DOI: 10.31276/VJSTE.2025.0083 *Mathematics and Computer Science* | *Computational Science;*
Physical Sciences | *Engineering*

Enhancing real-time drone trajectory tracking performance using model predictive control on FPGA-SoC

Quang Kien Trinh[1*], Van Tuan Nguyen[2], Thi Hong Tham Tran[1], Thanh Bang Le[1], Minh Thuong Nguyen[3], Phu Tuan Duong[3]

[1]Le Quy Don Technical University, 236 Hoang Quoc Viet Street, Nghia Do Ward,Hanoi, Vietnam

[2]Telecommunications University, 101 Mai Xuan Thuong Street, Bac Nha Trang Ward, Khanh Hoa Province, Vietnam

[3]Institute of Information Technology and Electronics, 17 Hoang Sam Street, Nghia Do Ward, Hanoi, Vietnam

Received 11 October 2025; revised 11 December 2025; accepted 23 March 2026

Abstract:

This study presents a robust control framework for enhancing real-time trajectory tracking of drones, overcoming the limitations of conventional proportional - integral - derivative (PID) controllers. The proposed architecture integrates a model predictive controller (MPC) with a Kalman filter to achieve accurate state estimation under noisy measurements, while explicitly handling system constraints. To address the computational bottleneck of MPC, we employ a fast gradient method-based solver with warm-start initialisation and fixed-iteration updates, ensuring deterministic and efficient execution. Experimental results on a Zynq-7000 FPGA-SoC demonstrate that the optimised software-only implementation achieves real-time feasibility, with an average computation time of 0.18 ms (≈1.4% of a 12.8 ms control cycle). Compared to PID, the proposed MPC reduces the maximum absolute axis tracking error from ≈0.5 to ≈0.2 m and generates significantly smoother control signals (~13x lower variance and =7x lower mean rate of change). These improvements enhance trajectory accuracy and flight stability while extending actuator lifetime, all achieved without requiring hardware acceleration.

Keywords: **drone, fast model predictive controller, proportional - integral - derivative, trajectory tracking.**

Classification numbers: **1.3, 2.3**

1. Introduction

Unmanned aerial vehicles, commonly referred to as drones, are increasingly utilised across diverse applications such as surveillance, delivery services, precision agriculture, and infrastructure inspection. However, achieving reliable trajectory tracking, particularly in noisy and dynamic real-world environments with stringent response-time requirements,

[*]*Corresponding author: Email: kien.trinh@lqdtu.edu.vn*

remains a significant challenge. Drone control systems must continuously process multi-sensor data to estimate position, orientation, and velocity relative to a predefined reference trajectory [1]. Any processing delay can result in trajectory deviations and degraded flight performance.

Conventional controllers, such as the PID controller, are widely adopted due to their simplicity and ease of implementation. Nevertheless, PID control exhibits notable limitations when applied to complex, nonlinear, and constrained systems such as drones. Specifically, PID parameter tuning can be difficult and time-consuming, particularly under varying operating conditions. Moreover, PID lacks the capability to predict future system behaviour, often resulting in delayed responses or excessive oscillations in the presence of disturbances or sudden trajectory changes.

To overcome these drawbacks, advanced control strategies have been investigated, among which MPC stands out for its ability to predict future system states and compute optimal control inputs while explicitly handling physical constraints. Unlike PID, MPC not only reacts to current errors but also proactively minimises future deviations, enabling smoother and more accurate trajectory tracking even during transient phases [2].

Advances in embedded computing, particularly with Field-Programmable Gate Array (FPGA) and System-on-Chip (SoC) platforms [3, 4], have been pivotal in migrating MPC to high-speed, real-time applications. These platforms enable a hardware-software co-design approach, where the massive parallelism of FPGAs accelerates MPC's core computations, while integrated processors handle higher-level tasks. This strategy has proven effective in experimental research, achieving control-loop frequencies in the kilohertz to megahertz range for demanding fields like power electronics, robotics, and automotive systems [5].

Numerous studies have applied MPC to UAV control problems, addressing aspects such as motion planning in uncertain environments [6], autonomous landing on moving platforms [7], and coordinated formation flight with collision avoidance [8, 9]. Parallel to these MPC advancements, classical control strategies have also evolved. Notably, V.P. Shankaran, et al. (2021) [10] recently proposed a fractional-order PI plus D (PI^λ-D) controller, demonstrating that fractional calculus can significantly enhance stability regions and robustness for integrating plants compared to integer-order schemes. However, while such advanced PID variants offer improved performance, they still lack the inherent ability to explicitly handle system constraints and predict future states. Unlike MPC, these feedback-based approaches generally lack the capability to optimise future trajectories while explicitly enforcing system constraints. Comparative analyses between MPC and PID for quadrotor trajectory tracking have been reported [11, 12], but significant challenges remain in implementing such computationally intensive algorithms on embedded systems with limited resources. The approach in [11], for example, is confined

to simulations and addresses only a single degree of freedom (e.g., altitude control), thus failing to capture the full 3D trajectory-tracking problem. In contrast, the study in [12] highlights the computational bottlenecks of conventional microcontrollers, which necessitate compromises such as long sampling times (up to 100 ms) and shortened prediction horizons, ultimately hindering real-time performance and control quality.

To address these issues, this study proposes a comprehensive approach that not only integrates a Kalman filter with MPC for improved state estimation under noisy conditions but also focuses on efficient implementation on an FPGA-SoC (Zynq-7000) platform. The primary contribution of this work is the design, implementation, and experimental validation of a complete control pipeline executed entirely in software on the processing system of a single FPGA-SoC platform, enabling high-performance MPC to meet hard real-time constraints. We specifically address the computational bottleneck of traditional MPC by: (1) employing a solver based on the fast gradient method, (2) leveraging warm-start and fixed-iteration techniques to ensure deterministic execution, and (3) demonstrating that this optimised software approach is sufficient to meet the deadline without requiring hardware acceleration.

The remainder of this paper is organised as follows: Section 2 presents the control system design, including the overall architecture, drone modelling, Kalman filter parameterisation, and MPC formulation. Section 3 details the experimental setup, performance evaluation, and comparative analysis between MPC and PID. Finally, Section 4 concludes the article.

2. Control system design

This section details the closed-loop architecture (Kalman filtering and MPC), drone modelling and estimator parameterisation, and the solver specialisation that enables deterministic timing on embedded hardware.

2.1. Overall system architecture

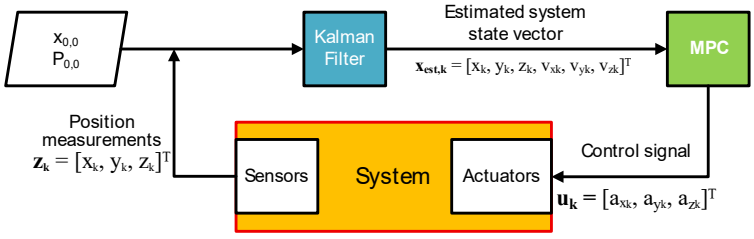


Fig. 1. System architecture for trajectory tracking using MPC with Kalman filter integration.

