doi: 10.82124/CEAS-GNC-2026-077 CEAS-GNC-2026-077

FuzzyBuzz: Development of Fuzzy Logic and MPC Attitude Control Algorithms for Experimental Validation Onboard the ISS

Ana Laveron-Simavilla *Professor, Universidad Politécnica de Madrid, ETSI Aeronáutica y del Espacio, Madrid, Spain. ana.laveron@upm.es*

Victoria Lapuerta *Professor, Universidad Politécnica de Madrid, ETSI Aeronáutica y del Espacio, Madrid, Spain. mariavictoria.lapuerta@upm.es*

Karl Olfe *Researcher, Universidad Politécnica de Madrid, ETSI Aeronáutica y del Espacio, Madrid, Spain. ks.olfe@upm.es*

Richard Mark Haes-Ellis *Researcher, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, Seville, Spain. rhaesellis@us.es*

Ignacio Alvarado *Associate Professor, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, Seville, Spain. ialvarado@us.es*

Daniel Limon *Professor, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, Seville, Spain. dlm@us.es*

Francisco Gavilan *Associate Professor, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, Seville, Spain. fgavilan@us.es*

Rafael Vazquez *Professor, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, Seville, Spain. rvazquez1@us.es*

ABSTRACT

The advancement of autonomous spacecraft operations demands robust and efficient attitude control systems capable of meeting increasingly stringent pointing requirements while minimizing power consumption. While classical linear controllers such as PID and H_∞ have demonstrated flight heritage, emerging technologies including quantum communications and precision formation flying require superior performance that advanced nonlinear control strategies can provide. This paper presents the development and pre-flight validation of fuzzy logic and Model Predictive Control (MPC) algorithms designed for experimental testing aboard the International Space Station.

The paper details the experimental design using NASA's Astrobee free-flying robotic system as a validation platform for advanced attitude control algorithms. Comprehensive simulation results of fuzzy logic and MPC controllers are presented. The flight experiment protocol, operational procedures, and expected performance metrics are introduced.

The paper describes the methodology for on-orbit validation planned for the end of 2026, which aims to provide experimental data for these advanced control strategies in true microgravity conditions, advancing their Technology Readiness Level for future space mission applications.

Keywords: attitude control, fuzzy logic, model predictive control, spacecraft control, microgravity experimentation, ISS, Astrobee

1 Introduction

Spacecraft attitude control systems represent one of the most complex and resource-intensive subsystems of space vehicles. Historical developments have progressively improved pointing accuracy and stability, yet future applications demand further advances. Emerging technologies such as quantum communications and optical inter-satellite links require pointing accuracies approaching microradians—performance levels that challenge conventional control approaches [1, 2].

Current flight-proven control laws include classical PID controllers, Linear Quadratic Gaussian (LQG) methods, Linear Fractional Transformation (LFT) based techniques such as H_∞ synthesis, and sliding mode control [3]. While these approaches have demonstrated reliability across numerous missions, they exhibit inherent limitations when addressing the coupled nonlinear dynamics characteristic of spacecraft attitude motion under varying operational conditions.

Artificial intelligence-based controllers, particularly fuzzy logic systems, have shown theoretical advantages in handling uncertainty, modeling nonlinear dynamics, and providing robust performance under complex conditions [4, 5]. Simulation studies have demonstrated that fuzzy logic controllers can outperform traditional PID implementations in satellite attitude control, offering improved error-cost performance and reduced actuator saturation through efficient maneuver planning [6, 7]. However, these promising results have not been validated experimentally in space environments, limiting their adoption in operational missions despite their theoretical benefits.

Model Predictive Control has emerged as the preeminent methodology for real-time optimal control, demonstrating exceptional capabilities in constraint handling, stabilizing control design, tracking formulations, and robustness [8]. Recent computational advances have made MPC feasible for deployment on resource-constrained processors, significantly reducing implementation risks for onboard systems [9]. MPC has proven particularly effective for complex space missions, including spacecraft rendezvous with constraint and uncertainty handling [10], space debris removal operations [11], and underactuated spacecraft attitude control [12].

This paper presents the FuzzyBuzz experiment development for the International Space Station using NASA's Astrobee free-flying robotic platform. The paper is organized as follows. Section 2 describes the experimental platform and mission architecture. Sections 3 and 4 detail the design and optimization of the fuzzy logic and MPC attitude control algorithms, respectively. Section 5 presents the simulation-based tuning and validations of both controllers. Section 6 outlines the operational planning for the on-orbit test under true microgravity conditions. Section 7 explains how the planned experimental campaign is designed to advance these technologies from Technology Readiness Level 4 (validated in laboratory environment) to TRL 5-6 (validated in relevant environment), enabling their consideration for future operational spacecraft missions. Finally, section 8 sumarizes the conclusions.

2 Experimental Platform and Mission Design

Astrobee is a free-flying robotic platform developed by NASA to operate autonomously inside the International Space Station, providing a versatile testbed for guidance, navigation, and control algorithms in a real microgravity environment. Its modular architecture, precise propulsion system, and rich sensor suite enable high-fidelity experimentation on spacecraft-like dynamics without the constraints and risks associated with on-orbit satellites [13]. As a result, Astrobee offers a unique opportunity to validate attitude-control strategies, evaluate localization and motion-planning methods, and study the

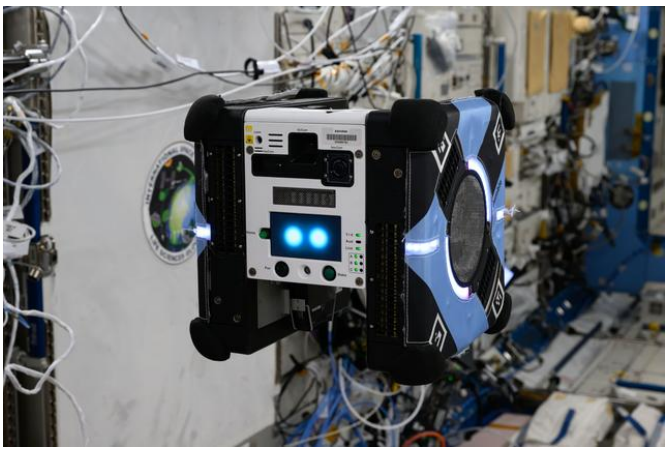

Fig. 1 One of the International Space Station's free-flying robots, Astrobee pictured by a NASA astronaut during Expedition 71. Courtesy of NASA.

behavior of control systems under realistic disturbances and operational conditions, thereby accelerating the development of advanced satellite technologies. The platform provides representative spacecraft dynamics while enabling software modifications and extensive telemetry collection capability.

