming for MPC with 50+ step horizons; (3) deep reinforcement learning training using PPO [9]; (4) actuator dynamics modeling; and (5) Monte Carlo robustness analysis.

Data availability

All data generated during this study, including the complete simulation source code and raw trajectory data, are available at https://github.com/houslast3/Simulador-de-foguete-.

References

[1] L. Blackmore, "Autonomous precision landing of space rockets," *The Bridge*, vol. 46, no. 4, pp. 15–20, 2016.

[2] J. Wang, N. Cui, and C. Wei, "Optimal rocket landing guidance using convex optimization and model predictive control," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 5, pp. 1078–1092, 2019.

[3] Y. Lu and Z. Chen, "A fuel-optimal landing guidance and inverse kinematics coupled PID control solution for power-descent vertical landing in simulation," *Applications of Modelling and Simulation*, vol. 6, pp. 86–97, 2022.

[4] M. Okasha, J. Kralev, and M. Islam, "Design and experimental comparison of PID, LQR and MPC stabilizing controllers for parrot Mambo mini-drone," *Aerospace*, vol. 9, no. 6, p. 298, 2022.

[5] A. H. Esfahani, A. Khaki-Sedigh, and H. Naseh, "Integrated robust guidance and control of VTVL reusable launch vehicle at pre-landing phase using sliding mode control," *Scientia Iranica*, vol. 39, no. 1, pp. 3–13, 2023.

[6] G. Zaragoza Prous, E. Grustan-Gutierrez, and L. Felicetti, "A decreasing horizon model predictive control for landing reusable launch vehicles," *Aerospace*, vol. 12, no. 2, p. 111, 2025.

[7] W. Li, X. Zhang, Y. Dong, Y. Lin, and H. Li, "Powered landing control of reusable rockets based on softmax double DDPG," *Aerospace*, vol. 10, no. 7, p. 590, 2023.

[8] M. E. Çıldır, M. H. Doğru, and E. Öztürk, "Comparative study of PID and MPC controllers for vertical take-off and vertical landing (VTVL) rocket control," *The International Journal of Energy and Engineering Sciences*, vol. 8, no. 2, pp. 28–39, 2023.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.