PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

University of Amar Telidji - Laghouat

Faculty of Technology

Department of Electronics

MASTER THESIS

DOMAINE: Science & Technology

OPTION: Automation & Industrial Computing

Makhloufi Hibat Erahmane & **Sahli Abdelnaceur**

Theme

MPC & ADRC Control for robots

Jury members:

Mohammed Benmiloud	MCB	President
Belkacem Rahmani	MCB	Examiner
Mohammed Belkhiri	Professor	Supervisor

2021 / 2022

Abstract

This project addresses the tracking control problem of a quadrotor with model uncertainties and external disturbances. First, the Newton-Euler approach is used to build the quadrotor mathematical model. Then, the MPC is applied successfully to the linearized decoupled subsystems of the drone based on two decentralization approaches. The high gain observer was used to estimate the quadrotor disturbances and integrated to the prediction model of MPC. Finally, simulation results show that the quadrotor can track the reference trajectory effectively even in the presence of disturbances.

Keywords: MPC, ESO, ADRC, Quadrotor Helicopter, Drone

Résumé

Ce projet aborde le problème de contrôle de suivi d'un quadrotor avec des incertitudes du modèle et des perturbations externes. Tout d'abord, l'apaproche Newton-Euler est utilisée pour construire le modèle mathématique quadrotor. Ensuite, le MPC est appliqué avec succès aux sous-systèmes découplés linéarisés du drone sur la base de deux approches de décentralisation. L'observateur à gain élevé a été utilisé pour estimer les perturbations du quadrotor et intégrés au modèle de prédiction de MPC. Enfin, les résultats de la simulation montrent que le quadrotor peut suivre efficacement la trajectoire de référence même en présence de perturbations.

Mots clés: MPC, ESO, ADRC, hélicoptère quadrirotor, drone

ملخص

Acknowledgements

First of all, I would like to thank God Almighty for giving us the opportunity and guidance to finish the final year project successfully. Without His blessing, it is hard for us to solve and face all problems during this project. I would like to acknowledge Pr. Mohammed Belkheiri. for his guidance and advice, and for his immense knowledge and patience.We also thank all of our teachers who contributed to our formation during our university career. I would also like to thank our family for their love, support, and encouragement. Not to forget our brothers and sisters who always encourage us to complete this project.

Away from work, special thanks go to our best friends Inass,Fatima, Nafisa, Houcen, Youcef, Jalil, Mani, and Abdelnour.

Contents

1 General Introduction **1**

1.1 Introduction 1

1.2 Objective 2

1.3 Thesis structure 2

2 Quadrotor Modeling **4**

2.1 Introduction 4

2.2 Theoretical concept of the quadrotor 4

2.3 Kinematic Model 6

2.3.1 Euler angles 6

2.4 Dynamic Model 7

2.4.1 Translational Equations of Motion 8

2.4.2 Rotational Equations of Motion 9

2.5 Aerodynamic Moment and Force 10

2.6 State Space Model 11

2.6.1 State vector 11

2.6.2 Input Vector 11

2.7 Linearization of Quadcopter Dynamics 13

2.7.1 Centralized Control 13

2.7.2 Linear model 14

2.8 Conclusion 16

3 Model Predictive Control **17**

3.1 Introduction 17

3.2 Model predictive control 17
3.2.1 Principle of Model Predictive control 17
3.2.2 Elements of predictive control 18
3.2.3 Predictive model 19
3.2.4 Optimal Control law 20
3.2.5 Constraints 22
3.3 Control applied in quadrotor (MPC) 23
3.3.1 The first decentralized control method 23
3.3.2 The second method for decentralized control 26
3.4 Simulation Results 29
3.4.1 Simulation 1 without disturbance 30
3.4.2 Simulation 2 without disturbance 31
3.4.3 Simulation results with disturbance 33
3.5 Disturbance estimator 34
3.5.1 Design a disturbance estimator 34
3.5.2 Simulation with disturbance 35
3.6 Conclusion 37