The mission architecture for validating attitude control algorithms using Astrobee aboard the International Space Station (ISS) is structured around a tightly integrated set of robotic, computational, and environmental components similar to [14]. In this case, its core is the Astrobee platform.

The experimental payload consists of software modules implementing the attitude control algorithms. These are deployed alongside an attitude trajectory planner that generates reference orientation profiles, and a fault detection and isolation system that ensures operational safety. The experiment unfolds within the constrained microgravity environment of the ISS cabin, where Astrobee operates in a bounded volume defined by crew safety protocols and hardware layout. Environmental disturbances, such as residual airflow or sensor noise, are treated as natural perturbations to test controller robustness.

The mission proceeds through distinct phases. Following autonomous undocking, Astrobee initializes the control payload and begins executing a sequence of attitude maneuvers, including pitch, yaw, and roll rotations. Throughout the experiment, Astrobee logs orientation states, angular velocities, actuator commands, and sensor data, which are later downlinked for ground-based analysis.

Safety is ensured through onboard collision avoidance routines and supervisory autonomy that monitors system health and enforces ISS operational constraints. Upon completion, Astrobee returns to its docking station for recharging and data offload. The collected data enables quantitative evaluation of controllers performance across multiple metrics.

3 Fuzzy Logic Control Design and Optimization

This section presents the fuzzy controllers designed for Astrobee platform. The methodology employs multi-objective optimization to generate Pareto fronts characterizing the fundamental tradeoff between pointing accuracy and power consumption.

3.1 Fuzzy Controller

To design a fuzzy controller, it is first necessary to define its key components, including fuzzy rules, membership functions, and defuzzification methods, which are discussed in the following sections. Three different controllers are designed, one per axis X, Y, or Z. The structure is the same for all three. The attitude is parametrized using quaternions and for each axis, the error quaternion, q_e is calculated fromt

the actual attitude, q, and the desired one, q_d, as

$$q_e = q_d^{-1} \otimes q \tag{1}$$

where q_d^{-1} denotes the inverse of quaternion q_d, which for unit quaternions corresponds to its conjugate, and $\otimes$ denotes the quaternion product. The controller input, $\mathbf{q_e} = \{e_x, e_y, e_z\}$ is the vector part of the quaternion error, $q_e = [q_{e0}, \mathbf{q_e}]$, where q_{e0} denotes the scalar part.

3.1.1 Membership functions

Membership functions were implemented and adjusted to achieve the desired controller response, which includes maintaining stationary states for extended periods and enabling rapid transitions between states with inverse output responses. Trapezoidal membership functions are suitable for this purpose because their horizontal sections promote stable stationary states, while right-trapezoidal shapes allow for abrupt output changes. In addition, they are relatively easy to parameterize.

In the stabilization area (near the commanded attitude), combining trapezoidal functions with triangular functions is advisable due to the parametric simplicity of triangular functions. Although Gaussian functions require a similar number of parameters, their response is less sensitive to small input variations than that of triangular functions and involves a higher computational cost.

Therefore, trapezoidal functions are used to maintain stationary states, while combinations of trapezoidal-triangular or triangular-triangular functions are employed in intervals where the controller requires slight modulation of actuation. The input membership functions are shown in Figure 2. The input variables are the error (e_x, e_y, e_z), which is the difference between the satellite's current and desired orientations, and the error derivative ($\dot{e_x}$, $\dot{e_y}$, $\dot{e_z}$) in each axis. The linguistic terms for the linguistic variables are: Negative Big (NB), Negative (N), Negative Small (NS), Zero (Z), Positive Small (PS), Positive (P), and Positive Big (PB).

Additionally, the following constraints have been imposed for the definition of the input membership functions:

(i) Symmetry with respect to the origin (for all functions).

(ii) As a general rule, a 50% overlap between adjacent membership functions to ensure a unitary membership value. Some adjacent functions do not overlap; specifically, those corresponding to P and PB and their symmetric counterparts N and NP share the boundary point due to the need for fast response transitions. To address this practical issue, the overlap has been centered at the shared point, and the smallest numerically representable interval has been established as the transition zone, making it by definition a scalene trapezoid, although in practice it is a right trapezoid (since the probability of falling into the transition zone, which is infinitesimally small, is practically zero).

As Figure 2 shows, six parameters define the membership function shapes:

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6]$$

where:

- x_1: distance from the origin to the end of the Z domain of the error.
- x_2: distance from the origin to the end of the PS domain of the error.
- x_3: distance from the origin to the end of the P domain of the error.
- x_4: distance from the origin to the end of the Z domain of the error derivative.
- x_5: distance from the origin to the end of the PS domain of the error derivative.
- x_6: distance from the origin to the end of the P domain of the error derivative.

