

Article

Non-Linear Model Predictive Control Using CasADi Package for Trajectory Tracking of Quadrotor

Mohamed Elhesasy ¹, Tarek N. Dief ^{1,*}, Mohammed Atallah ¹, Mohamed Okasha ¹, Mohamed M. Kamra ¹, Shigeo Yoshida ^{2,3} and Mostafa A. Rushdi ^{2,*}

¹ College of Engineering, UAE University, Al-Ain P.O. Box 15551, United Arab Emirates

² Research Institute for Applied Mechanics (RIAM), Kyushu University, Fukuoka 816-8580, Japan

³ Institute of Ocean Energy (IOES), Saga University, Saga 840-8502, Japan

* Correspondence: tnchief@uaeu.ac.ae (T.N.D.); rushdimostafa@riam.kyushu-u.ac.jp (M.A.R.)

Abstract: In this paper, we present the development of a non-linear model predictive controller for the trajectory tracking of a quadrotor using the CasADi optimization framework. The non-linear dynamic model of the quadrotor was derived using Newton–Euler equations, and the control algorithm and drone dynamics were wrapped in Matlab. The proposed controller was tested by simulating the tracking of a 3D helical reference trajectory, and its efficiency was evaluated in terms of numerical performance and tracking accuracy. The results showed that the proposed controller leads to faster computational times, approximately 20 times faster than the Matlab toolbox (nlmpc), and provides better tracking accuracy than both the Matlab toolbox and classical PID controller. The robustness of the proposed control algorithm was also tested and verified under model uncertainties and external disturbances, demonstrating its ability to effectively eliminate tracking errors.

Keywords: model predictive control (MPC); non-linear MPC (NLMPC); CasADi; quadrotor; PID; Simulink



Citation: Elhesasy, M.; Dief, T.N.;

Atallah, M.; Okasha, M.; Kamra, M.M.; Yoshida, S.; Rushdi, M.A.

Non-Linear Model Predictive Control Using CasADi Package for Trajectory Tracking of Quadrotor. *Energies* **2023**, *16*, 2143. <https://doi.org/10.3390/en16052143>

Academic Editors: Eric Cheng and Junfeng Liu

Received: 5 January 2023

Revised: 8 February 2023

Accepted: 17 February 2023

Published: 22 February 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

1.1. Background and Motivations

Unmanned aerial vehicles (UAVs) have gained considerable interest in recent years from both academic and industrial researchers due to their wide variety of military and commercial applications such as surveillance, exploration areal inspection, and photography [1–5]. UAVs are classified into two main categories: fixed-wing and rotatory-wing UAVs, which offer significant advantages over fixed-wing UAVs since they can take off and land vertically, maintain their position at a fixed location in 3D space, have agility, and have high maneuverability. Quad-copter or quad-rotor, a multi-rotor aircraft with four connected propellers, one for each rotor, is one of the extremely popular designs for smaller UAVs due to their ability to be controlled in the three dimensions translational and rotational motions by adjusting the angular velocities of rotors. Vertical movement is achieved by simultaneously increasing or decreasing the angular velocities of the four rotors [6,7]. Lateral motions are performed by changing the speed of the left and right rotors, while Longitudinal motions are performed by adjusting the speed of the front and rear rotors. Yaw motion results from imbalances of the torque generated by each pair of propellers, for example, increasing the speed of the two clockwise-turning rotors, while decreasing the counter-clockwise rotors' speed creates a counter-clockwise motion, and vice-versa. As a result, Quad-rotor provides a challenging control problem because of the non-linearity feature of the system. Moreover, it is an under-actuated system [8] since it has six degrees of freedom, while it has just four actuators [9–11].

1.2. Related Works

Researchers have proposed a considerable amount of control techniques, while linear control approaches such as PID controllers [12–15] and LQR controllers [15] have been used widely, but they are valid in conditional flight zones. On the other hand, non-linear control methods can expand the controllable flight envelope and achieve better performance in all flight domains. For example, backstepping control [16,17], nonlinear H_∞ control [18], and sliding mode control (SMC) [19]. One of the promising control techniques used recently to control quadrotors is model predictive control (MPC) [20,21] due to its capability to compensate highly non-linear MIMO systems in addition to handling control constraints. MPC consists of two main components: the optimizer and the internal plant dynamic model, which predicts the next time step data. Moreover, the optimizer solves the optimization problem in order to select the optimum input, which ensures that the system output tracks the required reference values [22]. NMPC uses a non-linear internal plant model to predict the system's behavior, which requires high computational power in real-time applications. Many approaches have been developed to overcome this problem, such as the explicit MPC method [23–25]. This technique solves the optimization problem offline and stores the optimal parameters, rather than solving it online at each sampling time. Furthermore, the NLMPC can be implemented in real-time by using fast optimizers [26].

In this paper [27], an autonomous system designed to support Search and Rescue (SAR) activities in a small area by providing airborne surveillance, inspection, and assistance. The suggested technology is intended for use in challenging real-world conditions with extremely small tunnels that are barely wider than the airborne platform and with restricted sight owing to darkness and dust. The emphasis is on rapid deployment, high resilience, safety, and exact localization in an uncharted area without the installation of any other infrastructure, such as an external computer or localization system. The intended SAR situations have these as their primary needs.

In order to take into consideration system uncertainties and actuator defects, a fault-tolerant control approach for UAVs is suggested in this study [28,29]. When actuator problems arise, a mathematical model of the quadcopter UAV is initially introduced. The chattering issue and system uncertainties are addressed using a normal adaptive sliding mode control (NASMC) strategy that does not need knowledge of the upper bound. Radial basis function neural networks are linked with an adaptive scheme to make a rapid adjustment in the face of system uncertainties and actuator defects, which improves the performance of the NASMC scheme. The stability of the suggested approaches is examined using the Lyapunov theory.

NLMPC is another widely used strategy. It has been shown that NLMPC performs better than linear MPC, especially when managing nonlinear dynamics and uncertainty. This method is especially applicable to quad rotors because of their well-known complicated and nonlinear dynamics.

1.3. Contributions

We implemented an NLMPC tool for trajectory tracking applications using the optimization framework CasADi, which gives a considerable boost in computing speed after comparing its effectiveness with that of the MPC toolbox that is supplied by MATLAB. In this study, we explain how we did this. When compared with the performance of the MATLAB toolbox, CasADi's results were evaluated to assess how successful they were. Furthermore, CasADi significantly improves the accuracy of our findings regarding trajectory tracking. In addition, we suggested that a contrast be drawn between the efficiency of the MPC methods (both MPC and CasADi) and the PID control method. Significant findings were obtained through CasADi, and those results were included in the real-time prototype of the drone. Additionally, the real-time prototype of the drone had a fast sampling time and an adequate degree of accuracy.

2. Dynamic Model of the Quadrotor

Maintaining the Integrity of the Specifications

To derive the quadrotor's mathematical model, we considered the following assumptions:

- The quadrotor is a rigid body subject to one lift force and three torques.
- The structure of the quadrotor is symmetric with four rotors aligned with the x and y -axes. Therefore, the moment of the Inertia tensor only contains the diagonal elements.
- The center of gravity of the quadrotor and the origin of the body's fixed frame coincide.
- The gyroscopic effects and aerodynamic forces are neglected.

As shown in Figure 1, and derived in [19,30], we define an inertial frame on a particular location at ground level and a body fixed frame at the center of gravity of the quadrotor. The linear position of the quadrotor in the inertial frame (x, y, z) axes is given by the vector $\zeta = [x \ y \ z]^T$. Meanwhile, the orientation of the quadrotor vector $\eta = [\phi, \theta, \psi]$ is defined by the three Euler angles roll (ϕ), pitch (θ), and yaw (ψ), which determines rotations around x , y , and z axes, respectively.