4 Active Disturbance Rejection Control 38
4.1 Introduction 38
4.2 Linear Active Disturbance Rejection Control 38
4.2.1 Extended State Observer- ESO 39
4.2.2 Control law 40
4.3 Application to Second-Order Linear ADRC 41
4.3.1 Second-Order Process 41
4.3.2 State Observer 42
4.3.3 Control structure 43
4.3.4 Closed loop dynamics 43
4.3.5 Observer dynamics 44
4.3.6 Simulation And Result LADRC 44
4.4 Discrete Time ADRC 46

4.4.1 Discretisation of the State Observer 46
4.5 Application to Second-Order Discrete ADRC 47
4.5.1 Second-Order Process 47
4.5.2 State Observe 47
4.5.3 Observer Dynamics 48
4.5.4 Simulation And Results Discrete ADRC 49
4.6 MPC With Discrete-Time Observer 50
4.7 Simulation 52
4.7.1 Simulation result 54
4.8 Conclusion 55

5 General Conclusion **56**

List of Tables

3.1 Constraints on the outer loop 24
3.2 Constraints on the inner loop 25
3.3 Constraints on the outer and inner loops 28
3.4 System parameters of quadrotor 29

List of Figures

2.1 Quadrotor configuration. 5
2.2 Quadrotor movement. 5
2.3 Quadrotor reference frame. 6

3.1 Principle of predictive control. 18
3.2 Predictive control strategy. 19
3.3 Control structure. 23
3.4 Structure of decentralized control 26
3.5 MPC MATLAB Simulink bloc diagram. 30
3.6 Simulation results with step reference 30
3.7 Simulation results with pseudo sinusoidal reference 31
3.8 MPC MATLAB Simulink bloc diagram 2 31
3.9 Simulation results with step reference 32
3.10 Simulation results with pseudo sinusoidal reference 32
3.11 Simulation results with pseudo sinusoidal reference 33
3.12 Circular trajectory in 3D 33
3.13 MPC MATLAB Simulink bloc diagram with estimator. 35
3.14 Simulation results with pseudo-sinusoidal reference 36
3.15 Simulation results of the disturbance. 36
3.16 Circular trajectory in 3D with estimator 37

4.1 Control loop structure with (ADRC) for a second-order process 43
4.2 Simulink bloc digram of LADRC with a 2nd order 44
4.3 (a) Actual and (b) Estimated outputs. 45

4.4 (a) Actual and (b) Estimated disturbances. 45
4.5 Simulink bloc digram of ADRC with a 2nd order 49
4.6 Step response and reaction on disturbance. 49
4.7 (a) Actual and (b) Estimated disturbances. 50
4.8 Schematic representation of the closed-loop system. 51
4.9 MPC MATLAB Simulink bloc diagram with Discrete-Time Observer . . 53
4.10 Illustration Figure for Discrete-Time Observer 53
4.11 Simulation results with pseudo-sinusoidal reference. 54
4.12 Simulation results of the disturbance. 54
4.13 Circular trajectory in 3D with ADRC 55

Abbreviations

UAV : Unmanned Aerial Vehicle

MIMO : Multi Input Multi Output

SISO : Single Input Single Output

MPC : Model Predictive Control

ADRC : Active Disturbance Rejection Control

LADRC : Linear Active Disturbance Rejection Control

ESO : Extended State Observer

LESO : Linear Extended State Observer

PID : Proportional integral derivative controller

LQR : Linear Quadratic Regulator

DOF : Degrees Of Freedom

Chapter 1

General Introduction

1.1 Introduction

For a few years, the development of flying platforms has not stopped growing due to the remarkable progress in the field of embedded systems such as the miniaturization of sensors, actuators, and the evolution of digital micro-controllers. These flying platforms are known as drones or UAVs (Unmanned Aerial Vehicles), among the rotary wing UAVs, the Quadrotor, which is the subject of this thesis, stands out as one of the most promising systems because of the diversity of applications for which it can be used: surveillance and observation missions, aerial photography, tracking, spying, monitoring the condition of a building that is difficult to access or even transporting goods. The quadcopter is a system with significant nonlinearities and fast dynamics [1]. Although employing a linearized model or a combination of linearized models to approximate the nonlinear dynamics of a quadcopter can achieve some control objectives, we must accept some performance reduction. As a result of this, we propose using the constrained MPC technique to control the quadcopter directly . In this case, we may not only achieve trajectory tracking while also utilizing MPC natural capacity, namely, explicit constraint management, which is extremely useful when dealing with constraints on states, inputs, and outputs and the reason behind that is to make small angle variations to keep the validity of the linearized model of the nonlinear system. Furthermore, when it comes to constraint-handling capabilities, MPC is very easy to apply for fault tolerance in actuators (i.e., quadcopter motors) by modifying its input