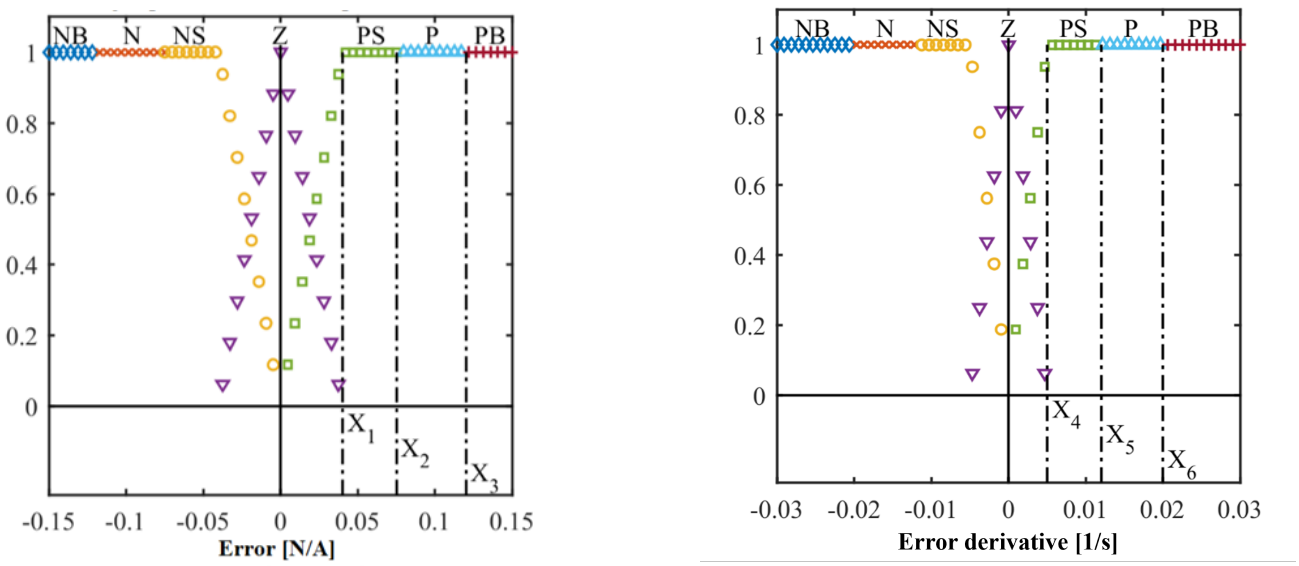

Fig. 2 Membership functions.

Table 1 Fuzzy rules for the X, Y, and Z axes.

Error (e_x, e_y, e_z)							
Derivative ($\dot{e_x}$, $\dot{e_y}$, $\dot{e_z}$)	NB	N	NS	Z	PS	P	PB
NB	PB	PB	PB	PB	PB	PB	Z
N	PB	PB	PB	PB	PB	Z	NB
NS	PB	PB	PB	PS	Z	NB	NB
Z	PB	PB	PS	Z	NS	NB	NB
PS	PB	PB	Z	NS	NB	NB	NB
P	PB	Z	NB	NB	NB	NB	NB
PB	Z	NB	NB	NB	NB	NB	NB

These six parameters per axis fully define the fuzzy controller's input membership functions, so a total of eighteen parameters are needed to fully define the input memberships.

3.1.2 Fuzzy Rules

The fuzzy rules for the X, Y, and Z axes share the same structure, as described in Table 1. These rules define the values of the output linguistic variables (T_{cx}, T_{cy},T_{cz}), which correspond to the actuation needed per axis, for each combination (AND operation) of the input variables namely, the error and the error derivative in that axis. For example:

$$\begin{aligned} &\text{if } (e_x = \text{NS}) \text{ and } (\dot{e_x} = \text{PB}) \text{ then } (T_{cx} = \text{NB}) \\ &\text{if } (e_y = \text{NS}) \text{ and } (\dot{e_y} = \text{Z}) \text{ then } (T_{cy} = \text{PS}) \\ &\text{if } (e_z = \text{PS}) \text{ and } (\dot{e_z} = \text{NB}) \text{ then } (T_{cz} = \text{PB}) \end{aligned} \tag{2}$$

The two-dimensional fuzzy input domain is divided into two distinct regions: the maneuver area (where both inputs are relatively large) and the stabilization area (where both inputs are relatively small). This division enables the controller to achieve the desired control law profile.

3.1.3 Defuzzification

The Takagi-Sugeno defuzzification method [16] is employed for the fuzzy controllers along all three axes. This method uses a list of discrete values representing fractions of the maximum actuation. These discrete values, known as singletons, are presented in Table 2.

Table 2 Singleton values for the output.

Output	NB	NS	Z	PS	PB
Actuation (T_{cx}, T_{cy}, T_{cz})	-1	-0.5	0	0.5	1

3.2 Optimization

In this section the optimization procedure incorporating dynamic simulation of Astrobee's attitude dynamics, actuator characteristics, and environmental disturbances is summarized. A Multiple Objective Particle Swarm Optimization (MOPSO) [7] has been used to minimize two objectives: referred to as "error" and "cost". The eighteen parameters defining the input membership functions were varied to minimize these objectives.

The optimization process is divided into two main steps. The first step aims to explore the population domain as broadly as possible, considering the constraints of time and computational resources. To achieve this, a wide range was chosen for the selection of populations. The second step refines the initial search and validates the results of the first optimization by selecting a narrower range based on its results. The final Pareto fronts are derived from the second optimization.

The same reference maneuver is used for the optimization in all three axes, see Figure 3. A 20-degree step command is applied to the axis being optimized 5 seconds after the simulation begins, and the simulation ends after 30 seconds.