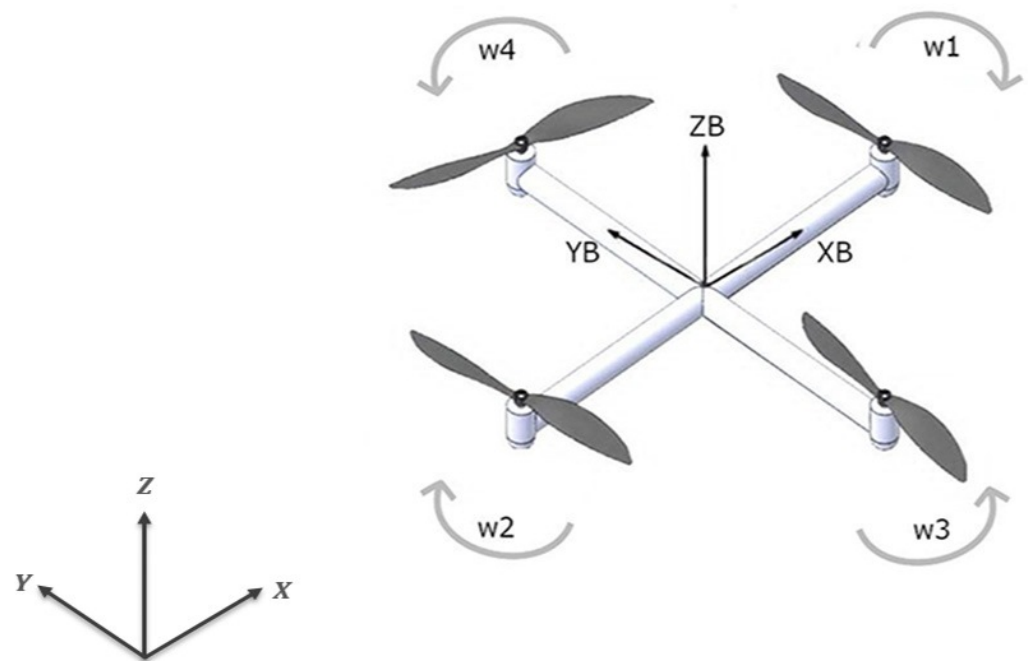


Figure 1. The inertial and body frames of the quadrotor.

The transformation between the body-fixed frame and the inertial frame is given by the rotation matrix R , which is obtained depending on the sequence of principle rotation: yaw, pitch, then roll:

$$R = \begin{pmatrix} C\psi C\theta & C\psi S\theta S\phi - S\psi C\phi & C\psi S\theta C\phi + S\psi S\phi \\ S\psi C\theta & S\psi S\theta S\phi + C\psi C\phi & S\psi S\theta C\phi - C\psi S\phi \\ -S\theta & C\theta S\phi & C\theta C\phi \end{pmatrix} \quad (1)$$

where C denotes $\cos$ and S denotes $\sin$.

The transformation between Euler angles rates in inertial frame $\dot{\eta} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]$ and the angular velocities measured in body frames $\Omega = [p, q, r]$ is defined as:

$$\Omega = W_{\eta} \dot{\eta} \quad (2)$$

where

$$W_\eta = \begin{pmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{pmatrix} \quad (3)$$

In this derivation, we approximate the non-linear model around the hovering equilibrium point by taking small roll and pitch angles assumptions, which reduces W_η to the Identity Matrix $I_{3 \times 3}$. As a result:

$$\Omega = \dot{\eta} \quad (4)$$

The Newton–Euler method is applied to describe the quadrotor dynamics. The translational equations of motion are derived in the inertial frame where the acceleration of the quadrotor is only affected by the gravitational force G and the total thrust of rotors T_B

$$m\ddot{\zeta} = G + RT_B \quad (5)$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + R \frac{T_B}{m} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (6)$$

In the body frame, the rotational motion is described in which the external torque applied to the quadrotor τ_B is equal to the angular acceleration of the inertia $I\dot{\Omega}$ and the centripetal forces $\Omega * I\Omega$.

$$\tau_B = I\dot{\Omega} + \Omega * I\Omega \quad (7)$$

The angular velocity of each rotor ω_i generates force f_i in the rotor axis direction, in addition, it generates torque M_i around its axis.

$$F_i = k\omega_i^2, \quad M_i = b\omega_i^2 \quad (8)$$

where k is the thrust constant and b is the drag factor.

The total thrust vector (T_B) applied in the body axes can be written by combining the forces of four rotors

$$T_B = \sum_{i=1}^4 |F_i| = k \sum_{i=1}^4 \omega_i^2 \quad (9)$$

while the total torque applied on the quadrotor (τ_B) about the x , y , and z -axes can be represented as follows

$$\tau_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(-\omega_2^2 + \omega_4^2) \\ lk(-\omega_1^2 + \omega_3^2) \\ b(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{bmatrix} \quad (10)$$

where l is the distance between the rotor and the center of mass of the quadrotor and τ_ϕ , τ_θ , τ_ψ denote the roll torque, pitch torque, and yaw torque, respectively.

Combining Equations (9) and (10), we can define the control input vector in terms of the rotors' angular velocities.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} T_B \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} k & k & k & k \\ 0 & -lk & 0 & lk \\ -lk & 0 & lk & 0 \\ b & b & b & b \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (11)$$

Now, we can represent the overall dynamic model of the quadrotor as follows:

$$\ddot{x} = (\cos\phi \cdot \sin\theta \cdot \cos\psi + \sin\phi \cdot \sin\psi) \cdot \frac{u_1}{m} \quad (12)$$

$$\ddot{y} = (\cos\phi \cdot \sin\theta \cdot \sin\psi - \sin\phi \cdot \cos\psi) \cdot \frac{u_1}{m} \quad (13)$$

$$\ddot{z} = -g + (\cos \phi \cdot \cos \theta) \cdot \frac{u_1}{m} \quad (14)$$

$$\ddot{\phi} = \dot{\theta} \dot{\psi} \left(\frac{I_y - I_z}{I_x} \right) + \frac{u_2}{I_x} \quad (15)$$

$$\ddot{\theta} = \dot{\phi} \dot{\psi} \left(\frac{I_z - I_x}{I_y} \right) + \frac{u_3}{I_y} \quad (16)$$

$$\ddot{\psi} = \dot{\theta} \dot{\phi} \left(\frac{I_x - I_y}{I_z} \right) + \frac{u_4}{I_z} \quad (17)$$

3. Control of Quad-Rotor Using Model Predictive Control

Model predictive control (MPC) has been widely used in recent decades due to its ability to control multi-input–multi-output (MIMO) systems, taking into consideration constraints on both the control inputs and the states of the system. MPC solves an optimization problem with a certain objective function over a prediction horizon to drive the output to the required reference. The output is a sequence of optimal control actions based on the current states and set of constraints. Using the dynamic model of the plant, the future behavior of the system is predicted and only the first control action is implemented, while the remaining samples are discarded, then the same steps are repeated by shifting the prediction horizon at each sampling instant. MPC has become increasingly utilized to control quadrotors due to its capability to handle the system's constraints through the optimization process [20,31,32].

3.1. MPC Mathematical Formulation

The objective of the control method [33] is to minimize the weighted cost function (18) by manipulating the optimization variables, which is the control sequence (u) using the optimal control problem (OCP), represented by Equation (21).

$$J_N(x, u) = \sum_{k=0}^{N-1} l((x_u(k), u(k))) + l_t(x_u(N), u(N)) \quad (18)$$

where,

$$l(x_u(k), u(k)) = \|x_u - x^r\|_Q^2 + \|u - u^r\|_R^2 \quad (19)$$

and $l_t(x_u(N), u(N))$ is the terminal cost that is defined as follows

$$l_t(x_u(N), u(N)) = \|x_u - x^r\|_{\tilde{Q}}^2 \quad (20)$$

These quadratic functions penalize the deviation between the predicted states, x_u , and the reference states, x_r , and it penalizes the difference between the control action, u , and its reference value u_r . The Q , R , and $\tilde{Q}$ are defined as the states, control, and terminal weights matrices, respectively. The optimal control problem of the MPC controller is then expressed as:

$$\begin{aligned} \min_u \quad & J_N(x, u) = \sum_{k=0}^{N-1} l((x_u(k), u(k))) + l_t(x_u(N), u(N)) \\ \text{s.t.} \quad & x_u(k+1) = f(x_u(k), u(k)) \\ & x_u(0) = x_0 \\ & u_{\min} \leq u_i \leq u_{\max} \end{aligned} \quad (21)$$

Algorithm 1 represents the approach utilized to control the quadrotor model with nonlinear MPC.

Algorithm 1 Non-linear MPC code.