Figure 1 illustrates the closed-loop architecture of the proposed fast MPC framework. To enable a fair comparison with the PID-based controller, the same feedback state-estimation concept was retained in both control structures. Specifically, similar to the PID scheme, the measured output of the system, $\boldsymbol{z_k = [x_k, y_k, z_k]^T}$, is first processed by a

Kalman filter to reconstruct the complete estimated state vector $\boldsymbol{x_{est,k}} = \left[\boldsymbol{x_k}, \boldsymbol{y_k}, \boldsymbol{z_k}, \boldsymbol{v_{xk}}, \boldsymbol{v_{yk}}, \boldsymbol{v_{zk}}\right]^{\boldsymbol{T}}$. The fast MPC controller then uses the estimated state x_est together with the reference trajectory x_ref to compute the control input $\boldsymbol{u_k} = \left[\boldsymbol{a_{xk}}, \boldsymbol{a_{yk}}, \boldsymbol{a_{zk}}\right]^{\boldsymbol{T}}$. In this way, both PID and fast MPC operate on the same type of estimated-state feedback, and the comparison focuses on the difference between the PID control law and the optimisation-based fast MPC strategy.

2.2. Drone modelling

The system under study is a drone navigating in three-dimensional space along a predefined trajectory, as depicted in Fig. 2. To comprehensively evaluate the system's response, the trajectory was designed to include various flight patterns. The drone's movement is controlled via acceleration commands, which are updated every 12.8 ms. A GPS sensor provides position feedback for real-time trajectory correction. The system model accounts for white Gaussian noise affecting both the control acceleration inputs and the GPS measurements.

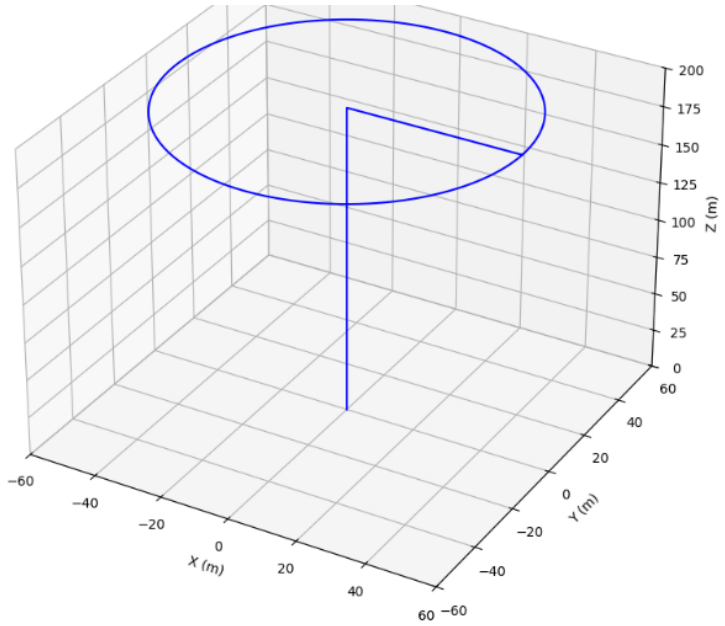


Fig. 2. The predefined 3D reference trajectory for the drone.

2.3. Kalman filter

To provide the MPC with accurate state estimates from noisy GPS data, a linear Kalman filter (LKF) was selected [13]. This choice strikes a deliberate balance between estimation accuracy and computational load, making it suitable for embedded deployment under real-time constraints [14]. Assuming the control acceleration remains constant during each 12.8 ms cycle, the discrete state-space model and measurements are formulated as follows [15]:

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + G\mathbf{u}_k + \mathbf{w}_k, \qquad \mathbf{z}_k = H\mathbf{x}_k + \mathbf{v}_k \tag{1}$$

where $u_k = [a_{xk}, a_{yk}, a_{zk}]^T$, $\mathbf{w}_k \sim \mathrm{N}(0, \mathbf{Q})$, $\mathbf{x}_k = [x_k, y_k, z_k, v_{xk}, v_{yk}, v_{zk}]^T$ and $\mathbf{v}_k \sim \mathrm{N}(0, \mathbf{R})$ for a constant-acceleration kinematic model and sampling time updated every $\Delta t = 12.8$ ms:

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_3 & \Delta t \mathbf{I}_3 \\ 0 & \mathbf{I}_3 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 0.5\Delta^2 t \mathbf{I}_3 \\ \Delta t \mathbf{I}_3 \end{bmatrix}, \text{and } \mathbf{H} = \begin{bmatrix} \mathbf{I}_3 & 0 \end{bmatrix} \tag{2}$$

We use a LKF with measurement noise matrix $\mathbf{R} = \mathrm{diag}\left(\left[\sigma^2_{x\_gps}, \sigma^2_{y\_gps}, \sigma^2_{z\_gps}\right]\right)$ and process noise matrix Q:

$$\mathbf{Q} = \sigma_a^2 \begin{bmatrix} \frac{\Delta t^4}{4}\mathbf{I}_3 & \frac{\Delta t^4}{4}\mathbf{I}_3 \\ \frac{\Delta t^3}{2}\mathbf{I}_3 & \Delta t^2 \mathbf{I}_3 \end{bmatrix} \tag{3}$$

In our setup, $\sigma_{x\_gps}, \sigma_{y\_gps}, \sigma_{z\_gps}$ follow vendor GPS specifications (i.e., noisy outdoor), and σ_a is tuned to 0.15 m/s^2 to accelerate covariance convergence while keeping MPC iterates within box constraints. This calibration reduces estimate jitter and stabilises the condensed QP gradient, improving fixed-iteration convergence**.**

2.4. Model predictive controller

2.4.1. The constrained optimisation problem

To address the challenges of achieving high performance and low latency in real-time drone control, this study adopts fast MPC, which is a variant of MPC specifically optimised for applications with short control cycles. The algorithm leverages efficient numerical techniques, including the fast gradient method (FGM) and warm-start initialisation [16].

The system dynamics are described by the prediction model:

$$x_{k+1} = \mathbf{F}x_k + \mathbf{G}u_k \tag{4}$$

where $x_k \in \mathbb{R}^{n_x}$ is the state vector at time step k, $u_k \in \mathbb{R}^{n_u}$ is the control input vector at time step k, $\boldsymbol{F} \in \mathbb{R}^{n_x \times n_x}$ is the state-transition matrix (from Kalman filter modelling), and $\boldsymbol{G} \in \mathbb{R}^{n_x \times n_u}$ is the control-input matrix (from Kalman filter modelling).

The cost function is formulated as:

$$J(x_0, \mathbf{U}) = \left(x_N - x_{\mathrm{ref},N}\right)^T \mathbf{Q}_\mathbf{f} \left(x_N - x_{\mathrm{ref},N}\right) + \sum_{k=0}^{N-1} \left[\left(x_k - x_{\mathrm{ref},k}\right)^T \mathbf{Q} \left(x_k - x_{\mathrm{ref},k}\right) + u_k^T \mathbf{R} u_k\right] \tag{5}$$

where N is the prediction horizon, x_0 is the known initial state, $\mathbf{U} = [u_0^T, u_1^T, \ldots, u_{N-1}^T]^T \in \mathbb{R}^{n_u \cdot N}$, $x_{\mathrm{ref},k}$ is the reference state at step k, $\boldsymbol{Q} \in \mathbb{R}^{n_x \times n_x}$ is the weight matrix for states, $\boldsymbol{R} \in \mathbb{R}^{n_u \times n_u}$ is the weight matrix for inputs, and $\boldsymbol{Q_f} \in \mathbb{R}^{n_x \times n_x}$ is the terminal cost weight matrix for states.