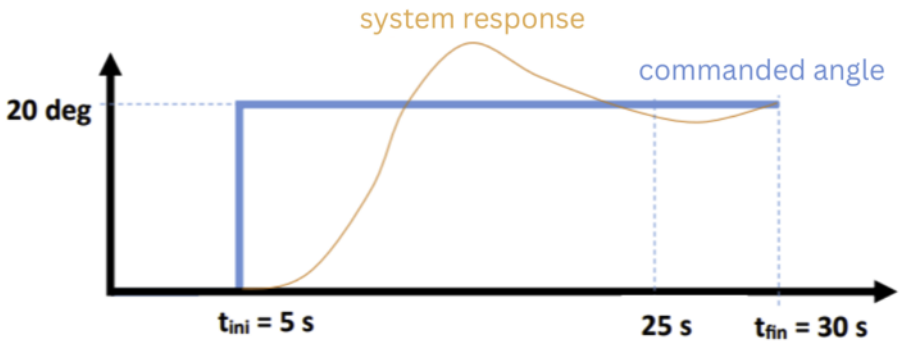

Fig. 3 The reference maneuver

The error metric is defined as the integral of the absolute value of the vector component of the error quaternion on that axis. This integration is performed during the maneuver phase (excluding the initial 5 seconds with no command), and the result is normalized with respect to a reference value:

$$\text{Error} = \frac{\int_{t_{\text{ini}}}^{t_{\text{fin}}} |e_{\text{axis}}(t)|\, dt}{\text{Error reference value}} \tag{3}$$

The cost represents the control effort, calculated from the absolute value of the angular acceleration on the optimized axis. The total metric is obtained as the integral of this quantity:

$$\text{Cost} = \frac{\int_{t_{\text{ini}}}^{t_{\text{fin}}} |\alpha(t)|\, dt}{\text{Cost reference value}} \tag{4}$$

4 Model Predictive Control Formulation

This section details the MPC formulation for spacecraft attitude control, including the prediction model based on quaternion kinematics and Euler's rotational dynamics, constraint specification incorporating actuator limitations and pointing requirements.

Computational implementation of MPC enabling its real-time execution on Astrobee's embedded processors is also described. The optimization algorithm and its code generation are developed by means of the SPCIES tool for embedded MPC [15]. Performance analysis demonstrating feasibility of MPC for resource-constrained space applications are also provided.

4.1 MPC

Model Predictive Control (MPC) can be seen as a practical implementation of an optimal control policy that computes the control input by repeatedly solving a finite-horizon optimization built on a prediction model of the plant. At each sample, the model forecasts the state trajectory and the optimizer selects inputs that minimize a given (typically quadratic) cost function while enforcing state and input constraints in a multivariable setting [18]. In what follows, we adopt the attitude model and conventions of NASA's *Astrobee Guest Science Guide* [17], derive the discrete prediction model used by MPC, present a standard MPC formulation, and describe the embedded implementation based on the SPCIES toolbox.

4.2 Quaternion attitude prediction model and linearization

We represent attitude with unit quaternions $q = \begin{bmatrix} q_0 & \mathbf{q}^\top \end{bmatrix}^\top \in \mathbb{S}^3$, a singularity-free parameterization of SO(3) that composes via the Hamilton product. Considering the error quaternion $q_e = q_d^{-1} \otimes q$, for small errors $\delta\theta \approx 2\,\text{vec}(q_e)$, and the state $x = \begin{bmatrix} \delta\theta^\top & \boldsymbol{\omega}^\top \end{bmatrix}^\top$ yields a controllable linear model suitable for MPC [19]

The kinematics and dynamics description is given by:

$$\dot{q} = \tfrac{1}{2}\,\Omega(\boldsymbol{\omega})\, q, \quad \Omega(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\boldsymbol{\omega}^\top \\ \boldsymbol{\omega} & -[\boldsymbol{\omega}]_\times \end{bmatrix}, \tag{5}$$

$$\mathbf{J}\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) = \boldsymbol{\tau} + \mathbf{d}. \tag{6}$$

Linearizing (5)–(6) at $q_e = [1, \mathbf{0}],\ \boldsymbol{\omega} = \mathbf{0}$ gives

$$\dot{x} = A_c x + B_c u, \quad A_c = \begin{bmatrix} \mathbf{0} & \frac{1}{2}\mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad B_c = \begin{bmatrix} \mathbf{0} \\ \mathbf{J}^{-1} \end{bmatrix},\ u = \boldsymbol{\tau}. \tag{7}$$

With sample time T_s and zero-order hold,

$$x_{k+1} = Ax_k + Bu_k, \qquad A = e^{A_c T_s}, \quad B = \int_0^{T_s} e^{A_c \tau} B_c \, \mathrm{d}\tau. \tag{8}$$

Model (8) is adopted as the MPC predictor. The model is completed defining the constraints on the variables. We consider box constraints on the states and inputs, given by:

$$x_{\min} \leq x_k \leq x_{\max}, \quad u_{\min} \leq u_k \leq u_{\max}$$

These constraints allow us to encode actuator saturation limits and bounds on angular velocity, preventing excessively fast body rotations that could violate the validity region of the local reduced-quaternion model, degrade vision-based localization, and reduce safety margins within the confined ISS environment.

4.3 Stabilizing MPC Formulation

MPC is a family of control formulations that can enjoy nice theoretical properties, such as constraint satisfaction, recursive feasibility and asymptotic stability. The most extended stabilizing formulation of MPC is the one that incorporates terminal ingredients, that is, a terminal cost, penalizing the terminal state, and a terminal constraint forcing the terminal state to reach a suitable neiborhood of the target. If they are appropriately chosen, the resulting predictive control law stabilizes the plant while satisfying the constraints.

In this paper we adopt the so-called lax-MPC formulation that considers a quadratic stage cost function and a quadratic terminal cost, without terminal constraint [9]. The control law is derived from the solution of the following optimization problem:

$$\min_{\{x_i,u_i\}} \sum_{i=0}^{N-1} \left(\|x_i - x_r\|_Q^2 + \|u_i - u_r\|_R^2 \right) + \|x_N - x_r\|_T^2 \tag{9}$$

$$\text{s.t.} \quad x_{i+1} = Ax_i + Bu_i, \quad i = 0, \ldots, N-1, \tag{10}$$

$$x_{\min} \leq x_i \leq x_{\max}, \quad i = 1, \ldots, N-1, \tag{11}$$

$$u_{\min} \leq u_i \leq u_{\max}, \quad i = 0, \ldots, N-1, \tag{12}$$

$$x_0 = x_p(k). \tag{13}$$

where $x_p(k)$ is the measured/estimated state at time k and $(x_r(k), u_r(k))$ the reference pair.