- 1: Initialize the system with sampling time T_s , prediction horizon N and initial states.
- 2: Calculate the optimal control function over the prediction horizon.
- 3: Get the optimized control inputs.
- 4: Implement the first control sample to the non-linear model of the system.
- 5: Updates the system states.
- 6: Initialize the next iteration with the current inputs and states.
- 7: Repeats the same procedure from step 2 every sampling time T_s .

3.2. MPC Implementation

The quadrotor's dynamic model represented in Equations (12)–(17) can be represented in the general nonlinear system form as follows:

$$\dot{x}(t) = f(x(t), u(t)) \quad (22)$$

where, $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$, n, m are the numbers of states and control inputs, respectively.

The discrete-time of this system can be expressed as:

$$x(t+1) = F(x(t), u(t)) \quad (23)$$

where $x(t)$ is the states of the system $x(t) = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$, $u(t)$ denotes the control inputs $u(t) = (\omega_1^2, \omega_2^2, \omega_3^2, \omega_4^2)$, which represent the squared angular velocities of the four rotors. This non-linear model was used as the plant model provided to NLMPC to predict the output of the system. The quadrotor dynamics were discretized using the Runge–Kutta fourth order method. Since the system is under-actuated with six DOFs and only four control inputs, we can determine the states needed to be controlled through the weighting matrix Q and restrict the variation of other states through constraints that consider one of the major factors for the success of the MPC algorithm. In our design we assigned weights on the three positions $[x, y, z]$ and three Euler angles $[\phi, \theta, \psi]$. Furthermore, we applied weights to the control inputs through the R matrix but lower than the weight applied to the tracking states R , to keep the quadrotor operating around the hovering equilibrium point. In this work, the CasADi package and NLMPC toolbox in Matlab were used to stabilize the quadrotor.

3.2.1. CasADi Package

The presented work in this paper excluded alternative software options from their investigation and instead focused only on CasADi for a number of reasons. To begin with, CasADi offers a versatile and potent optimization framework created especially for nonlinear dynamic systems. This makes it the perfect solution for the complex, nonlinear system used in the publication's examination of trajectory tracking. As employing the MPC tools offered by MATLAB, computational performance is likewise substantially faster when compared to CasADi. This was shown in the research by comparing the two programs' performances, which demonstrated that CasADi offers faster solutions for the same optimization problem. Another factor is the committed user and development community that supports CasADi and works to continuously update and enhance it. This offers a degree of support and knowledge that may not be present in other packages. The authors chose to focus only on CasADi because it provides an optimization framework that is adaptable, effective, and well-supported for nonlinear dynamic systems. For their specific research and application, the authors feel that this framework is the best option.

CasADi is an open-source software platform for numerical optimization [34], which facilitates the solution of NLP's. The external solver Interior point Optimizer Package (IPOPT) is able to handle convex and non-convex optimization problems to obtain the optimum solution that satisfies the design requirements. We applied a direct multiple shooting technique to discretize the OCP (18) in which both the states and control inputs

are explicitly defined as decision variables. The dynamics of the quadrotor were discretized using three commonly used numerical methods: Euler, Heun, and Runge–Kutta fourth order. The computational time and tracking error were measured for each method and compared. The results show that while the difference in computational time between the methods is minimal (2.5%), RK4 exhibited the least tracking error among the three methods. For this reason, we have implemented the RK4 method for the discretization of the dynamics of the system.

3.2.2. Matlab Toolbox

Matlab provides this toolbox for designing, simulating, and prototyping the MPC controller. In our implementation, we used the non-linear MPC function (NLMPC). This function uses the default (fmincon) solver with the SQP algorithm to provide the manipulated variables and states. Algorithm 2 represents the approach to control the quadrotor model with nonlinear MPC.

Algorithm 2 Non-linear MPC implementation in Matlab toolbox.

- 1: Set number of states, inputs, and outputs.
 - 2: Define the prediction non-linear model state function.
 - 3: Provide sampling time, prediction horizon, and control horizon.
 - 4: Set constraints on the control inputs and states.
 - 5: Specify weights of outputs and manipulated variables.
 - 6: Set the reference states and the nominal control target.
 - 7: Simulate trajectory tracking.
-

3.3. PID Controller

PID controller is widely used to stabilize the quadrotors, so we developed a controller to evaluate the performance and stability of the MPC controller and compare their results. The following Figure 2, as described in [35], shows the control methodology implemented in the quadrotor system, which is categorized into two loops; the inner loop that has the attitude parameters and the outer loop that controls the position parameters of the quadrotor.

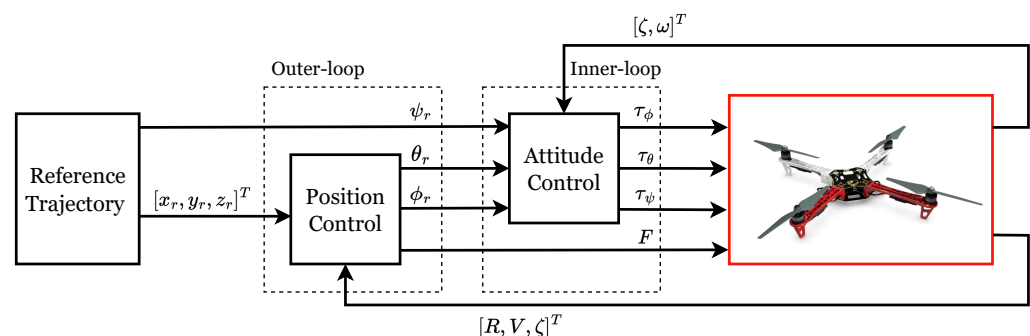


Figure 2. Control strategy for the PID technique.

This architecture to control the quadrotor needs to create and tune six parameters using the PID algorithm, one for each state, and all of them interact with each other, so in order to achieve this task, we used Simulink/Matlab via PID tuning to acquire the appropriate gains for each state.

The following Figure 3 illustrates the Simulink/Matlab model developed to test the response of the system.

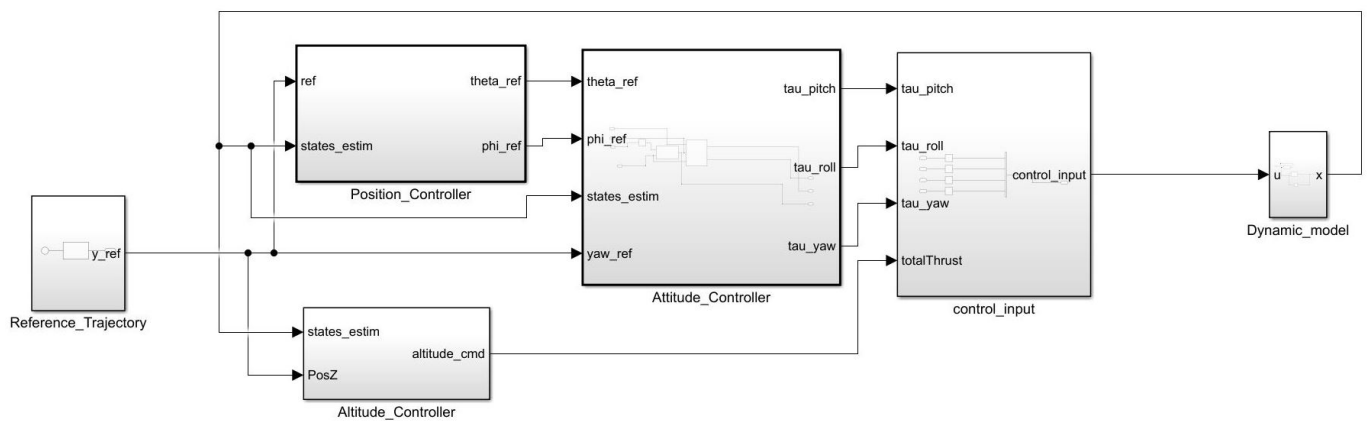


Figure 3. Simulink model for the quadrotor using PID controller.

4. Simulation Results and Discussion

The proposed controller is simulated in MATLAB to follow a 3D helical reference trajectory as follows:

$$x = 5\cos(t/2) - 5 \quad (24)$$

$$y = 5\sin(t/2) \quad (25)$$

$$z = t/2 \quad (26)$$

The parameters of the quadrotor used in the simulation are obtained from [36] and are presented in the following Table 1.