Control constraints are defined as:

$$u_{\min,j} \le u_{k,j} \le u_{\max,j} \quad \forall k \in [0, N-1], \forall j \in [1, n_u] \tag{6}$$

or

$$\mathbf{U}_{\min} \le \mathbf{U} \le \mathbf{U}_{\max}. \tag{7}$$

This optimisation problem can be expressed in standard quadratic programming (QP) form [17]:

$$J(\mathbf{U}) = \frac{1}{2}\mathbf{U}^T H_m \mathbf{U} + f_m^T \mathbf{U} + \text{const} \tag{8}$$

The Hessian matrix is given as:

$$\mathbf{H}_m = 2\left(\mathbf{S}_u^T \mathbf{Q}_{\text{blk}} \mathbf{S}_u + \mathbf{R}_{\text{blk}}\right) \tag{9}$$

The gradient vector is:

$$f_m = 2\mathbf{S}_u^T \mathbf{Q}_{\text{blk}} \left(\mathbf{S}_x x_0 - \mathbf{X}_{\text{ref}}\right) \tag{10}$$

Finally, the cost function gradient is:

$$\nabla J(\mathbf{U}) = \mathbf{H}_m \mathbf{U} + f_m, \tag{11}$$

with

$$\mathbf{S}_x = \left[\mathbf{F},\ \mathbf{F}^2,\ \ldots,\ \mathbf{F}^N\right]^T \in \mathbb{R}^{(N\cdot n_x)\times n_x} \tag{12}$$

$$\mathbf{S}_u = \begin{bmatrix} \mathbf{G} & 0 & 0 & \cdots & 0 \\ \mathbf{FG} & \mathbf{G} & 0 & \cdots & 0 \\ \mathbf{F}^2\mathbf{G} & \mathbf{FG} & \mathbf{G} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{F}^{N-1}\mathbf{G} & \mathbf{F}^{N-2}\mathbf{G} & \cdots & \mathbf{FG} & \mathbf{G} \end{bmatrix} \in \mathbb{R}^{(N\cdot n_x)\times(N\cdot n_u)} \tag{13}$$

$$\mathbf{X}_{\text{ref}} = \begin{bmatrix} x_{\text{ref},1}^T & x_{\text{ref},2}^T & \cdots & x_{\text{ref},N}^T \end{bmatrix}^T \in \mathbb{R}^{n_x \cdot N} \tag{14}$$

$$\mathbf{Q}_{\text{blk}} = \begin{bmatrix} \mathbf{Q} & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{Q} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mathbf{Q} & 0 \\ 0 & 0 & \cdots & 0 & \mathbf{Q}_f \end{bmatrix} \in \mathbb{R}^{(n_x\cdot N)\times(n_x\cdot N)} \tag{15}$$

$$\mathbf{R}_{\text{blk}} = \begin{bmatrix} \mathbf{R} & 0 & \cdots & 0 \\ 0 & \mathbf{R} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{(n_u\cdot N)\times(n_u\cdot N)} \tag{16}$$

The MPC solves this constrained optimisation problem at each time step [17], aiming to compute a future control sequence that drives the predicted system trajectory to closely follow the reference trajectory while satisfying control and dynamic constraints given by

$$\min_{\mathbf{U}_{\min} \le \mathbf{U} \le \mathbf{U}_{\max}} J(\mathbf{U}) \tag{17}$$

2.4.2. Real-time QP solver (fixed-iteration FGM)

We solve the condensed QP using a fixed number of FGM iterations. At each control step, we warm-start by shifting the previously optimised input sequence by one stage and

appending a zero input at the end. Each FGM iteration consists of (i) one gradient step on the unconstrained objective and (ii) a Euclidean projection onto the input box-constraint set: $\mathcal{U} \triangleq \{u \in \mathbb{R}^{3N} \mid u_{\min} \leq u \leq u_{\max}\}$.

This yields strictly bounded compute and memory footprints. On the Zynq-7000 processing system, the average MPC time is 0.18 ms (1.4% of the 12.8 ms cycle), with no observed overruns in our tests.

Algorithm 1. Pseudo code for real-time MPC using FGM with fixed iterations and warm-start.

Input: prediction horizon N, number of iterations m, learning rate α, weighting matrices $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{Q}_f$, control limits $U_{\max}, U_{\min}$.

Output: optimal control input sequence $\boldsymbol{U}_{optimal}$

1: Set $U^{(0)} \leftarrow U_{\text{prev}}$ (If unavailable, set $U^{(0)}$ from the desired acceleration sequence) [Warm start]

/ Main loop */*

2: for $i = 0$ to $m - 1$ **do**

3: $\nabla J\left(U^{(i)}\right) = \mathbf{H}_m U^{(i)} + f_m$ [a]

4: $U^{(i+1)}_{\text{unconstrained}} = U^{(i)} - \alpha \nabla J\left(U^{(i)}\right)$ [b]

5: $U^{(i+1)} = \text{proj}_{[U_{\min}, U_{\max}]}\left(U^{(i+1)}_{\text{unconstrained}}\right) = \begin{cases} U_{\min}, & \text{if } U^{(i+1)}_{\text{unconstrained}} < U_{\min} \\ U^{(i+1)}_{\text{unconstrained}}, & \text{if } U_{\min} \leq U^{(i+1)}_{\text{unconstrained}} \leq U_{\max} \\ U_{\max}, & \text{if } U^{(i+1)}_{\text{unconstrained}} > U_{\max} \end{cases}$ [c]

6: end for

7: $U_{\text{optimal}} \approx U^{(m)}$

8: $U_{\text{prev}} = \left[u_{o1}, u_{o2}, \ldots, u_{o(N-1)}, 0\right])$ [d]

[a] Compute the gradient of the cost function at $U^{(i)}$
[b] Gradient descent update (unconstrained step)
[c] Projection onto control constraints
[d] Prepare warm start for the next step: Shift the control sequence forward by one block and append a zero block at the end from the optimal solution.

2.4.3. Nominal stability analysis

To assess the closed-loop stability, we analyse the discrete-time linear model used by the controller. Following standard MPC practice, an equivalent LQR gain K_{eq} is computed from the stage weights Q, R. The nominal closed-loop matrix is formed as $F_{cl} = F - GK_{eq}$.

We evaluate the Lyapunov condition by solving the discrete-time Lyapunov equation:

$$F_{cl}^{\bullet} P F_{cl} - P = -Q, \quad Q \succ 0 \tag{18}$$

With $Q = I$, a positive definite solution $P \succ 0$ is obtained, yielding eigenvalues $\lambda_{\min}(P) =$

4.61 and $\lambda_{\max}(P) = 142.44$. The spectral radius satisfies $\rho(F_{cl}) = 0.939 < 1$, confirming that the nominal closed-loop system is asymptotically stable.

3. Experimental results and analysis

3.1. Experimental setup

The system was implemented on an FPGA-SoC Zynq-7000 platform. The hardware design, including the processing system (PS) configuration and peripheral interfaces, was developed using Xilinx Vivado. The control software, executed on the ARM cores, was developed in Xilinx Vitis. Data transfer from the board to the host computer was performed via a UART interface.

To ensure a fair and transparent evaluation, both controllers were tested under identical trajectory tracking tasks and hardware conditions. The comparative study focuses on three key aspects: (i) trajectory tracking accuracy, quantified through maximum error and error distribution across all three axes; (ii) control signal quality, evaluated in terms of smoothness and variance to assess stability and actuator stress; and (iii) computational performance, measured by the average per-cycle execution time.

With the evaluation criteria defined, we next specify the controller parametrisation used in all experiments.

3.1.1. MPC parameters

The selection and tuning of the controller parameters were carried out based on a set of criteria aimed at optimising overall performance, including: (i) ensuring stable operation of the UAV with smooth motion, (ii) maintaining trajectory tracking accuracy within allowable error bounds, and (iii) keeping computation time within real-time requirements.

Table 1. Parameters used in the fast MPC implementation.