This controller has proved to be asymptotically stabilizing if the terminal weighting cost is chosen as the solution of the discrete-time LQR Riccati equation, since this makes the optimal cost function to serve as Lyapunov function for the controlled system.

The prediction horizon N is typically chosen as large as possible, since a larger N makes the domain of attraction larger and the MPC control law closer to the infinite-horizon optimal control policy. However, the computational cost is increased for larger prediction horizons, making necessary to find a trade-off between performance and computational time.

Furthermore, the resulting MPC control law provides inherent robustness to moderate model mismatch and bounded disturbances (local ISS), even though the predictor is linear and the plant is nonlinear.

4.4 Real-time implementation in embedded system using SPCIES

While MPC enjoys remarkable properties with a theoretical background, its implementation in real time is challenging, since it requires the solution of the optimization problem (9) in real time. This is particularly relevant when the control platform is an embedded system with limited computational resources.

To meet real–time requirements, a taylored solver that exploits the structure of the optimization problem must be used. Besides, the solver must take into account, not only the computational capability

of the control platform, but also the available memory resources, which results to be typically the bottleneck of the implementation.

In this paper, we use the tool SPCIES (Suite for Predictive Controllers in Industrial Embedded Systems), which is a code–generation toolbox that produces MPC solvers based on First-Order Methods that exploit the structure of the condensed QP arising from linear MPC [9]. Rather than relying on general–purpose QP engines, SPCIES tailors the numerical linear algebra to the block–banded KKT system generating library-free solvers suitable for real–time execution on PLCs, MCUs, and low–power CPUs.

A key aspect of SPCIES' solvers is the *pre–factored banded system*: by exploiting the block–banded KKT structure and using a banded Cholesky factorization, the memory footprint grows linearly with the prediction horizon and the per–iteration cost remains $O(N)$. The generated code is *portable C*, stand–alone and free of dynamic allocation, with clear APIs that simplify integration and retuning.

This tool matches our requirements of strict constraint satisfaction, predictable timing on resource–limited hardware, and easy retuning of (Q, R, T, N) without reauthoring solver code. The condensed, banded approach keeps iteration cost linear in N and enables stable performance at the prescribed sampling frequency for Astrobee (10 Hz).

To this aim, the MPC optimization problem is recasted as a quadratic program of the form

$$\min_{z} \ \frac{1}{2} z^\top H z + q^\top z \quad \text{s.t.} \quad z \in \mathcal{Z}, \ \ Gz = b,$$

where $z \in \mathbb{R}^{n_z}$ is the primal decision vector with n_z decision variables, $G \in \mathbb{R}^{m_z \times n_z}$ is the equality-constraint matrix associated with m_z equality constraints, $b \in \mathbb{R}^{m_z}$, H is the Hessian matrix, q is the linear cost term, and $\mathcal{Z}$ is a box-constrained set.

For the solver, we use the dual FISTA variant (See Algorithm 1) with tolerance ε and a fixed iteration cap. At each iteration, the algorithm requires the solution of the following ancillary problems:

$$z(\lambda) \ = \ \arg\min_{z \in \mathcal{Z}} \frac{1}{2} z^\top H z + q^\top z - \lambda^\top (Gz - b) \tag{14}$$

$$\Delta\lambda(z) \ = \ \{\Delta\lambda : GH^{-1}G^\top \Delta\lambda = -\big(Gz - b\big)\} \tag{15}$$

Notice that, because H is diagonal and $\mathcal{Z}$ is box-constrained, $z(\lambda)$ can be obtained exactly by component-wise minimization followed by interval clipping. Besides, $\Delta\lambda$ can be efficiently solved exploiting the block-banded structure of the equations [9].

Algorithm 1 Dual FISTA with banded linear solves

Require: initial dual λ_0, tolerance ε

$k \leftarrow 0, \eta_0 \leftarrow \lambda_0, t_0 \leftarrow 1$

repeat

  $k \leftarrow k + 1$

  Calculate $z_{k-1} \leftarrow z(\lambda_{k-1})$ from (14).

  Calculate $\Delta\lambda_{k-1} \leftarrow \Delta\lambda(z_{k-1})$ from (15).

  $\eta_k \leftarrow \lambda_{k-1} + \Delta\lambda_{k-1}$

  $t_k \leftarrow \frac{1}{2}\left(1 + \sqrt{1 + 4t_{k-1}^2}\right)$

  $\lambda_k \leftarrow \eta_k + \dfrac{t_{k-1} - 1}{t_k}(\eta_k - \eta_{k-1})$

  residual $\Gamma \leftarrow G\, z(\lambda_k) - b$

until $\|\Gamma\| \leq \varepsilon$

return $z^\star = z(\lambda_k)$ and apply $u_k = u_0^\star$

5 Simulation Results and Performance Predictions

In this section, the fuzzy controller and the MPC are tested and compared for a model of the Astrobee under the same reference maneuver. Both controllers are designed considering the same performance indexes under the practical constraint that the resulting control law must be implementable in Astrobees' platform.

5.1 Fuzzy controller results

Figure 4 shows the Pareto front obtained from the MOPSO optimization for roll axis. Similar results are obtained for the other axes. The most interesting aspect extracted from this figure is the wide range of configuration possibilities for cost and error objectives, which means that the fuzzy controller is very adaptable and can be used in different scenarios when the cost-accuracy directive is changed.

For experimental protocol, three points will be selected from the Pareto fronts (minimum-cost, minimum-error, and central point) and compared, see Figure 4. The central point is defined as the point with an intermediate error and cost on the Pareto front.