Table 1. Quad-Rotor parameters.

Description	Symbol	value
Inertia in x -axis	I_x	0.0213
Inertia in y -axis	I_y	0.02217
Inertia in z -axis	I_z	0.0282
Lift coefficient	k	4.0687×10^{-7}
Distance between rotor and center of mass	l	0.243
Quad-rotor mass	m	1.587
Drag coefficient	b	8.4367×10^{-9}

The constraints considered on the control input mainly relies on the limits of the motors' angular velocity. For the given quadrotor, the maximum angular velocity for each motor is $\omega_i = 4720$ rpm. From Equations (8) and (9), we can set the maximum control input limit and, by taking the angular velocity of each rotor at the hovering point, we can obtain the lower limit for the control input. The constraints applied to the states:

$$0 \leq T_B \leq 36.257 \quad (27)$$

$$-1.257 \leq \tau_\phi \leq 1.257 \quad (28)$$

$$-1.257 \leq \tau_\theta \leq 1.257 \quad (29)$$

$$-0.2145 \leq \tau_\psi \leq 0.2145 \quad (30)$$

The weighting matrices were tuned via trial and error to obtain the output close to the reference. The weighting matrix $Q = \text{Diag}[1, 1, 1, 0.6, 0.6, 1, 0, 0, 0, 0, 0]$, while the control input weighting matrix $R = \text{Diag}[0.3, 0.3, 0.3, 0.8]$. Furthermore, the sampling time of the controller was set to 0.1 s with a prediction horizon of 30 samples.

4.1. Results and Discussion

As depicted in Figures 4 and 5, the quadrotor position follows the reference trajectory in both the NLMPC package and CasADi toolbox, but the optimization time was about 0.02 s with the aid of CasADi software compared to 0.4 s for the NLMPC algorithm.

On the other hand, the PID controller showed less accurate results compared with the MPC controllers. Moreover, the mean Euclidean error value of the states was significantly smaller in the CasADi approach compared with the NLMPC results. The same helical trajectory was used to test the PID controller. It is clearly shown in Figures 4 and 5 that that PID controller has problems in tracking (x, y) trajectories, while it performs attitude and altitude stability perfectly.

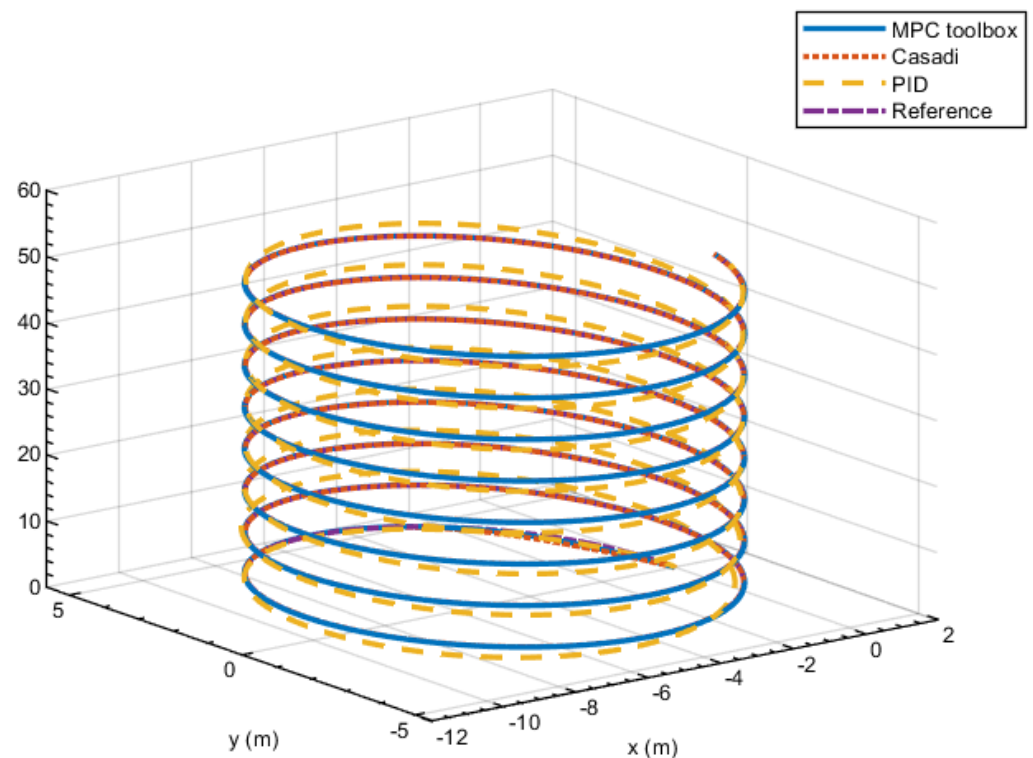


Figure 4. The 3-D plot of helical trajectory for the 3 controllers and the reference trajectory.

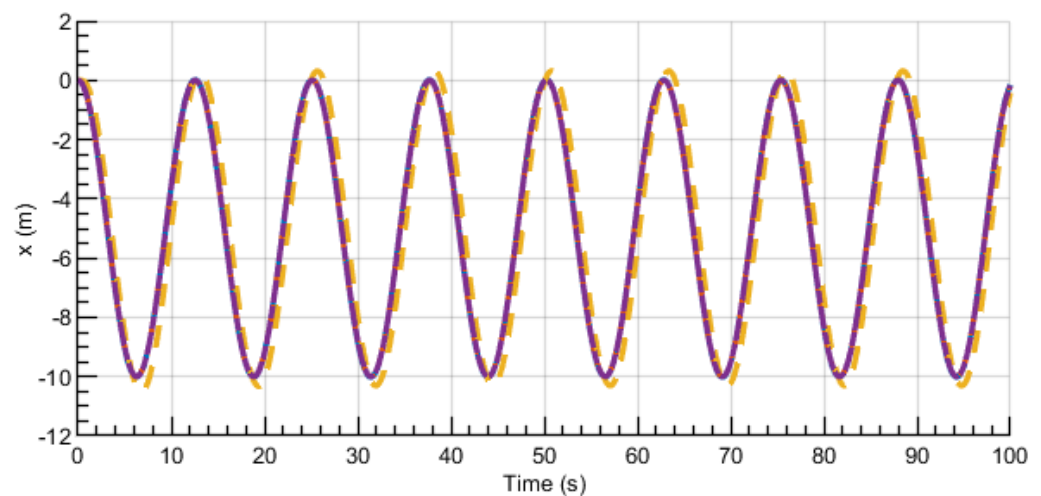


Figure 5. Cont.