Parameter	**Initial value**
Prediction horizon N	N=5
Number of iterations m	m=100
Learning rate α	$\alpha = 10^{-2}$
Weight matrix for states **Q** (position and velocity)	**Q**=diag([200, 200, 200, 20, 20, 20])
Weight matrix for inputs **R** (acceleration)	**R**=diag([0.01, 0.01, 0.01])
Terminal cost weight matrix for states $\mathbf{Q}_f$	$\mathbf{Q}_f$=10*$\mathbf{Q}$
Control limits U_{max}	$U_{max}=[10, 10, 10]^T$
Control limits U_{min}	$U_{min}=[-10, -10, -10]^T$

The resulting optimal parameters are: $N = 15$, $m = 30$, $\alpha = 4\times10^{-2}$, $\mathbf{Q} = \mathrm{diag}([150,150,150,5,5,5])$, $\mathbf{Q}_f = 50\cdot\mathbf{Q}$, and $\mathbf{R} = \mathrm{diag}([0.15,0.15,0.15])$.

3.1.2. Proportional - integral - derivative controller parameters

To enable a comparative evaluation of the proposed MPC, a PID controller was also implemented for the same trajectory tracking problem. This controller computes the control input $u(t)$ based on the tracking error $e(t)$ between the reference trajectory and the actual system state, using a weighted combination of the proportional (P), integral (I), and derivative (D) terms. The parameters, tuned for the simulation, were set as follows: $k_p = 34.2$, $k_i = 8.0$, and $k_d = 23.4$.

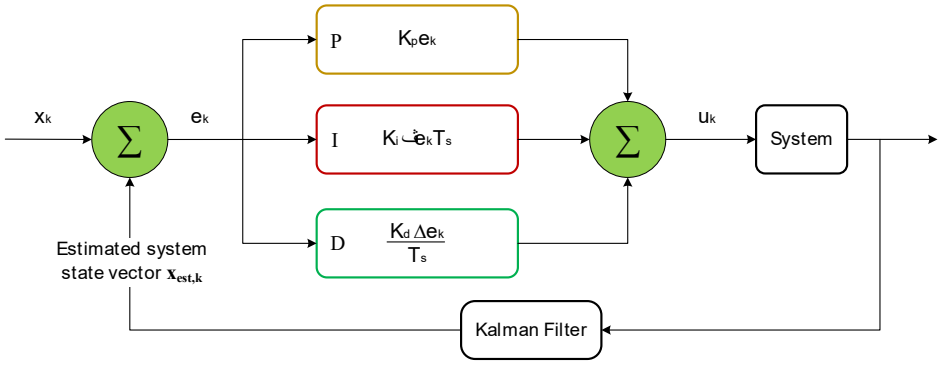
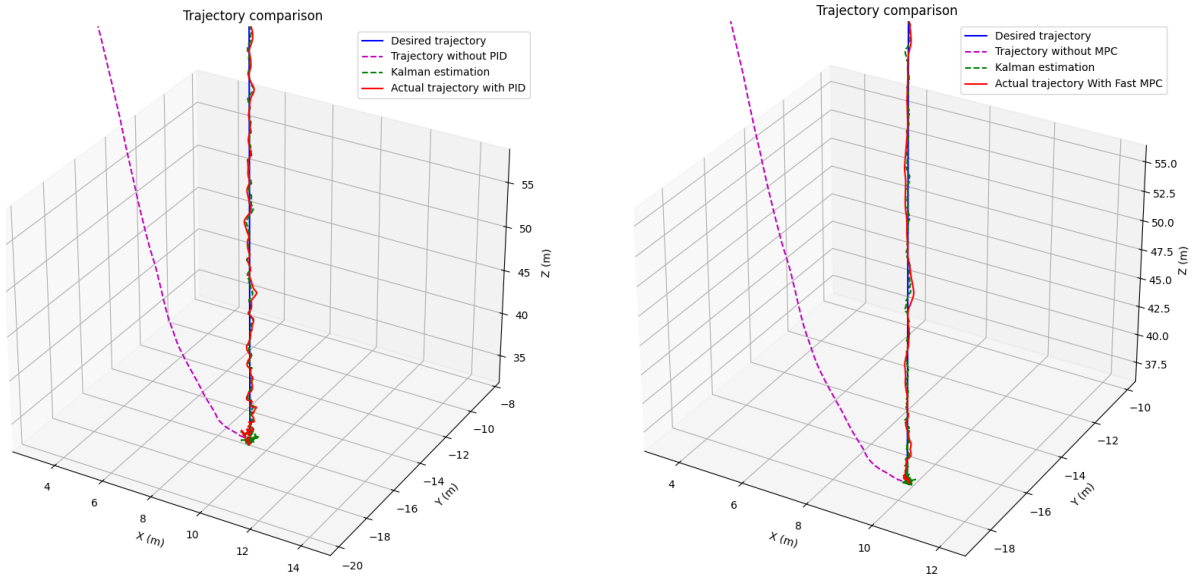


Fig. 3. Block diagram of the PID control system for trajectory tracking.

3.2. Trajectory tracking performance analysis

The trajectory tracking performance of the MPC and PID controllers was experimentally evaluated.