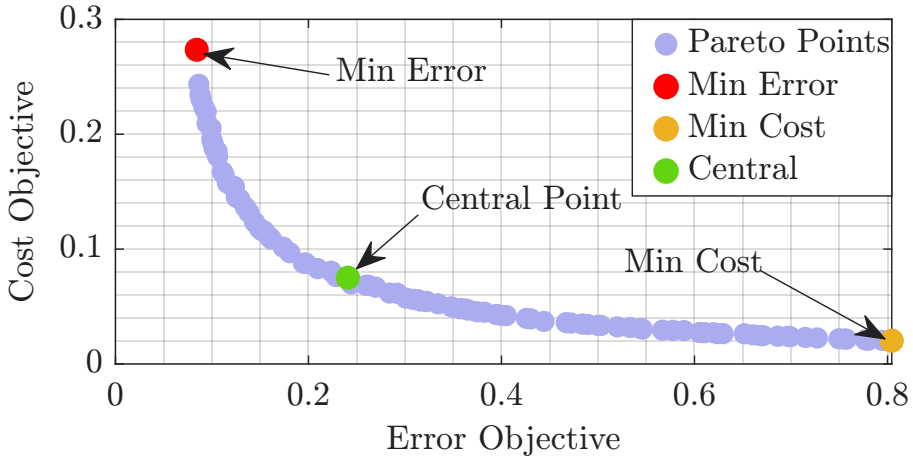

Fig. 4 Pareto front of the fuzzy controller for the roll axis.

Figure 5 shows the response to a 20-degree step maneuver for the central point. The simulation was carried out for all axes. The response is very smooth, slightly faster on Roll axis, reaching the objective

in less than 10 seconds. No overshooting is observed, which is a typical problem in linear controllers, which implies lower energy consumption.

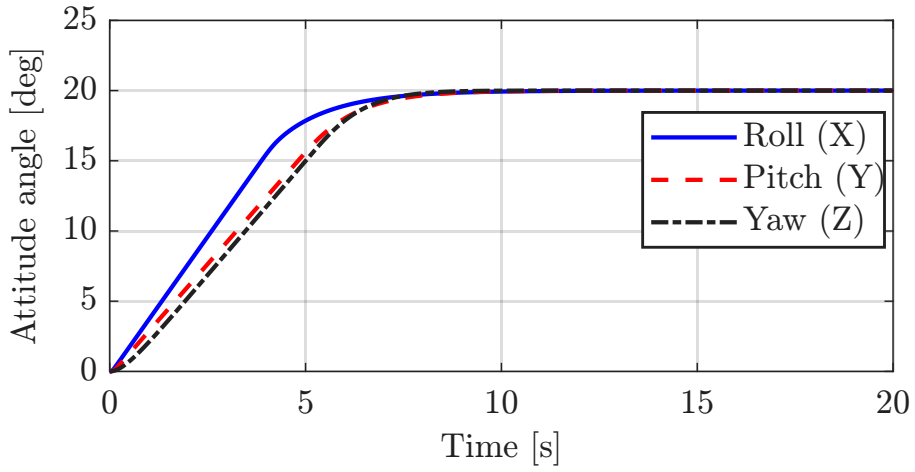

Fig. 5 Angle evolution of the Central Point fuzzy controller during a 20° maneuver about the three axes.

5.2 MPC tuning and results

Since there is no prevalence among the angles of the quaternions we select $Q = \mathbf{I}$. Considering also the same cost for every torque, we choose $R = \rho\,\mathbf{I}$, where $\rho \in [0, \infty)$ that plays the role of design parameter. Larger ρ penalizes control effort more strongly and then, smoother, slower responses; smaller ρ prioritizes tracking yielding to faster and more aggressive responses. The terminal weight T is chosen as the discrete LQR solution for (A, B, Q, R).

For each ρ on a logarithmic grid (e.g., $\rho \in [10^1, 10^2]$), we simulate the commanded maneuvers and compute the two metrics defined earlier : Error (3) (J_1) and Cost (4) (J_2). We select $\rho^\star$ that jointly balances both objectives using the minimum normalized sum or, equivalently, the knee point on the Pareto curve (J_1, J_2). Figure 6 illustrates a typical sweep.

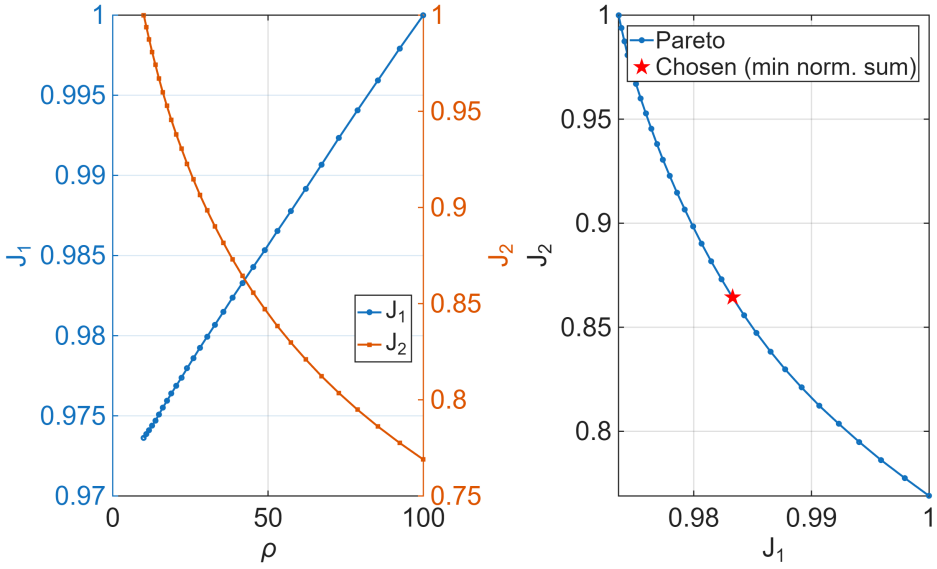

Fig. 6 Pareto front for MPC w.r.t the parameter ρ: trade–off between Error (J_1) and Cost (J_2).