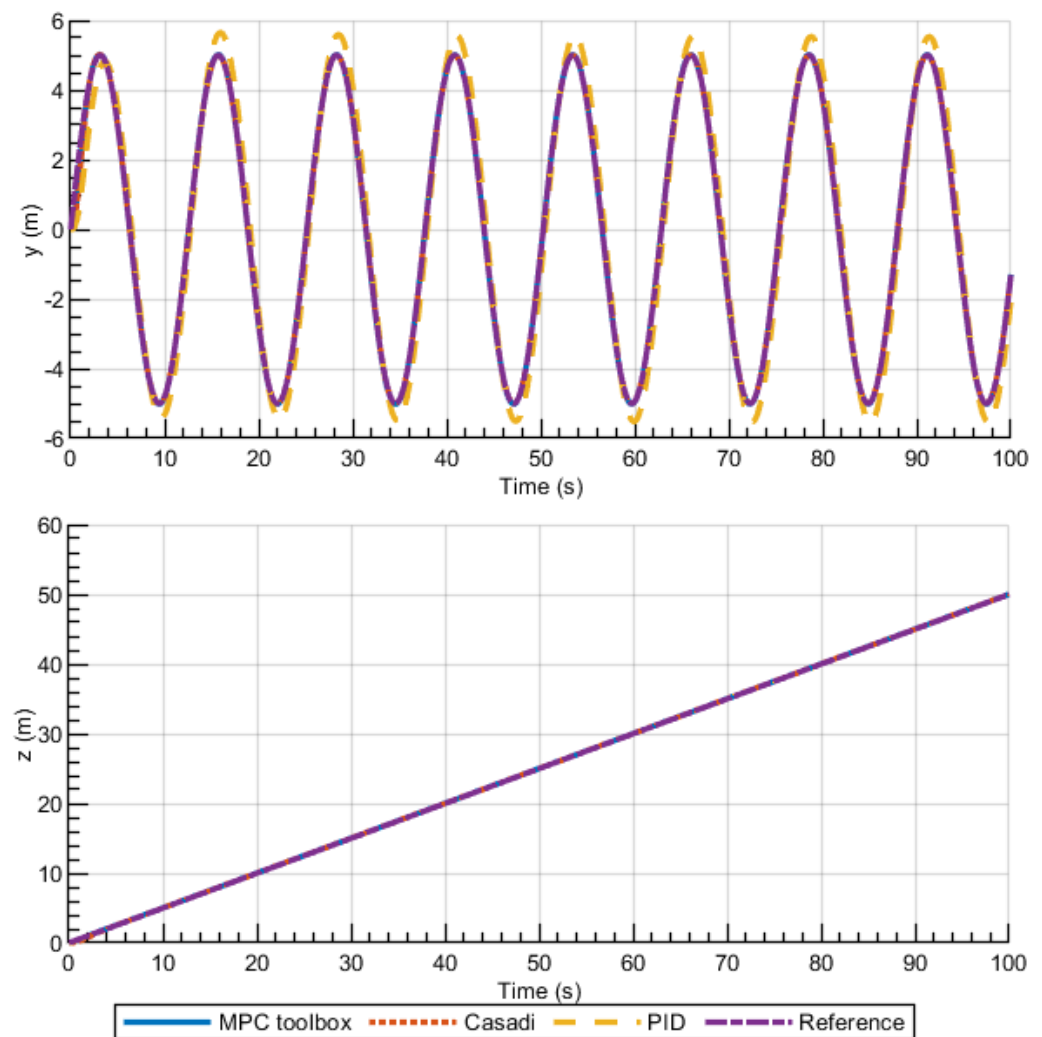


Figure 5. Time history of the translational states.

Figure 6 shows the time history for the attitude angles during flying in a helical path. The data clearly displays the difference in PID controller results compared with the two types of MPC controllers. This difference is considered as a delay/gap between the desired and the drone's exact angle/s. The difference happened in the rolling and pitch angles but in the yaw it acts as the other two MPC controllers without a significant gap.

Figure 7 shows the time history for the calculated controller actions for hovering commanded to the actuators. It indicates at the beginning of the simulation that the PID decays quite fast with higher fluctuations compared with the MPC controllers. The commanded path for the drone is helical, which is why the gap appeared clearly in roll and pitch angles but it usually does not appear in the step inputs as it is a constant value with the time that the system can settle at it after a short period.

Figure 8 depicts the time history for the rolling angle command to motors 2 and 4 in order to rotate around the x -axis. The control action shown in the figure is multiplied by the drag constants, that is why it is a very small number. The same situation happens for the pitch angle shown in Figure 9 but it commands motors 1 and 3 in order to rotate around the y -axis. On the other hand, Figure 10 has all responses near the same frequency but the PID controller response is very smooth compared with the other two controllers.

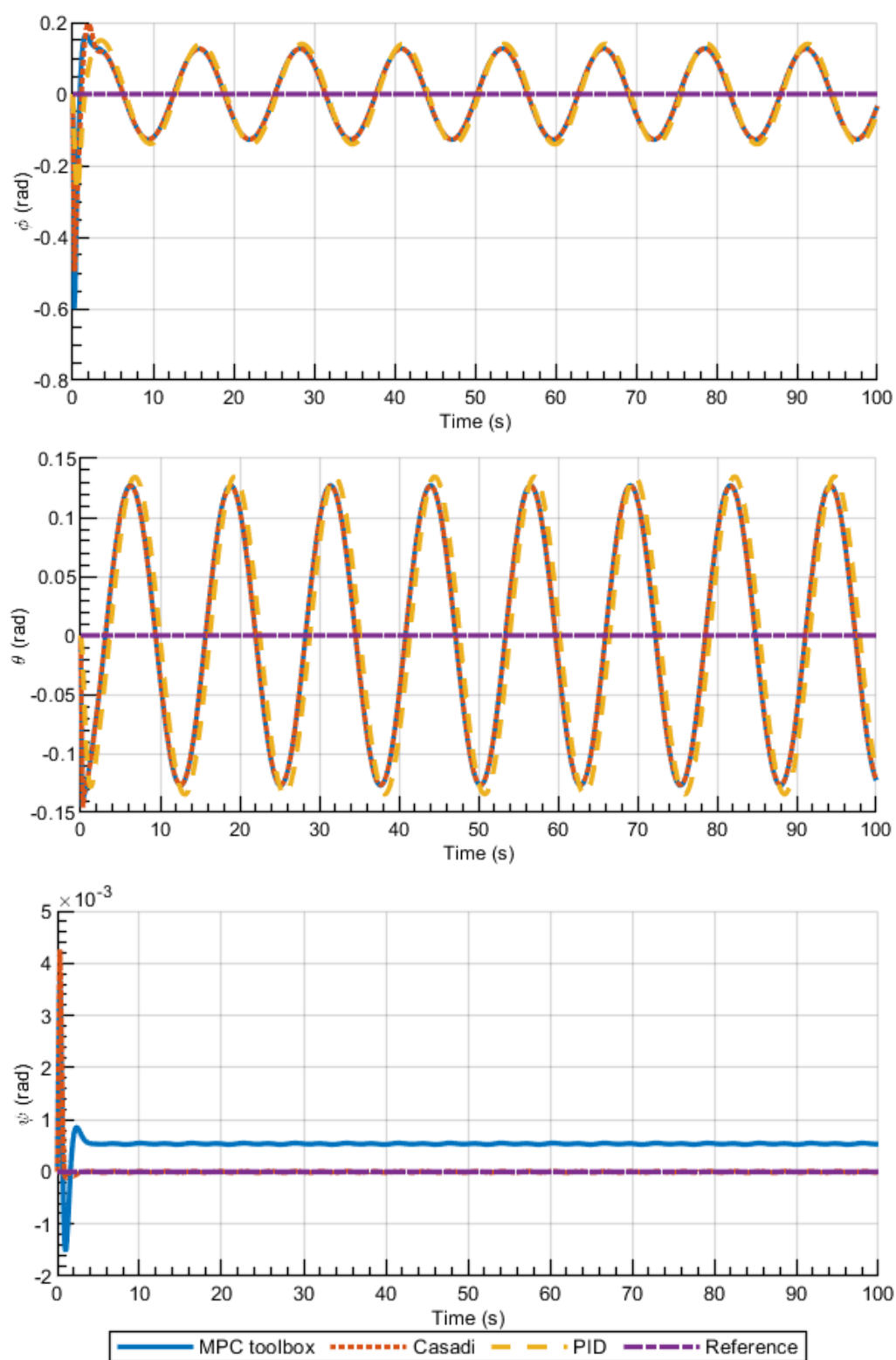


Figure 6. Time history of the rotational states.

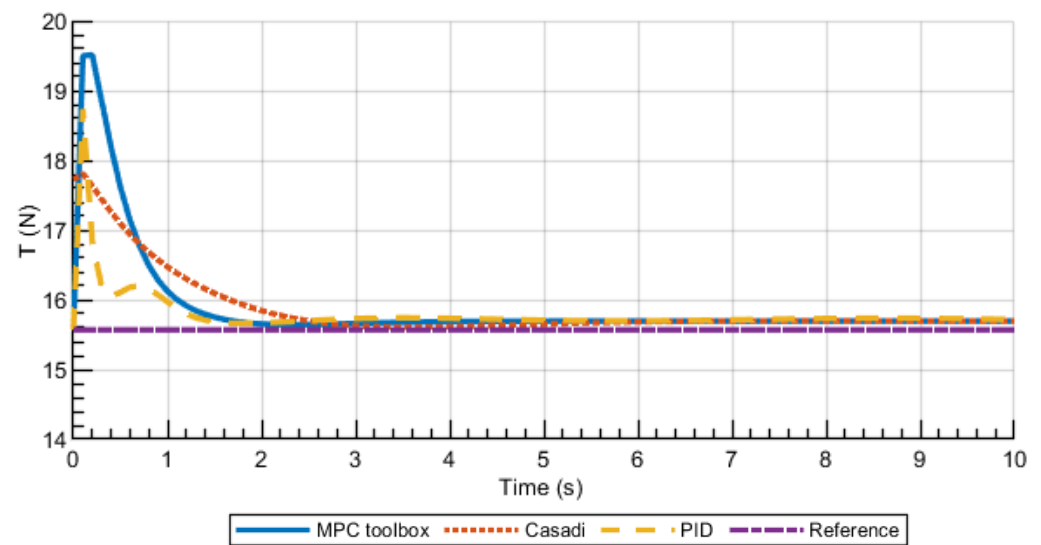


Figure 7. Time history of the actuator actions.