constraints, which is of crucial importance to many works. Among all the attitude control techniques, the active disturbance rejection control (ADRC) method has proven to be effective in handling uncertainties performance and does not require an accurate mathematical model of the system to be controlled. The tracking differentiator, the extended-state-observer (ESO), and the nonlinear feedback controller are the three parts of the ADRC in general [2]. Because the ESO includes an inherent decoupling function, it is an excellent approach for controlling quadrotor attitude systems. Recently, few important research publications have used ADRC to control the attitude of quadrotors.

1.2 Objective

The main contribution of this project is to propose the model predictive control (MPC) technique for controlling the drone trajectory tracking in the presence of constraints and disturbances, which we will estimate in two ways. First, the disturbance estimator is used, followed by the ADRC. The MPC strategy is used, which is a control algorithm to achieve optimal and predictive control by solving an optimization problem with the 'Yalmip' toolbox. As a result, simulation results by applying the proposed MPC strategy to a quadcopter simulator implemented in MATLAB environment are shown in order to verify the effectiveness of the proposed MPC strategy.

1.3 Thesis structure

This master thesis is organized as follows:

Chapter 1 : Introduction

In this chapter, an explanation of the introduction of the project, objectives of this master's thesis and its organization are provided.

Chpater 2 : Quadrotor Modeling

This chapter shows the mathematical model of the quadcopter system using the Newton-Euler formalism, it gives the details about the kinematic and dynamic model, and finally the nonlinear state-space model is derived.

Chapter 3: Model Predictive Control

In this chapter, the MPC control theory is presented. It covers model predictive control theory without and with constraints and disturbance and its implementation for the quadrotor system is given.

Chapter 4 : Active Disturbance Rejection Control

This chapter, explains the ADRC with an extended linear state observer and a discrete-time observer with a second-order example . Then it exploits in more details the discrete observer for disturbances estimation in the quadrotor MPC controller.

Chapter 5 : Conclusion

This chapter shows the conclusions and future work for the thesis.

Chapter 2

Quadrotor Modeling

2.1 Introduction

The ordinary quadcopter is a robot designed with a symmetrical structure. It consists of four rotors fixed at an equal distance from the central body. The quadcopter's design makes it a highly maneuverable vehicle that can achieve a wide range of missions and tasks. [3]. In this chapter, the dynamic model of the drone will be derived.

2.2 Theoretical concept of the quadrotor

The quadrotor or quadcopter is a flying machine that has four motors(M1, M2, M3, M4), to which are fixed with four propellers Fig.2.1 mounted symmetrically on the crossbeam, and separated into two groups rotating in opposite directions (One pair rotates clockwise, while the other rotates counter clockwise) in order to balance the torques.

The quadrotor can conduct complicated tasks such as maneuvering, mapping, and navigation, etc. Since the quadrotor has 4 actuators with 6 degrees of freedom (DOF), it is an under actuated system. The system dynamics for the linear motions and the angular rotations of the quadrotor are coupled.

The motors M1, M3 rotate in counterclockwise direction with speed ω_1,ω_3 respectively. Like-wise, the motors M2, M4 rotate in clockwise direction with speeds ω_2, ω_4 respectively.

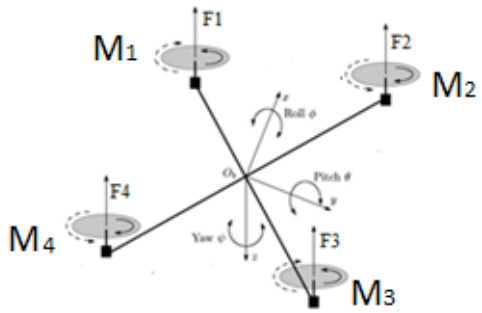


Figure 2.1: Quadrotor configuration.