We adopt a prediction horizon $N = 12$, suitable to achieve a computation time within the sampling period $T_s = 0.1$ s (10 Hz). In desktop tests on an Intel Core Ultra 9, the total solve time for representative settings was on the order of ~ 0.025 ms, i.e., $\sim 0.025\%$ of the 100 ms budget at 10 Hz (per–iteration times are measured directly and the cap gives a hard upper bound). Even under a conservative desktop$\rightarrow$ flight scaling of 20×–50× to account for Astrobee's smartphone–class ARM CPUs and contention with onboard processes, the solve time remains $\lesssim 1.3$ ms per tick (from ~ 0.025 ms on the Ultra 9). This is well within a 100 ms budget at 10 Hz, leaving ample headroom for sensor/actuator I/O and estimation. The worst case execution time stays bounded, so the selected (N, T_s) meet real-time margins on Astrobee while preserving the closed-loop performance achieved in simulation.

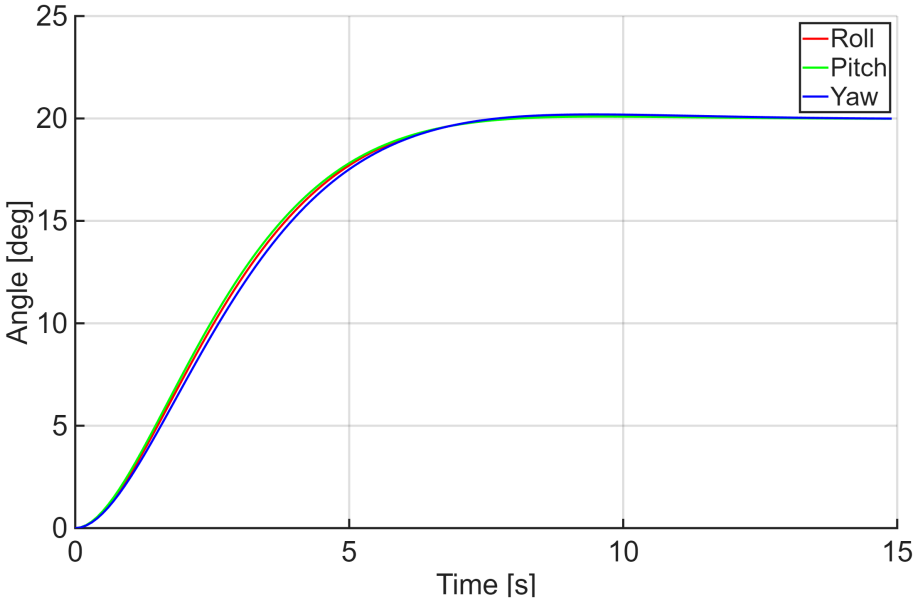

Fig. 7 Angle evolution for MPC at 20º.

Figure 7 shows a step angle on each axis of 20º, we can see that the response is fairly smooth and fast. In this case the control inputs do not saturate to their limits. In order to validate the MPC controller for maneuver with active constraints, we also show in figure 8 a step of 90º. In this case we can see the motion is very smooth and yet the thrust constraints are satisfied.

6 Flight Operations Planning and Expected Outcomes

The experiment is planned for execution aboard the International Space Station in 2027, following a comprehensive validation campaign throughout the preceding year. During this preparation phase, the software components of the guidance, navigation, and control pipeline will undergo rigorous testing both within the official Astrobee simulator and in laboratory facilities. These ground-based activities will include integration tests, hardware-in-the-loop evaluations, and motion experiments using air-carriage platforms that replicate frictionless dynamics. Flight readiness preparations also include safety certification processes, crew procedure development, and mission integration planning. The operational concept will minimize crew time requirements through autonomous execution with ground monitoring.

The experimental protocol encompasses a number of different test runs organized into six priority levels, systematically evaluating controller performance across varying operational conditions. The experimental design includes single-axis and multi-axis maneuvers, parametric robustness testing, and mission-representative scenarios such as geodetic pointing, pointing vector alignment for large-angle

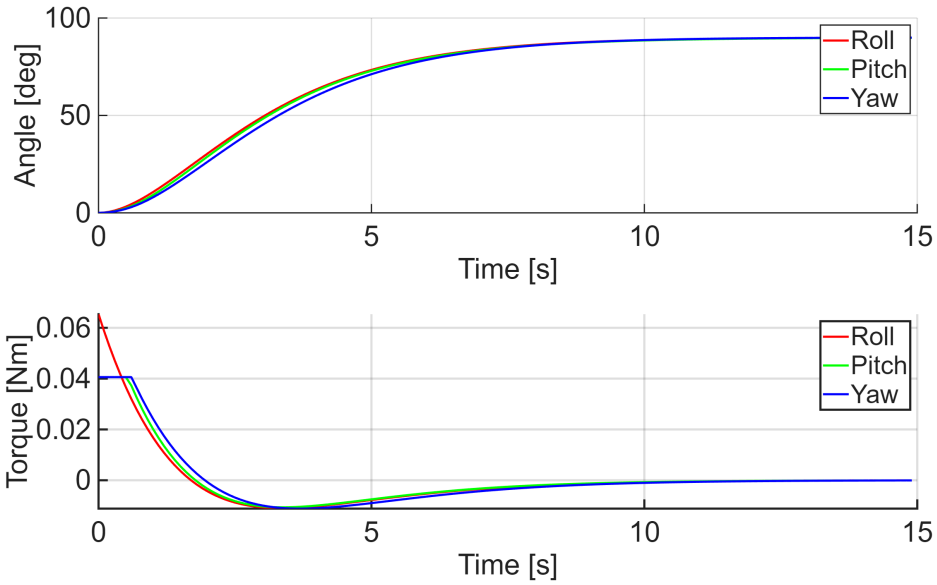

Fig. 8 Angle evolution for MPC at 90º with constraints saturated.

slews, and sinusoidal tracking for bandwidth characterization. The prioritization scheme ensures collection of critical comparison data within mission constraints.