3.2.1. Flight paths and error profiles

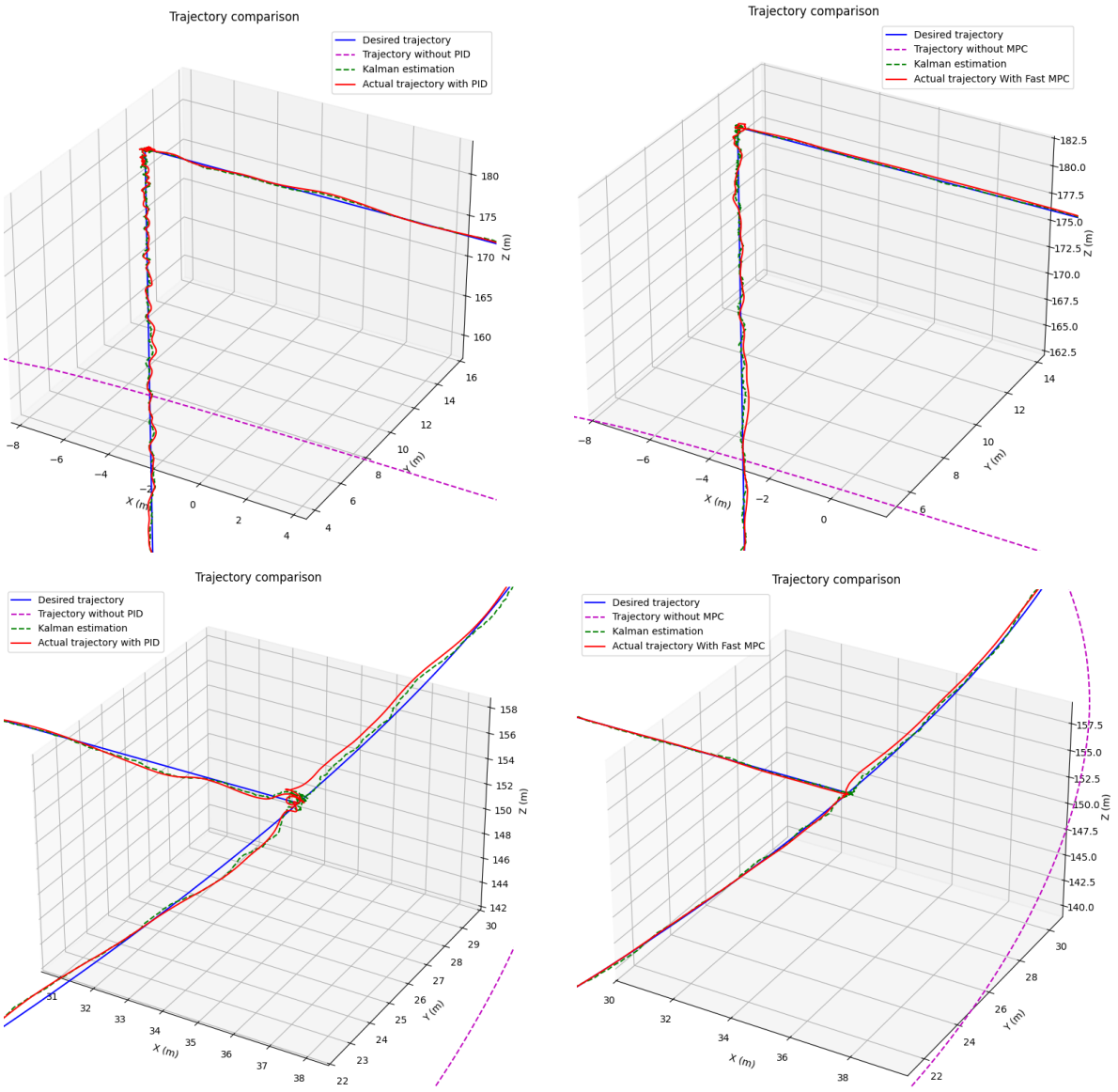


PID **MPC**

Fig. 4. Trajectory tracking performance during the initial flight phase: A comparison between MPC and PID controllers.

The results, illustrated in Figs. 4 and 5, demonstrate the drone's flight paths during different phases of operation. As shown in those figures, the MPC controller provides

superior tracking from the initial phase and during complex transitional manoeuvres. Its predictive capability allows it to anticipate trajectory changes, resulting in a flight path that closely follows the desired route with minimal deviation. In contrast, the PID controller, which reacts only to existing errors, exhibits noticeable overshoot and oscillations when the trajectory changes abruptly.



PID MPC

Fig. 5. Trajectory tracking performance during complex transitional manoeuvres: A comparison between MPC and PID controllers.

3.2.2. Quantitative performance comparison

Table 2 summarises the main tracking and control-signal metrics. The fast MPC controller achieves consistently higher accuracy than the PID controller across all axes. Specifically, MPC yields lower RMSE values (0.082-0.088 m vs. 0.108-0.113 m) and

significantly smaller maximum absolute errors (<0.24 m vs. up to 0.61 m). These improvements are also reflected in the integral error metrics (IAE, ISE, ITAE, ITSE), where MPC reduces total accumulated error by approximately 20-30%.

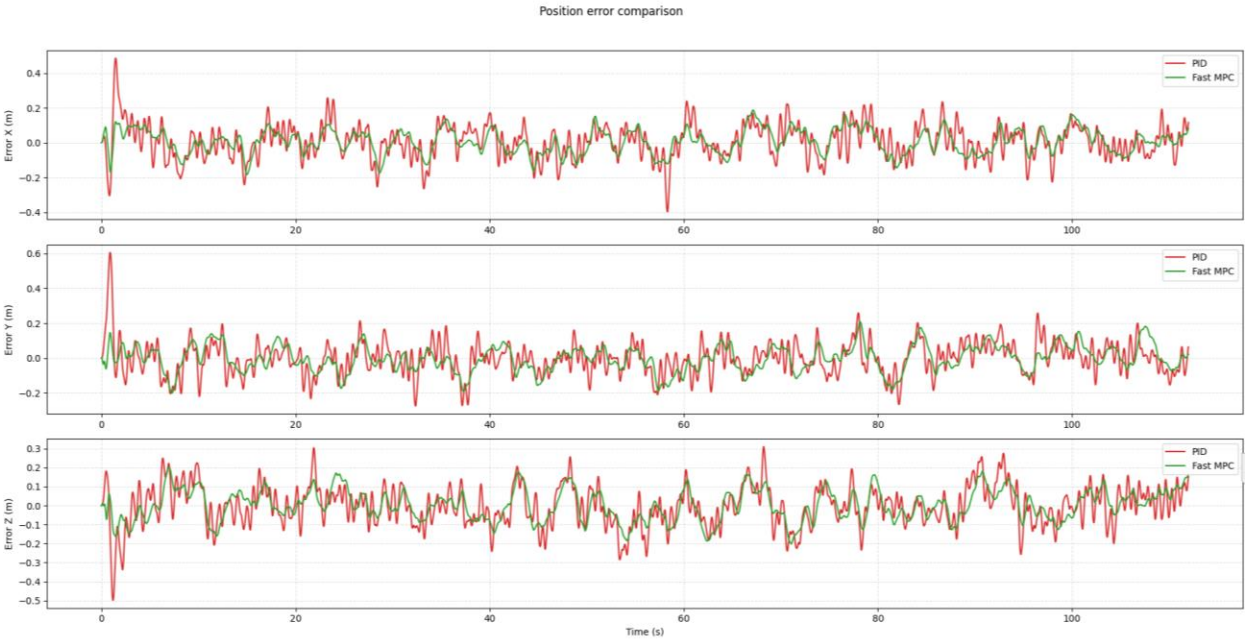


Fig. 6. Comparative analysis of trajectory-tracking errors along the X, Y, and Z axes: MPC vs. PID.

The quality of the generated control signals further differentiates the two controllers. The fast MPC outputs significantly smoother acceleration profiles, with mean rates of change of 16.19-16.58 m/s³ and variances of only 1.17-1.86 $(m/s^2)^2$. In contrast, the PID controller exhibits aggressive fluctuations, with mean rates exceeding 114 m/s³ and variances above 22 $(m/s^2)^2$. These high-frequency oscillations in the PID actuation directly contribute to its larger maximum absolute errors (up to 0.61 m) and higher accumulated error metrics (e.g., IAE 9.26-10.05 vs. 7.29-8.27 for MPC). Overall, the numerical results confirm that fast MPC provides smoother actuation and more stable, well-damped trajectory tracking, while remaining fully compatible with real-time execution on the embedded platform.

Table 2. Quantitative comparison between MPC and PID.

Performance metric	MPC controller	PID controller
Max Abs Error (X, Y, Z) (m)	[0.236, 0.236, 0.207]	[0.486, 0.606, 0.499]
RMSE (X, Y, Z) (m)	[0.082, 0.086, 0.088]	[0.111, 0.108, 0.113]
IAE (X, Y, Z) (m·s)	[6.986, 8.737, 7.764]	[12.433, 10.215, 9.943]
ISE (X, Y, Z) ($m^2 \cdot s$)	[0.673, 1.029, 0.849]	[2.371, 1.679, 1.362]
ITAE (X, Y, Z) ($m \cdot s^2$)	[398.531, 502.689, 431.181]	[705.565, 588.987, 546.251]
ITSE (X, Y, Z) ($m^2 \cdot s^2$)	[39.581, 61.222, 46.987]	[119.079, 88.491, 72.828]

Control signal	[16.189, 16.276, 16.578]	[114.693, 114.122, 114.167]
Smoothness (Mean rate of change) (m/s^3)		
Control signal variance (ax, ay, az) ($(m/s^2)^2$)	[1.658, 1.855, 1.171]	[22.391, 22.964, 22.638]
Average computation Time (ms)	0.18	0.02

3.2.3. Control signal quality

The superior quality of the MPC-generated control signal, illustrated in Fig. 7, is quantitatively validated by the results summarised in Table 2. The MPC controller produces substantially lower control-signal variance across all axes (e.g., 1.66 $(m/s^2)^2$ on the x-axis), in contrast to the PID controller, whose variance exceeds 22 $(m/s^2)^2$ on all axes. A similar trend is observed in the smoothness metric, where the mean rate of change of the MPC control input remains approximately 16 m/s^3, while the PID controller exhibits values above 114 m/s^3. These findings indicate that the MPC generates a markedly smoother and more deliberate control profile, free from the high-frequency fluctuations that characterise PID-based actuation. Such smoothness not only enhances the overall stability of the vehicle but also yields practical benefits, including reduced energy consumption and decreased mechanical stress on actuators, ultimately extending the operational lifespan of the drone system.