The three orientation movements, roll, pitch, and yaw (ϕ,θ,ψ), can be achieved by the speeds difference of the four motors.

The yaw movement is obtained from the counter torque between each of the propellers. While each rotor rotates at an equal angular velocity, the net yaw is zero, but the velocities difference between the two pairs creates a positive or negative yaw.

Forward or backward motion which is related to the pitch, θ angle can be obtained by increasing the back rotor thrust and decreasing the front rotor thrust. Finally, a sideways motion which is related to the roll, ϕ angle can be achieved by increasing the left rotor thrust and decreasing the right rotor thrust. Fig.2.2 shows the various movements of a quadrotor due to changes in rotor speeds.

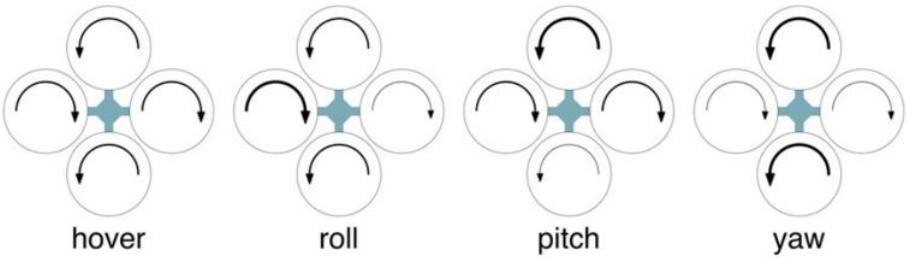


Figure 2.2: Quadrotor movement.

2.3 Kinematic Model

To set up the model of the quad-copter, the reference coordinate frames should be defined. The earth inertial references frame is with N, E, and D axes, and A, B, and C body frame is with x, y, and z axes. As shown in Fig.2.3, the inertial reference is fixed on a specific place and it uses the N, E, D notation which represents North, East, and Downwards respectively. For the body frame, the origin located in the center of the quad-copter body with the x-axis pointing toward propeller 1, the y-axis pointing towards propeller 2, and the z-axis is pointing to the ground [4].

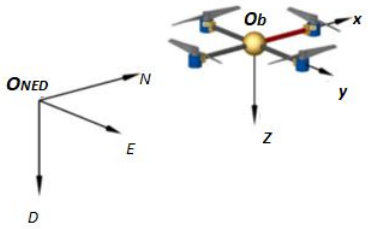


Figure 2.3: Quadrotor reference frame.

2.3.1 Euler angles

The three Euler angles ϕ , θ, and ψ represent the orientation of the quadcopter, where ϕ is the roll angle about the x-axis,θ is the pitch angle about the y-axis, and ψ is the yaw angle about the z-axis. The Euler angles are important for the definition of the transformation matrix from different reference frames. The transformation matrix is formed by a sequence of three plane rotations, which are named R_{yaw}, R_{pitch}, and R_{roll}.

To describe the orientation of the quad-copter can be used the rotation matrices. We then have the following rotation matrices

$$R_{roll} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.1}$$

$$R_{pitch} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.2}$$

$$R_{yaw} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

From equations 2.1, 2.2 and 2.3, we can easily find the matrix of rotation R_{B2W} from the body frame "B" to the inertial frame "W" :

$$R_{B2W}(\psi, \theta, \phi) = R_{yaw}.R_{pitch}.R_{roll} \tag{2.4}$$

$$R_{B2w} = \begin{bmatrix} c(\theta).c(\psi) & c(\psi).s(\theta).s(\phi) - c(\phi).s(\psi) & s(\phi).s(\psi) - c(\phi).c(\psi).s(\theta) \\ c(\theta).s(\psi) & c(\phi).s(\psi) + s(\theta).s(\phi).s(\psi) & c(\phi).s(\theta).s(\psi) - c(\psi).s(\phi) \\ -s(\theta) & c(\theta).s(\phi) & c(\theta).c(\phi) \end{bmatrix} \tag{2.5}$$

where c and s denote "cos" and "sin" respectively.