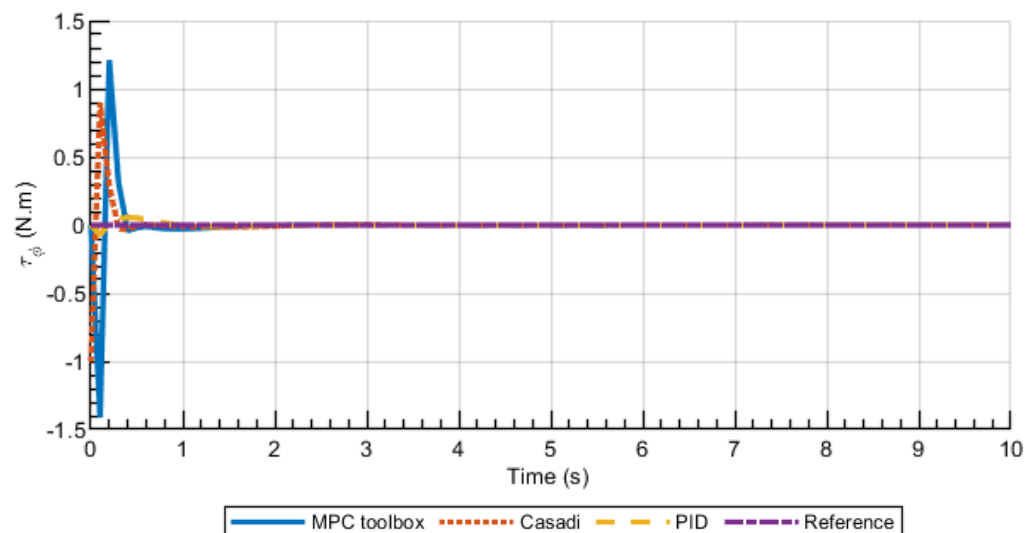


Figure 8. Time history of the roll control action.

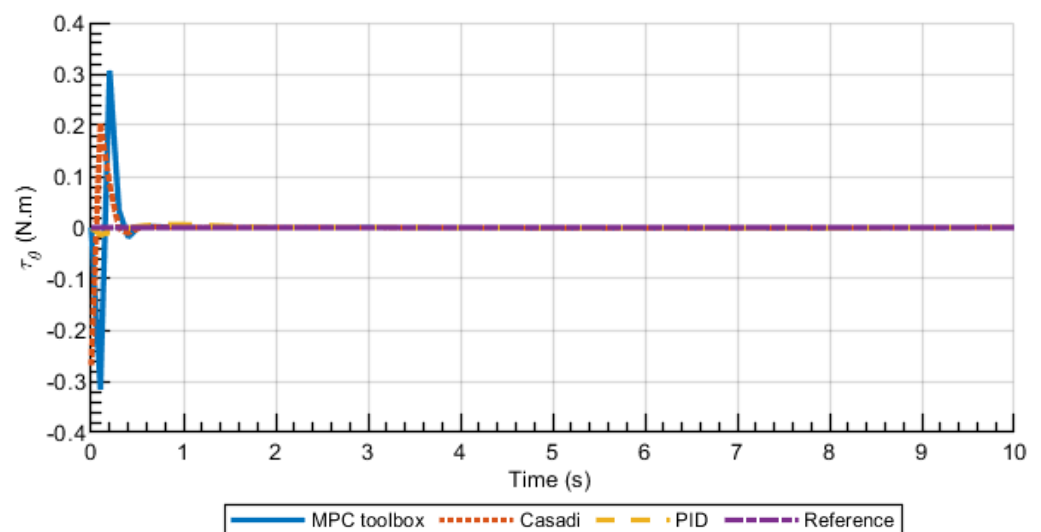


Figure 9. Time history of the pitch control action.

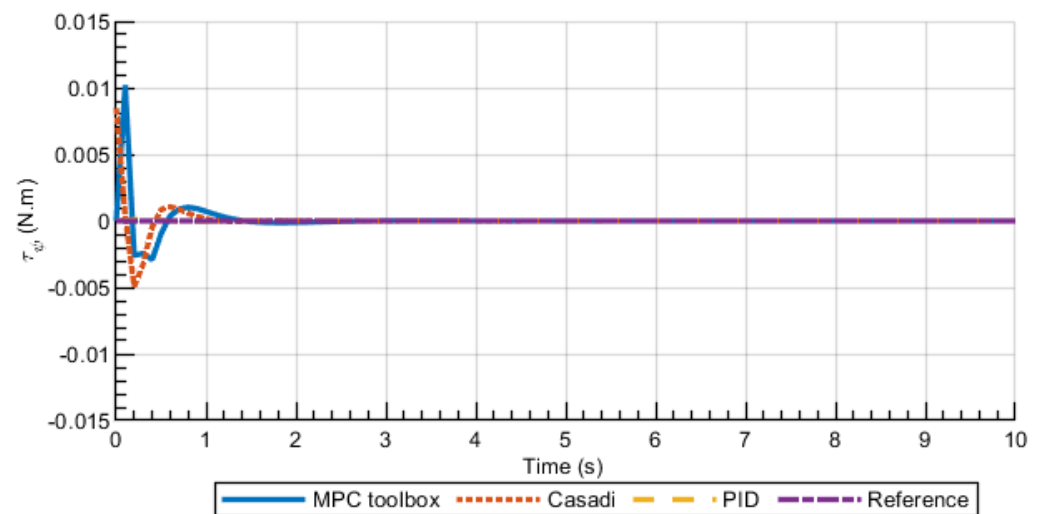


Figure 10. Time history of the yaw control action.

The most important result from the comparison between the three types of controllers is that CasADi control has better steady-state error compared with the NLMPC implemented in Simulink/Matlab. It was shown in the error analysis in Figure 11 where the NLMPC settled with high error (position) compared with the CasADi and PID controllers. Moreover, the PID controller had less error compared with NLMPC but with a high rate of fluctuations. Therefore, the concluded result from these results is that the CasADi controller is the best choice between the three different controllers, and also it can be easily implemented in a Microcontroller such as Raspberry-Pi with highly optimistic results.

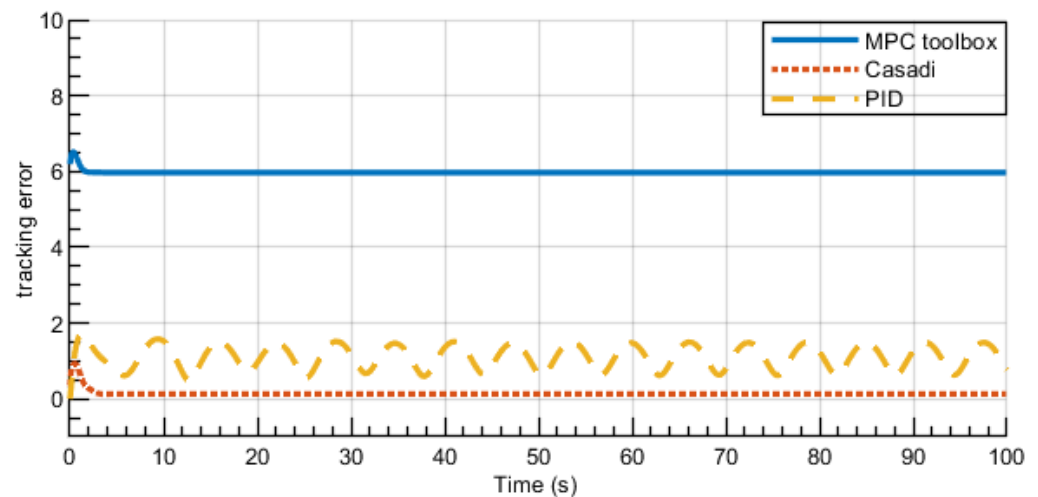


Figure 11. Time history of the tracking errors.