3.2.4. Disturbance rejection capability

To evaluate robustness, a simulated constant lateral wind gust of $0.6\ m/s^2$ was injected into the Y-axis acceleration channel from $t = 20$ s to $t = 35$ s. The fast MPC maintained stable behaviour with a bounded steady-state error of approximately 0.77 m during the disturbance. As illustrated in Fig. 8, once the gust ceased, the controller rapidly recovered the trajectory with a rejection time of 2.85 s, eventually returning to its nominal tracking performance (0.133 m steady-state error). These results highlight the high robustness of the solver even without explicit integral action.

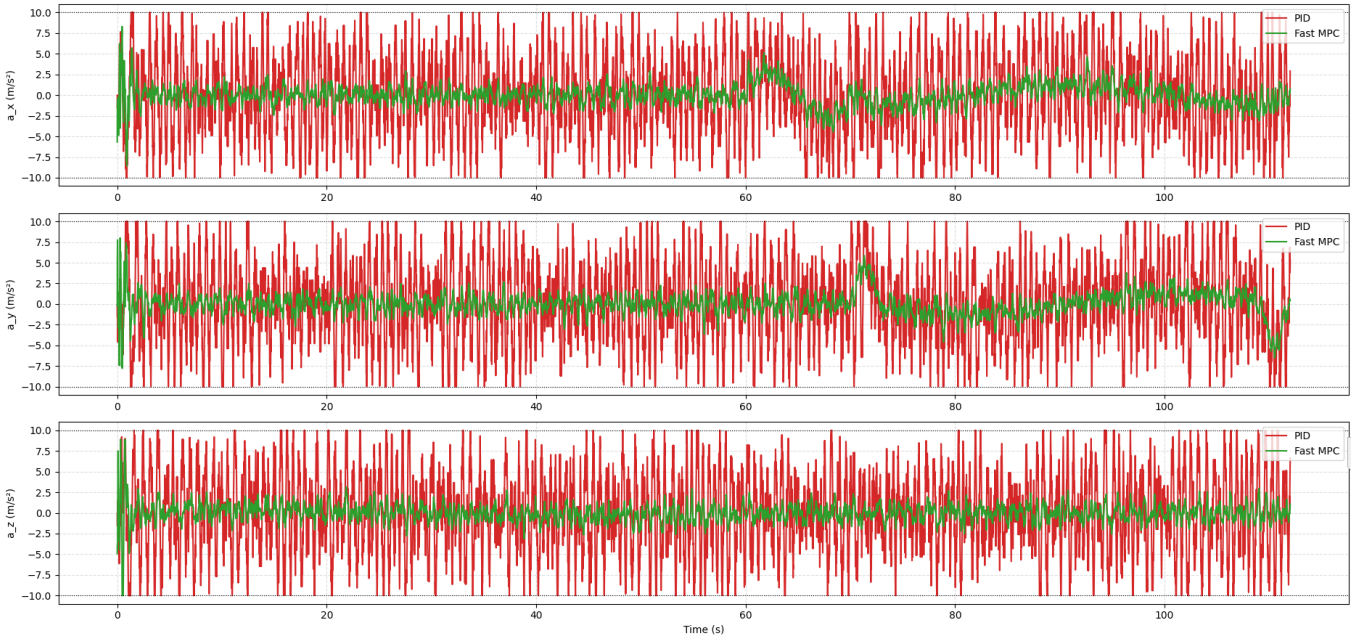


Fig. 7. Comparison of control signals (acceleration) generated by MPC and PID controllers.

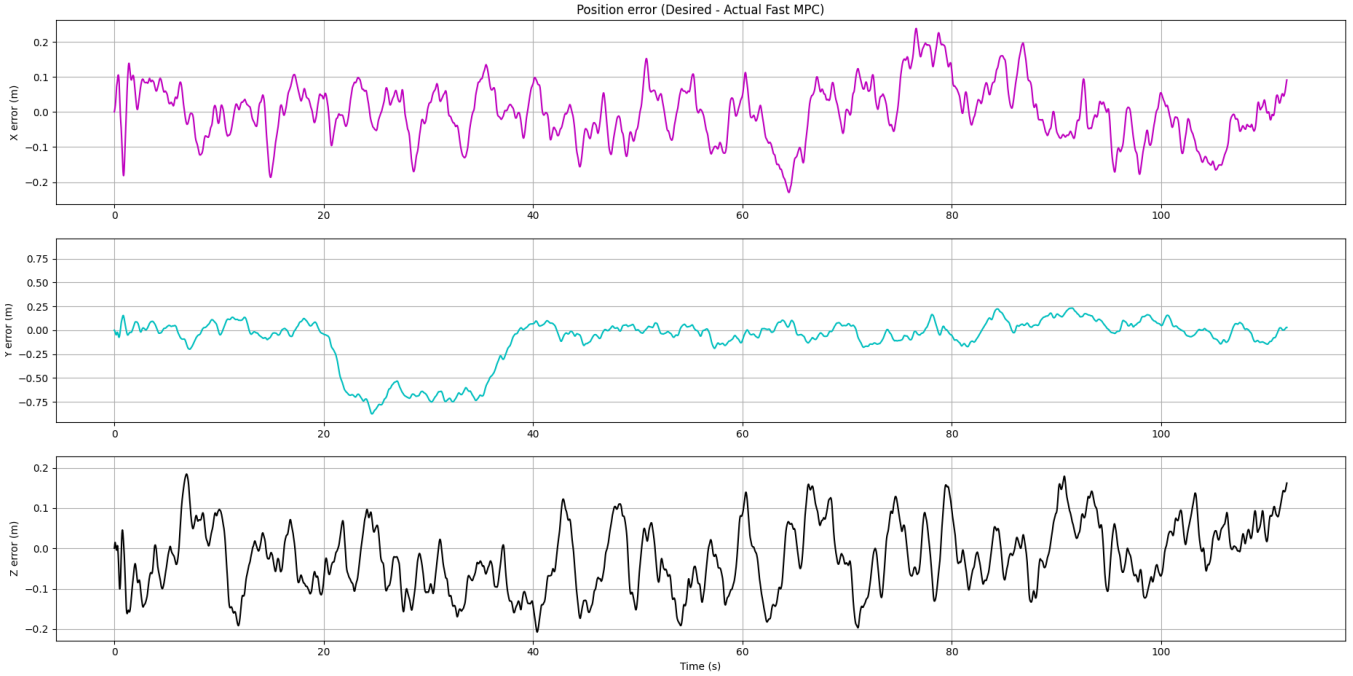


Fig. 8. Fast MPC response to external disturbance.

3.2.5. Step-response and transient performance analysis

To provide a clearer quantitative interpretation of the trajectory-tracking responses shown in Fig. 9, the transient-response performance of the fast MPC and PID controllers is summarised in Table 3 in terms of rise time, peak time, overshoot, and settling time for the X-, Y-, and Z-axis motions.