2.4 Dynamic Model

The motion of the quadcopter can be divided into two subsystems, translational motion (x,y positions, and altitude), and rotational motion (roll, pitch, and yaw).Based on the Newton-Euler formalism, the forces and moments acting on the quadcopter will be investigated. The twelve state variables are, three translational positions, their corresponding velocities, and three angular positions and their corresponding velocities. The position vector in an earth-fixed inertial coordinate system is indicated as $P = [x, y, z]^T$, while the velocity rate vector is $\dot{P} = [\dot{x}, \dot{y}, \dot{z}]^T$. The roll, pitch, and yaw angles are indicated as $\sigma = [\phi, \theta, \psi]^T$, and their derivatives are the body angular rates $\omega = [p, q, r]^T$.

2.4.1 Translational Equations of Motion

Based on the assumption that the quadrotor is made of a rigid symmetrical structure with a diagonal inertia matrix I and in the absence of aerodynamic external disturbances, according to Newton's second law.

$$\vec{F_\omega} = \frac{d}{dt}(m.\vec{v}) \tag{2.6}$$

Where F_ω=G - F , G is the gravity and F is the thrust generated from rotors. Note that the F is described in the body frame, and must be transformed from body frame to the inertial frame by following equations 2.6 . Then, the dynamic of the quadrotor can be written as:

$$\dot{v} = \frac{1}{m}\left[\begin{bmatrix}0\\0\\mg\end{bmatrix} - R_{B2W}.\begin{bmatrix}0\\0\\F_z\end{bmatrix}\right] \tag{2.7}$$

In this equation, substituting R_{B2W} gives an expression for the derivative of velocity rate in the inertial frame.

$$\dot{v} = \frac{1}{m}\begin{bmatrix}-F_z(c(\phi).s(\theta).c(\psi) + s(\phi).s(\psi))\\-F_z(c(\phi).s(\theta).s(\psi) + s(\phi).c(\psi))\\mg - F_z(c(\phi).c(\theta))\end{bmatrix} \tag{2.8}$$

This vector equation can be written in component form as

$$\begin{cases}\ddot{x} = -\frac{1}{m}F_z(c(\phi).s(\theta).c(\psi) + s(\phi).s(\psi))\\\ddot{y} = -\frac{1}{m}F_z(c(\phi).s(\theta).s(\psi) + s(\phi).c(\psi))\\\ddot{z} = -\frac{1}{m}F_z(c(\phi).c(\theta)) + g\end{cases} \tag{2.9}$$

2.4.2 Rotational Equations of Motion

In the body frame the total torque applied to the quadrotor is expressed as :

$$\vec{M_b} = \vec{I}.\dot{\vec{\omega}} + \vec{\omega} \times (\vec{I}.\vec{\omega}) \tag{2.10}$$

where $M_b = [\tau_x \tau_y \tau_z]^T \in R^3$, is the total torque and I is the diagonal inertia matrix:

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \in R^{3\times 3} \tag{2.11}$$

where the right-hand side is a vector of the applied torques:

$$\vec{\omega} \times (\vec{I}.\vec{\omega}) = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_x p \\ I_y q \\ I_z r \end{bmatrix} = \begin{bmatrix} (I_z - I_y)qr \\ (I_x - I_z)rp \\ (I_y - I_x)pq \end{bmatrix} \tag{2.12}$$

substituting this into equation 2.12 yields:

$$\begin{cases} \tau_x = I_x \dot{p} + (I_z - I_y)qr \\ \tau_y = I_y \dot{q} + (I_x - I_z)rp \\ \tau_z = I_z \dot{r} + (I_y - I_x)pq \end{cases} \tag{2.13}$$

These equations can be solved for the derivatives of the angular rates to obtain:

$$\begin{cases} \dot{p} = \frac{\tau_x + I_y qr - I_z qr}{I_x} \\ \dot{q} = \frac{\tau_y + I_z rp - I_x rp}{I_y} \\ \dot{r} = \frac{\tau_z + I_x pq - I_y pq}{I_z} \end{cases} \tag{2.14}$$