4.2. Robustness Analysis

The performance of the proposed tracking systems is tested and verified under model uncertainties and external disturbances. The mass and the moment of inertia are assumed to be uncertain with errors of zero mean and standard deviation as presented in Table 2. In addition, the dynamic system is subjected to external disturbances of zero mean and standard deviation values presented in Table 3.

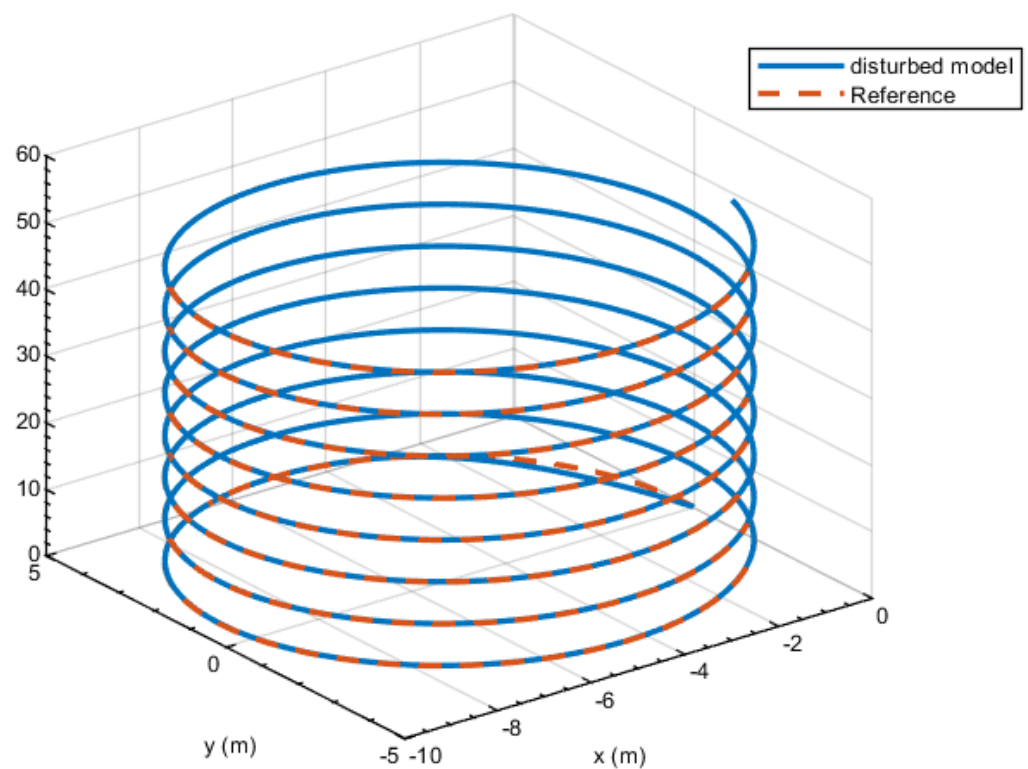
Table 2. Uncertainties of the inertial parameters.

Parameter	Value
σ_m	10^{-1}
σ_{I_x}	10^{-3}
σ_{I_y}	10^{-3}
σ_{I_z}	10^{-3}

Table 3. The standard deviation of the external disturbances.

Parameter	Value
σ_{T_B}	1
σ_{τ_ϕ}	10^{-1}
σ_{τ_θ}	10^{-1}
σ_{τ_ψ}	10^{-2}

Figure 12 shows that the reference trajectory is followed by the quadrotor in the presence of parameter uncertainties and external disturbance. As depicted in Figure 13, the robustness analysis shows that the state error remains bounded for all ranges of disturbance in Tables 2 and 3.

**Figure 12.** Trajectory tracking under model uncertainties and external disturbances.

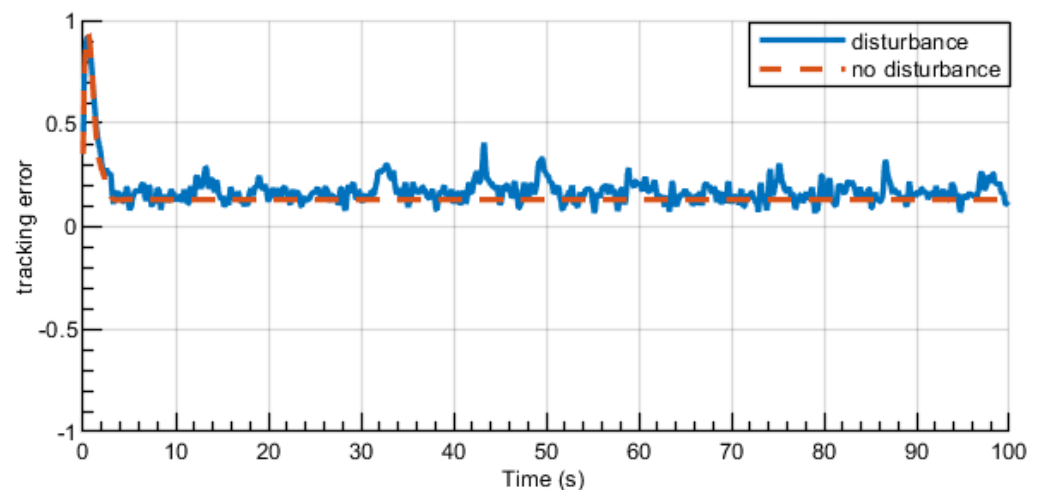


Figure 13. Tracking error under model uncertainties and external disturbances.

5. Conclusions

In the presented paper, we employed the CasADi algorithm in drone application as an MPC technique. We compared an existing NLMPC algorithm available in Matlab (not open source) and a free package (CasADi) beside the classical control in the application of the drone and found the following:

- The performance of CasADi was optimistic compared with the other two control types from the sampling time and system response point of view.
- The system error of the two MPC algorithms (NLMPC and CasADi) are very close to each other with higher accuracy than the PID controller.
- CasADi algorithm can run much faster than the NLMPC package in Matlab for the same accuracy.
- PID controller runs in a low sampling time compared with the NLMPC, however, its accuracy is very low and might lead to insatiability in noisy/windy conditions and might not achieve the trajectory defined for the flight tests.
- CasADi algorithm gives better steady-state error than the NLMPC package for position control.
- Our preliminary investigations highlighted the potential of the CasADi technique to be implemented in real-time for the small drone with suitable micro-controllers with a sampling time of less than 0.1 s. It will be the first time a drone flies with the CasADi algorithm with accepted flight performance.

The Matlab environment's integrated CasADi optimization framework is used by our control method. We can model and simulate the whole system, including the drone dynamics, using our novel technique without the requirement for co-simulation. A robust and adaptable environment for the design and implementation of sophisticated control algorithms for drone systems is made available by the combination of Matlab and CasADi. We can do thorough and effective simulations of the whole system since the drone dynamics and control algorithms are integrated inside the Matlab environment. This enables us to assess the performance of the control algorithm under a variety of operational settings and to see any possible problems or constraints before installation on the actual system. CasADi is a great option for real-time implementation on a drone system since it offers a considerable boost in computing speed compared to conventional MPC toolboxes. This is crucial for applications such as aerial navigation, obstacle avoidance, and autonomous flying that call for quick and precise control. Finally, the integration of Matlab with CasADi offers a strong foundation for the development of sophisticated control algorithms for drone systems. Our control technique makes the most of this integration, allowing us to smoothly model and simulate the whole system and assess its performance under a variety of operating situations.

As a future work, we plan to implement the CasADi algorithm in a small drone using a Raspberry-Pi Micro-controller in order to achieve the simulated results in the submitted paper.

Author Contributions: Conceptualization, M.O.; Formal analysis, M.A. and M.M.K.; Funding acquisition, S.Y. and M.A.R.; Methodology, M.E. and T.N.D.; Supervision, T.N.D. and S.Y.; Writing—original draft, M.E. and T.N.D.; Writing—review & editing, M.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by Kyushu University Operating Grant.