On the X-axis, the fast MPC slightly improves all major transient indicators compared with PID, reducing the rise time from 0.819 s to 0.793 s, the peak time from 2.840 s to 1.241 s, the overshoot from 16.04% to 14.81%, and the settling time from 15.632 s to 15.086 s. On the Y-axis, the advantage of the fast MPC becomes more pronounced: the rise time is reduced from 0.806 s to 0.768 s, the overshoot decreases dramatically from 19.87% to 4.02%, and the settling time is shortened from 16.162 s to 7.961 s. Although the peak time on this axis increases from 2.557 s to 7.292 s, the substantially lower overshoot and much faster settling indicate a significantly better-damped response. On the Z-axis, the fast MPC again outperforms PID, with the rise time reduced from 0.832 s to 0.755 s, the peak time reduced from 2.614 s to 1.433 s, the overshoot reduced from 18.17% to 9.21%, and the settling time reduced from 14.623 s to 7.126 s.

Overall, the quantitative results in Table 3 demonstrate that the proposed fast MPC provides a faster initial response, significantly lower overshoot, and faster stabilisation than the PID controller, particularly on the Y- and Z-axes. These improvements confirm the superior damping and transient trajectory-tracking capability of the proposed control approach.

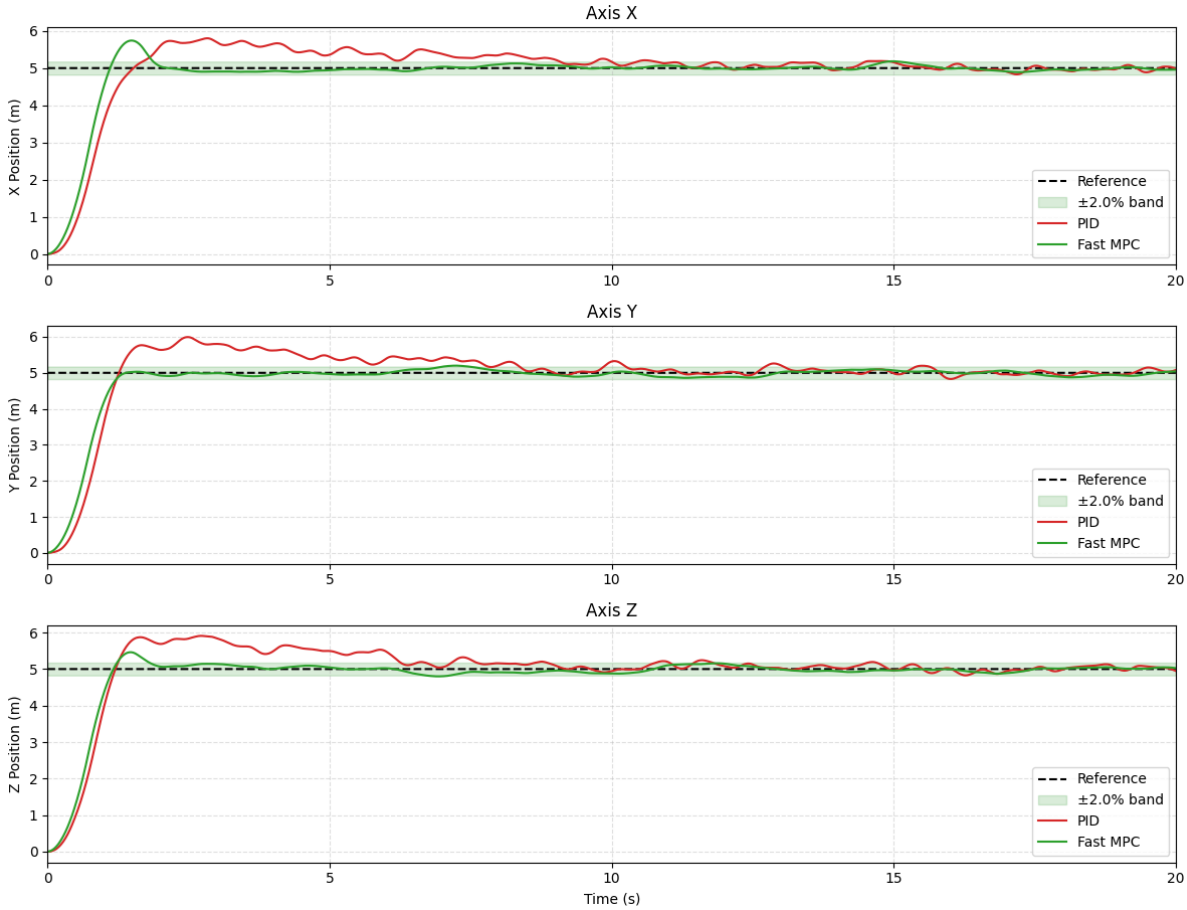


Fig. 9. Step-response comparison along the X, Y, and Z axes for PID and fast MPC.

Table 3. Transient response performance comparison between fast MPC and PID.

Metric	Fast MPC	PID
Rise time (X, Y, Z) (s)	[0.793, 0.768, 0.755]	[0.819, 0.806, 0.832]
Peak time (X, Y, Z) (s)	[1.241, 7.292, 1.433]	[2.840, 2.557, 2.614]
Overshoot (X, Y, Z) (%)	[14.81, 4.02, 9.21]	[16.04, 19.87, 18.17]
Settling time (X, Y, Z) (s)	[15.086, 7.961, 7.126]	[15.632, 16.162, 14.623]

3.2.6. Computational performance

As detailed in Fig. 10, there is a clear trade-off in computational demand. The PID controller is exceptionally fast, averaging 0.021 ms per cycle. The MPC is computationally more intensive, with an average time of 0.18 ms, approximately 8.6 times longer than the PID. However, this processing time is still far below the system's 12.8 ms control cycle deadline. This confirms that the fast MPC algorithm, implemented on the FPGA-SoC platform, meets the stringent real-time requirements for this application while delivering superior control performance.

Per control cycle, the timing budget covers sensing; LKF update; reference update; building the condensed-QP gradient; running FGM iterations with box projections; and writing to the actuators. The observed mean±SD timings sum <12.8 ms, with 1.4% spent on MPC, explaining why additional hardware acceleration is not required.

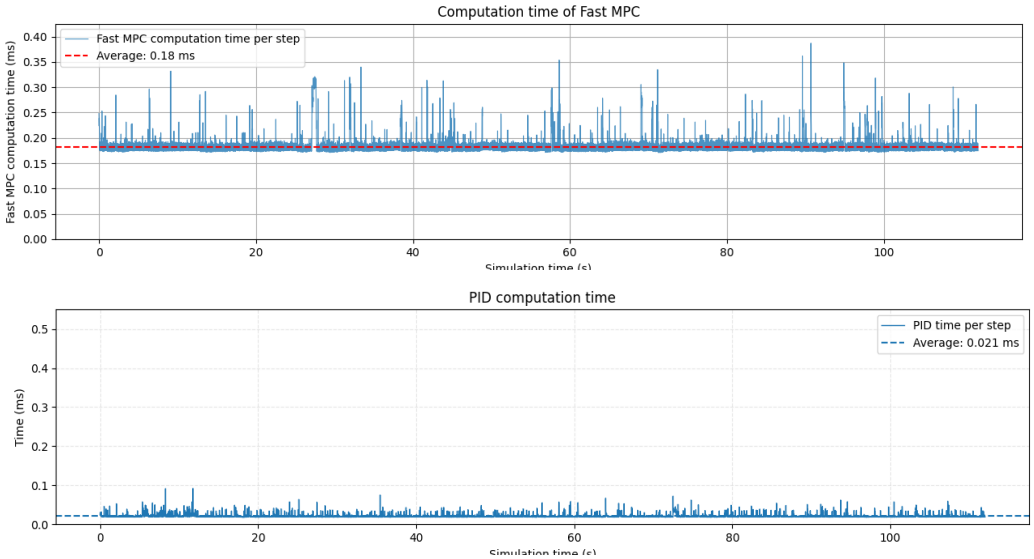


Fig. 10. Per-step computation time analysis: A comparison between MPC and PID implementations.