Now a simplification is made by setting $[\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T = [p \quad q \quad r]^T$. This assumption holds true for small angles of movement [5] . So, the dynamic model of the quad-rotor in the inertial frame is:

$$\begin{cases} \ddot{\phi} = \frac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} + \frac{\tau_x}{I_x} \\ \ddot{\theta} = \frac{I_z - I_x}{I_y}\dot{\phi}\dot{\psi} + \frac{\tau_y}{I_y} \\ \ddot{\psi} = \frac{I_x - I_y}{I_z}\dot{\phi}\dot{\theta} + \frac{\tau_z}{I_z} \end{cases} \tag{2.15}$$

2.5 Aerodynamic Moment and Force

The aerodynamic forces and moments depend on the shape and dimensions of a body, the speed of its translational motion, its orientation to the direction of the velocity, the properties and the state of the medium in which the motion takes place, and in some cases on the angular rotational velocities and the acceleration of the body's motion [6].

- The thrust force is generated from rotors is proportional to the square of the motor speed:

$$F_i = K_f.\omega_i^2 \tag{2.16}$$

- The rotors also generate the aerodynamic moment, which is proportional to the square of the rotor speed:

$$\tau_i = K_m.\omega_i^2 \tag{2.17}$$

Where K_f is the thrust factor ,K_m is the drag factor and ω^2 the squares of the motor speed driving the four propellers, i = 1, 2, 3, 4.

- The thrust force vector F is expressed by the superposition of the command force:

$$\begin{cases} F_x = 0 \\ F_y = 0 \\ F_z = -F_1 - F_2 - F_3 - F_4 \end{cases} \tag{2.18}$$

$$F_z = -K_f(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \tag{2.19}$$

- We define l as the arm length of each of the quadrotor's arms, F_2 multiplied by l generates a negative moment about the y-axis, while in the same manner, F_4 generates a positive moment. The total moment about the x-axis is:

$$\begin{cases} \tau_x = l(F_2 - F_4) \\ \quad\quad = k_f l(\omega_2^2 - \omega_4^2) \end{cases} \tag{2.20}$$

- For the moments about the body frame's y-axis, the thrust of rotor 1 generates a positive moment, while the thrust of rotor 3 generates a negative moment. The

total moment about the y-axis is:

$$\begin{cases} \tau_y = l(F_1 - F_3) \\ \quad = k_f l(\omega_1^2 - \omega_3^2) \end{cases} \tag{2.21}$$

- The moment about the body frame's z-axis is caused by the rotors' rotation from aerodynamic moment, while the thrust force has no effect on the moment for z-axis [4]. The total moment about the z-axis is:

$$\begin{cases} \tau_z = \tau_1 - \tau_2 + \tau_3 - \tau_4 \\ \quad = k_m(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{2.22}$$

2.6 State Space Model

2.6.1 State vector

Organizing the state's vector in the following way:

$$x = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T = [x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ \phi_1\ \theta_1\ \psi_1\ \phi_2\ \theta_2\ \psi_2]^T \tag{2.23}$$

2.6.2 Input Vector

The input vector U consisting of four inputs, is defined as:

$$U = [U_1\ U_2\ U_3\ U_4]^T = [F_z\ \tau_x\ \tau_y\ \tau_z]^T \tag{2.24}$$

To put the equations of motion into state space form, we introduce the state variables

$$\begin{gathered} x_1 = x,\ x_2 = \dot{x} \\ y_1 = y,\ y_2 = \dot{y} \\ z_1 = z,\ z_2 = \dot{z} \\ \phi_1 = \phi,\ \phi_2 = \dot{\phi} \\ \theta_1 = \theta,\ \theta_2 = \dot{\theta} \\ \psi_1 = \psi,\ \psi_2 = \dot{\psi} \end{gathered} \tag{2.25}$$