Data Availability Statement: The study did not report or use any experimental data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sarris, Z.; Atlas, S. Survey of UAV applications in civil markets. In Proceedings of the IEEE Mediterranean Conference on Control and Automation, Dubrovnik, Croatia, 27–29 June 2001; p. 11.
2. Cox, T.H.; Nagy, C.J.; Skoog, M.A.; Somers, I.A.; Warner, R. *Civil UAV Capability Assessment*; Technical Report, Draft Version; NASA: Washington, DC, USA, December 2004.
3. Benotsmane, R.; Vásárhelyi, J. Towards Optimization of Energy Consumption of Tello Quad-Rotor with Mpc Model Implementation. *Energies* **2022**, *15*, 9207. [\[CrossRef\]](#)
4. Roy, R.; Islam, M.; Sadman, N.; Mahmud, M.P.; Gupta, K.D.; Ahsan, M.M. A review on comparative remarks, performance evaluation and improvement strategies of quadrotor controllers. *Technologies* **2021**, *9*, 37. [\[CrossRef\]](#)
5. Yan, D.; Zhang, W.; Chen, H. Design of a Multi-Constraint Formation Controller Based on Improved MPC and Consensus for Quadrotors. *Aerospace* **2022**, *9*, 94. [\[CrossRef\]](#)
6. Dief, T.N.; Kamra, M.M.; Yoshida, S. Modeling, System Identification, and PID-A Controller for Tethered Unmanned Quad-Rotor Helicopter. *Int. Rev. Aerosp. Eng.* **2017**, *10*, 215–223. [\[CrossRef\]](#)
7. Dief, T.N.; Yoshida, S. System identification and adaptive control of mass-varying quad-rotor. *Evergreen* **2017**, *4*, 58–66. [\[CrossRef\]](#)
8. Hou, H.; Zhuang, J.; Xia, H.; Wang, G.; Yu, D. A simple controller of minisize quad-rotor vehicle. In Proceedings of the 2010 IEEE International Conference on Mechatronics and Automation, Xi'an, China, 4–7 August 2010; pp. 1701–1706.
9. Dief, T.N.; Yoshida, S. System identification for quad-rotor parameters using neural network. *Evergreen* **2016**, *3*, 6–11. [\[CrossRef\]](#)
10. Abdelkhalek, M.A.; El-Demerdash, M.S.; El-Tahan, A.A.; Dief, T.N. Attitude stability of quadcopter using classic control with angular acceleration. *Int. J. Comput. Sci. Inf. Technol. Secur.* **2015**, *5*, 325–331.
11. Deif, T.; Kassem, A.; El Baioumi, G. Modeling, Robustness, and Attitude Stabilization of Indoor Quad Rotor Using Fuzzy Logic Control. *Int. Rev. Aerosp. Eng.* **2014**, *7*, 197–201. [\[CrossRef\]](#)
12. Tayebi, A.; McGilvray, S. Attitude stabilization of a four-rotor aerial robot. In Proceedings of the 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601), Nassau, Bahamas, 14–17 December 2004; Volume 2, pp. 1216–1221.
13. Dikmen, İ.C.; Arisoy, A.; Temeltas, H. Attitude control of a quadrotor. In Proceedings of the 2009 4th International Conference on Recent Advances in Space Technologies, Istanbul, Turkey, 11–13 June 2009; pp. 722–727.
14. Zuo, Z. Trajectory tracking control design with command-filtered compensation for a quadrotor. *IET Control Theory Appl.* **2010**, *4*, 2343–2355. [\[CrossRef\]](#)
15. Bouabdallah, S.; Noth, A.; Siegwart, R. PID vs. LQ control techniques applied to an indoor micro quadrotor. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2451–2456.
16. Madani, T.; Benallegue, A. Backstepping control for a quadrotor helicopter. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 3255–3260.
17. Zemalache, K.M.; Beji, L.; Marref, H. Control of an under-actuated system: Application a four rotors rotorcraft. In Proceedings of the 2005 IEEE International Conference on Robotics and Biomimetics-ROBIO, Shatin, China, 5–9 July 2005; pp. 404–409.
18. Raffo, G.V.; Ortega, M.G.; Rubio, F.R. An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter. *Automatica* **2010**, *46*, 29–39. [\[CrossRef\]](#)
19. Bouabdallah, S.; Siegwart, R. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; pp. 2247–2252.
20. Raffo, G.V.; Ortega, M.G.; Rubio, F.R. MPC with nonlinear H_∞ Control Path Track. A Quad-Rotor Helicopter. *IFAC Proc. Vol.* **2008**, *41*, 8564–8569. [\[CrossRef\]](#)
21. Islam, M.; Okasha, M. A Comparative Study of PD, LQR and MPC on Quadrotor Using Quaternion Approach. In Proceedings of the 2019 7th International Conference on Mechatronics Engineering (ICOM), Putrajaya, Malaysia, 30–31 October 2019; pp. 1–6.
22. Allgöwer, F.; Zheng, A. *Nonlinear Model Predictive Control*; Birkhäuser: Basel, Switzerland, 2012; Volume 26.
23. Liu, C.; Lu, H.; Chen, W.H. An explicit MPC for quadrotor trajectory tracking. In Proceedings of the 2015 34th Chinese Control Conference (CCC), Hangzhou, China, 28–30 July 2015; pp. 4055–4060.

24. Okasha, M.; Kraleev, J.; Islam, M. Design and Experimental Comparison of PID, LQR and MPC Stabilizing Controllers for Parrot Mambo Mini-Drone. *Aerospace* **2022**, *9*, 298. [\[CrossRef\]](#)
25. Nan, F.; Sun, S.; Foehn, P.; Scaramuzza, D. Nonlinear MPC for Quadrotor Fault-Tolerant Control. *IEEE Robot. Autom. Lett.* **2022**, *7*, 5047–5054. [\[CrossRef\]](#)
26. Tajeddin, S. Automatic Code Generation of Real-Time Nonlinear Model Predictive Control for Plug-In Hybrid Electric Vehicle Intelligent Cruise Controllers. Master's Thesis, University of Waterloo, Waterloo, ON, Canada, 2016.
27. Petrlik, M.; Báča, T.; Heřt, D.; Vrba, M.; Krajník, T.; Saska, M. A robust UAV system for operations in a constrained environment. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2169–2176. [\[CrossRef\]](#)
28. Nguyen, N.P.; Hong, S.K. Fault-tolerant control of quadcopter UAVs using robust adaptive sliding mode approach. *Energies* **2018**, *12*, 95. [\[CrossRef\]](#)
29. Hermand, E.; Nguyen, T.W.; Hosseinzadeh, M.; Garone, E. Constrained control of UAVs in geofencing applications. In Proceedings of the 2018 26th Mediterranean Conference on Control and Automation (MED), Zadar, Croatia, 19–22 June 2018; pp. 217–222.
30. Luukkonen, T. Modelling and control of quadcopter Independent research project in applied mathematics. *Espoo* **2011**, *22*, 22.
31. Kamel, M.; Burri, M.; Siegwart, R. Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles. *IFAC-PapersOnLine* **2017**, *50*, 3463–3469. [\[CrossRef\]](#)
32. Bangura, M.; Mahony, R. Real-time model predictive control for quadrotors. *IFAC Proc. Vol.* **2014**, *47*, 11773–11780. [\[CrossRef\]](#)
33. Grüne, L.; Pannek, J. Nonlinear model predictive control. In *Nonlinear Model Predictive Control*; Springer: Cham, Switzerland, 2017; pp. 45–69.
34. Andersson, J.A.; Gillis, J.; Horn, G.; Rawlings, J.B.; Diehl, M. CasADi: A software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* **2019**, *11*, 1–36. [\[CrossRef\]](#)
35. Nicotra, M.M.; Naldi, R.; Garone, E. Nonlinear control of a tethered UAV: The taut cable case. *Automatica* **2017**, *78*, 174–184. [\[CrossRef\]](#)
36. Amadi, C.A. Design and Implementation of a Model Predictive Control on a Pixhawk Flight Controller. Ph.D. Thesis, Stellenbosch University, Stellenbosch, South Africa, 2018.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.