3.3. Discussion and comparative analysis

Our approach marks a notable improvement upon previous experimental works, such as that of M. Okasha, et al. (2022) [12], who also implemented MPC for 3D trajectory tracking. While their study confirmed the theoretical benefits of MPC, it also highlighted the severe computational bottlenecks of traditional microcontrollers. To achieve real-time

feasibility, they were forced to compromise the control system's performance by increasing the sampling time to 100 ms. This control cycle, nearly eight times longer than our 12.8 ms, has direct consequences for flight: longer sample periods introduce additional loop delay, weakening disturbance rejection and degrading tracking during agile manoeuvres or under external perturbations. By leveraging the computational power of an FPGA-SoC platform and an efficient solver, our approach overcomes this limitation, showing that the full potential of MPC can be realised without loosening control parameters. The key contrasts are summarised in Table 4.

Table 4. Comparative analysis of MPC implementation approaches.

Feature	Our study	M. Okasha, et al. (2022) [12]	G. Ganga, et al. (2017) [11]
Validation method	Experimental	Experimental	Simulation only
Implementation platform	FPGA-SoC (Zynq-7000)	Microcontroller	MATLAB Simulation
Scope of control problem	Full 3D trajectory tracking	Full 3D trajectory tracking	Single-axis altitude control
Control cycle	12.8 ms (High-fidelity)	100 ms (Hardware-Limited)	N/A (Non-real-time)
MPC computation time	0.18 ms (1.4% of cycle)	Not reported	N/A
Key outcome	High-performance, real-time MPC execution achieved without compromising control parameters.	MPC performance limited by hardware; control parameters were degraded for real-time feasibility.	Theoretical validation of MPC superiority over PID in a simplified, non-real-time context.

Relative to simulation-only studies, G. Ganga, et al. (2017) [11], our evaluation extends from single-axis altitude control to full 3D trajectory tracking (Table 4), which involves complex, coupled dynamics. Crucially, our model also moves beyond an idealised simulation by accounting for practical factors such as sensor noise, leveraging a kalman filter-MPC architecture to ensure robust performance. This demonstration of precise tracking under more demanding conditions confirms the viability of our proposed system for deployment in complex autonomous applications.

4. Conclusions

This study presented a comprehensive control solution that combines Kalman filtering with Model Predictive Control to enhance trajectory tracking under noisy conditions, while ensuring efficient real-time execution on an FPGA-SoC (Zynq-7000) platform. The main contribution lies in the complete design, implementation, and experimental validation of a full control pipeline executed purely in software on the SoC processing system. By employing a fast gradient method-based solver with warm-start initialisation and a fixed number of iterations, our approach overcomes the computational challenges of traditional MPC, ensuring deterministic performance. Crucially, the results demonstrate that this

optimised software-only implementation meets stringent real-time requirements without needing dedicated hardware acceleration, confirming its feasibility and practical effectiveness for UAV trajectory tracking applications.

CrediT author statement

Quang Kien Trinh: Conceptualisation, Methodology, Mathematical analysis, Writing - Reviewing, and Supervision. Van Tuan Nguyen: Formal analysis, Mathematical modelling, Validation, Writing - Original Draft; Thi Hong Tham Tran: Methodology, System modelling, Validation; Thanh Bang Le: Formal analysis, Simulation design, Performance evaluation, Data curation; Minh Thuong Nguyen: Simulation implementation, Numerical analysis, Writing - Reviewing; Phu Tuan Duong: Validation, Result interpretation, Writing - Review and Editing.

ACKNOWLEDGEMENTS

This study was supported by the Ministry of National Defence project, grant KC-KT.25/23.

COMPETING INTERESTS

The authors declare that there is no conflict of interest regarding the publication of this article.

REFERENCES

[1] S. Yeom (2025), "Drone state estimation based on frame-to-frame template matching with optimal windows", *Drones,* **9(7)**, p.457, DOI: 10.3390/drones9070457.

[2] J.B. Rawlings, D.Q. Mayne, M.M. Diehl (2017), *Model Predictive Control: Theory, Computation, and Design*, Nob Hill Publishing, 770pp.

[3] S. Saeidi (2015), *FPGA-Based Nonlinear Model Predictive Control of Electric Drives*, Doctoral Thesis, Technische Universität München.

[4] K.V. Ling (2008), "Embedded model predictive control (MPC) using a FPGA", *Proceedings of The 17th World Congress of the International Federation of Automatic Control (IFAC)*, **41(2)**, pp.15250-15255, DOI: 10.3182/20080706-5-KR-1001.02579.

[5] S.K. Dong, D. Nikiforov, W. Soedarmadji, et al. (2025), "Characterizing and optimizing real-time optimal control for embedded SoCs", *2025 IEEE International Symposium on Workload Characterization (IISWC)*, pp.489-503, DOI: 10.1109/IISWC66894.2025.00047.

[6] D.N. Bui, T.H. Khuat, M.D. Phung (2024), "Model predictive control for optimal motion planning of unmanned aerial vehicles", *2024 International Conference on Control, Robotics and Informatics (ICCRI)*, DOI: 10.1109/ICCRI64298.2024.00025.

[7] M. Misin, D.V.P. Cayuela (2020), *Model Predictive Controller of a UAV Using the LPV Approach*, Doctoral Dissertation, Universitat Politècnica de Catalunya.

[8] T. Baca, D. Hert, G. Loianno, et al. (2018), "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.6753-6760, DOI: 10.1109/IROS.2018.8594266.

[9] D. Oliveira, B.J. Guerreiro (2024), "In-flight capture maneuver of drones using model predictive control", *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp.771-778, DOI: 10.1109/ICUAS60882.2024.10556947.

[10] V.P. Shankaran, S.I. Azid, U. Mehta (2021), "Fractional-order PI plus D controller for second-order integrating plants: Stabilization and tuning method", *ISA Transactions*, **129(5)**, pp.592-604, DOI: 10.1016/j.isatra.2021.12.012.

[11] G. Ganga, M.M. Dharmana (2017), "MPC controller for trajectory tracking control of quadcopter", *International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pp.1-6, DOI: 10.1109/ICCPCT.2017.8074380.

[12] M. Okasha, J. Kralev, M. Islam (2022), "Design and experimental comparison of PID, LQR and MPC stabilizing controllers for Parrot Mambo mini-drone", *Aerospace*, **9(6)**, p.298, DOI: 10.3390/aerospace9060298.

[13] D. Simon (2006), *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, John Wiley & Sons, 552pp.

[14] S.T. Goh, O. Abdelkhalik, S.A.R. Zekavat (2013), "A weighted measurement fusion Kalman filter implementation for UAV navigation", *Aerospace Science and Technology*, **28(1)**, pp.315-323, DOI: 10.1016/j.ast.2012.11.012.

[15] A. Becker (2023), *Kalman Filter from The Ground Up*, Kalmanfilter.net, 426pp.

[16] G. Frison (2015), *Algorithms and Methods for Fast Model Predictive Control*, PhD Thesis, Technical University of Denmark.

[17] S.V. Raković, W.S. Levine (2019), *Handbook of Model Predictive Control*, Springer, DOI: 10.1007/978-3-319-77489-3.