it is possible to rewrite the equations of the dynamics of the quadrotor in the state-space from 2.9 and 2.15 ,2.25 :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{y}_1 = y_2 \\ \dot{z}_1 = z_2 \\ \dot{x}_2 = -\frac{1}{m}F_z(c(\phi).s(\theta).c(\psi) + s(\phi).s(\psi)) \\ \dot{y}_2 = -\frac{1}{m}F_z(c(\phi).s(\theta).s(\psi) + s(\phi).c(\psi)) \\ \dot{z}_2 = -\frac{1}{m}F_z(c(\phi).c(\theta)) + g \\ \dot{\phi}_1 = \phi_2 \\ \dot{\theta}_1 = \theta_2 \\ \dot{\psi}_1 = \psi_2 \\ \dot{\phi}_2 = \frac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} + \frac{\tau_x}{I_x} \\ \dot{\theta}_2 = \frac{I_z - I_x}{I_y}\dot{\phi}\dot{\psi} + \frac{\tau_y}{I_y} \\ \dot{\psi}_2 = \frac{I_x - I_y}{I_z}\dot{\phi}\dot{\theta} + \frac{\tau_z}{I_z} \end{cases} \tag{2.26}$$

2.7 Linearization of Quadcopter Dynamics

2.7.1 Centralized Control

The dynamics of the whole system in vector form is given by:

$$\begin{aligned} \dot{X} &= F(x) + G(x, \vec{\omega})U \\ Y &= Cx \end{aligned} \tag{2.27}$$

Where :

$$F(x) = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 0 \\ 0 \\ g \\ \phi_2 + \theta_2[s(\phi_1)t(\theta_1)] + \psi_2[c(\phi_1)t(\theta_1)] \\ \theta_2 c(\phi_1) - \psi_2 s(\phi_1) \\ \theta_2[\frac{s(\phi_1)}{c(\theta_1)}] + \psi_2[\frac{c(\phi_1)}{c(\theta_1)}] \\ [\frac{I_y - I_z}{I_x}]\theta_2\psi_2 \\ [\frac{I_z - I_x}{I_y}]\phi_2\psi_2 \\ [\frac{I_x - I_y}{I_z}]\theta_2\phi_2 \end{bmatrix} \tag{2.28}$$

$$G(x) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m}(c(\phi_1).s(\theta_1).c(\psi_1) + s(\phi_1).s(\psi_1)) & 0 & 0 & 0 \\ -\frac{1}{m}(c(\phi_1).s(\theta_1).s(\psi_1) + s(\phi_1).c(\psi_1)) & 0 & 0 & 0 \\ -\frac{1}{m}(c(\phi_1).c(\theta_1)) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & I_x & 0 & 0 \\ 0 & 0 & I_y & 0 \\ 0 & 0 & 0 & I_z \end{bmatrix} \tag{2.29}$$

2.7.2 Linear model

We now are interested in linearizing the nonlinear dynamics of the quadcopter given in 2.26 about its equilibrium points. To do this, we must first find the equilibrium points of the quadcopter dynamics.

Equilibrium point $(\bar{x}, \bar{u})$ given by solving:

$$0 = f(x_{eq}, u_{eq}) \tag{2.30}$$

As said above, in order to perform the linearization, an equilibrium point is needed. Such an equilibrium point can be:

$$\bar{x} = [\bar{x_1}\ \bar{y_1}\ \bar{z_1}\ 0\ 0\ 0\ 0\ 0\ \bar{\psi_1}\ 0\ 0\ 0]^T \in R^{12} \tag{2.31}$$

From the equations, we can find that the equilibrium point 2.31 is obtained by the constant input value:

$$\bar{u} = [mg\ 0\ 0\ 0]^T \in R^4 \tag{2.32}$$

Linearizing the nonlinear dynamics 2.26 about the steady state values results in:

$$\Delta\dot{x} = A\Delta x + B\Delta u \tag{2.33}$$

where the Jacobian matrices :

$$A = \frac{\partial f(x,u)}{\partial x}_{\substack{x=\bar{x} \\ u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.34}$$

$$B = \frac{\partial f(x,u)}{\partial u}_{\substack{x=\bar{x}\\u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{-1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \tag{2.35}$$