During the maneuvers, high-frequency telemetry including quaternion states, angular velocities, actuator commands, and onboard sensor data are recorded for post-flight analysis. The experiment is conducted autonomously using Astrobee's onboard planner and safety monitors. Collision avoidance and fault detection routines ensure safe operation within the ISS cabin.

Expected outcomes from the experimental campaign include quantitative validation or refinement of simulation-based performance predictions, identification of discrepancies between ground testing and microgravity behavior, and demonstration of real-time MPC feasibility on space-qualified embedded processors. The flight data will provide an experimental assessment of fuzzy logic and MPC attitude control in relevant operational environment.

7 Technology Maturation Path

The planned experimental campaign represents a critical step in maturing fuzzy logic and MPC attitude control from laboratory validation to operational readiness. The paper aims to discuss how flight demonstration addresses key risk reduction objectives for future mission adoption, including performance validation in representative environment, computational feasibility confirmation, and robustness characterization.

The real-time MPC implementation on resource-constrained embedded processors addresses concerns about computational feasibility that have historically limited MPC adoption in space applications. Successful flight demonstration will help position MPC as a viable technology for next-generation spacecraft requiring autonomous operation in complex scenarios.

8 Conclusions

This paper presents the development and pre-flight validation of fuzzy logic and Model Predictive Control algorithms for spacecraft attitude control, with experimental testing planned aboard the International Space Station using NASA's Astrobee platform. Initial simulation results suggest measur-

able improvements over traditional methods, particularly for applications requiring tight accuracy with minimal power consumption or operation near actuator limits.

These planned experiments contribute to the broader objectives of autonomous spacecraft operations and precision control capabilities essential for future space exploration and applications. Results from this campaign will inform the space community's understanding of when and how advanced control algorithms provide practical advantages over established approaches.

Declaration of Use of Artificial Intelligence

Artificial intelligence was used to assist in organizing the structure and refining the English presentation of this article.

References

[1] M. Milaševičius & L. Mačiulis (2024). "A Review of Mechanical Fine-Pointing Actuators for Free-Space Optical Communication". *Aerospace*, 11(1), 5. doi: 10.3390/aerospace11010005.

[2] G.-N. Kim, S.-Y. Park *et al.* (2022). "Design of Novel Laser Crosslink Systems Using Nanosatellites in Formation Flying: The VISION". *Aerospace*, 9(8), 423. doi: 10.3390/aerospace9080423.

[3] B. Wie, *Space Vehicle Dynamics and Control*, 2nd ed. Reston, VA: AIAA, 2008.

[4] B. Jiang, Z. Gao, P. Shi, and Y. Xu, "Adaptive Fault-Tolerant Tracking Control of Near-Space Vehicle Using Takagi–Sugeno Fuzzy Models," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 5, pp. 1000-1007, 2010.

[5] L. Mazmanyan and M.A. Ayoubi, "Fuzzy Attitude Control of Spacecraft With Fuel Sloshing via Linear Matrix Inequalities," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 5, 2018.

[6] A. Bello, A. Del Castañedo, K.S. Olfe, J. Rodríguez, and V. Lapuerta, "Parameterized fuzzy-logic controllers for the attitude control of nanosatellites in low earth orbits: A comparative study with PID controllers," *Expert Systems with Applications*, vol. 174, 2021, doi: 10.1016/j.eswa.2021.114679.

[7] R. Albareda, K.S. Olfe, A. Bello, J.J. Fernández, and V. Lapuerta, "Comparison of Optimization Methods for the Attitude Control of Satellites," *Electronics*, vol. 13, no. 17, p. 3363, 2024, doi: 10.3390/electronics13173363.

[8] J.B. Rawlings, D.Q. Mayne, and M.M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Madison, WI: Nob Hill Publishing, 2020.

[9] P. Krupa, D. Limon, and T. Alamo, "Implementation of model predictive control in programmable logic controllers," *IEEE Transactions on Control Systems Technology*, vol. 29, 2021, doi: 10.1109/TCST.2020.2992959.

[10] F. Gavilan, R. Vazquez, and E.F. Camacho, "Chance-constrained model predictive control for spacecraft rendezvous with disturbance estimation," *Control Engineering Practice*, vol. 20, 2012, doi: 10.1016/j.conengprac.2011.09.006.

[11] K. Dong, J. Luo, and D. Limon, "A novel stable and safe model predictive control framework for autonomous rendezvous and docking with a tumbling target," *Acta Astronautica*, vol. 200, pp. 176-187, 2022, doi: 10.1016/j.actaastro.2022.08.012.

[12] M. Mirshams and M. Khosrojerdi, "Attitude control of an underactuated spacecraft using tube-based MPC approach," *Aerospace Science and Technology*, vol. 48, 2016, doi: 10.1016/j.ast.2015.09.018.

[13] T. Smith et al., “Astrobee: Free-Flying Robots for the International Space Station,” *IEEE Transactions on Field Robotics*, doi: 10.1109/TFR.2026.3664810.

[14] K. Olfe, J. M. Ezquerro, J. Rodríguez, A. Arshadi, A. Laverón-Simavilla and V. Lapuerta, “Attitude control experiment design for OPS-SAT Missions,” *IEEE Aerospace and Electronic Systems Magazine*, doi: 10.1109/MAES.2026.3664808

[15] SPCIES: Suite for Predictive Controllers in Industrial Embedded Systems. [Online]. Available: https://github.com/GepocUS/Spcies

[16] L. A. Zadeh, “Fuzzy sets,” *Information and Control*,vol.8, 1965, doi: 10.1016/S0019-9958(65)90241-X.

[17] K. Albee, M. Ekal, C. Oestreich, and P. Roque, A Brief Guide to Astrobee’s Flight Software.", 2021.

[18] E. F. Camacho and C. B. Alba, Model Predictive Control. *Springer Science and Business Media*, 2013.

[19] Y. Yang, *Spacecraft Modeling, Attitude Determination, and Control: Quaternion-Based Approach*. Chapters 3–4.