The linear model is:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{y}_1 = y_2 \\ \dot{z}_1 = z_2 \\ \dot{x}_2 = -g\theta_1 \\ \dot{y}_2 = g\phi_1 \\ \dot{z}_2 = \frac{F_z}{m} \\ \dot{\phi}_1 = \phi_2 \\ \dot{\theta}_1 = \theta_2 \\ \dot{\psi}_1 = \psi_2 - \bar{\psi}_1 \\ \dot{\phi}_2 = \frac{\tau_x}{I_x} \\ \dot{\theta}_2 = \frac{\tau_y}{I_y} \\ \dot{\psi}_2 = \frac{\tau_z}{I_z} \end{cases} \tag{2.36}$$

Then, using the following equations, another inversion is used to determine the command vector composed of the angular speeds ω_i^2 that are delivered to the dc motor controller, which is the real inputs, the inputs to the quadcopter system are the squares of the motor speeds driving the four propellers [7].

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} K_f & K_f & K_f & K_f \\ 0 & -K_f l & 0 & K_f l \\ -K_f l & 0 & K_f l & 0 \\ -K_m & K_m & -K_m & K_m \end{bmatrix}^{-1} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{2.37}$$

2.8 Conclusion

This chapter shows a mathematical model of a quadrotor dynamics using Newton's and Euler's laws. Then a linearized version of the model is obtained. In the following chapter, the resulting linearized model will be used to control the quadrotor using MPC control based on two decentralized techniques.

Chapter 3

Model Predictive Control

3.1 Introduction

The goal of this chapter is to design a trajectory tracking system for the quadcopter in the presence of disturbances. As such, before we start designing trajectory controllers, we first go over the basic knowledge of MPC control theory.

3.2 Model predictive control

3.2.1 Principle of Model Predictive control

The main idea of predictive control is to "use a model to predict the behavior of the system and choose the optimal control in terms of a particular cost while respecting the constraints" [8].

The principle of predictive control is illustrated in Fig 3.1, is a control strategy based on online numerical optimization where future system commands (inputs) and responses (outputs) are model-based predicted in order to optimize a performance index over a finite "N" prediction horizon while respecting operating constraints [9]. This idea is simple and practiced in a rather systematic way in life. For example, the driver of a vehicle knows the desired reference trajectory in advance (the road) on a finite control horizon (his field of view), and taking into account the characteristics of the car (mental model of the vehicle's behavior), he decides which actions (accelerating, braking or

turning the steering wheel) to perform in order to follow the desired trajectory. Only the first (red) driving action is executed at each moment, and the procedure is repeated again for the next actions Thus, at each sampling period, an optimisation problem must be solved in real time.

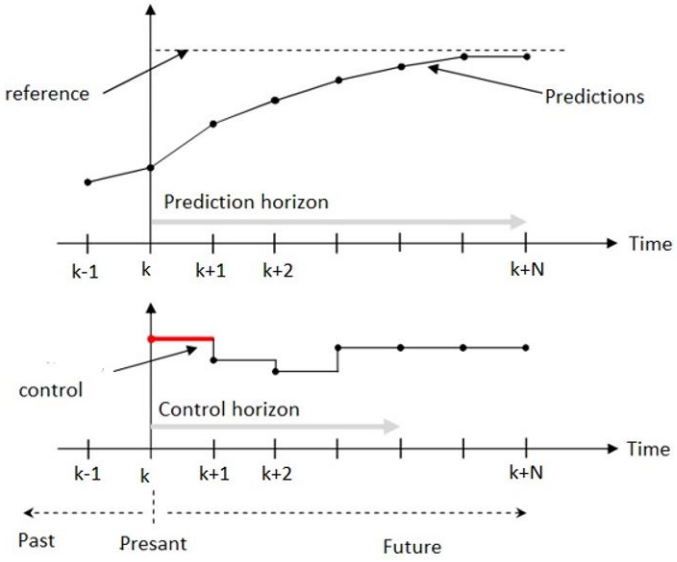


Figure 3.1: Principle of predictive control.

3.2.2 Elements of predictive control

The basic elements of predictive control see Fig 3.2 are:

1 . A model to make predictions.

2 . A cost function to minimize plus constraints.

3 . An optimization algorithm (to calculate the future order).

Different options can be considered for each element, which gives a variety of predictive control algorithms.

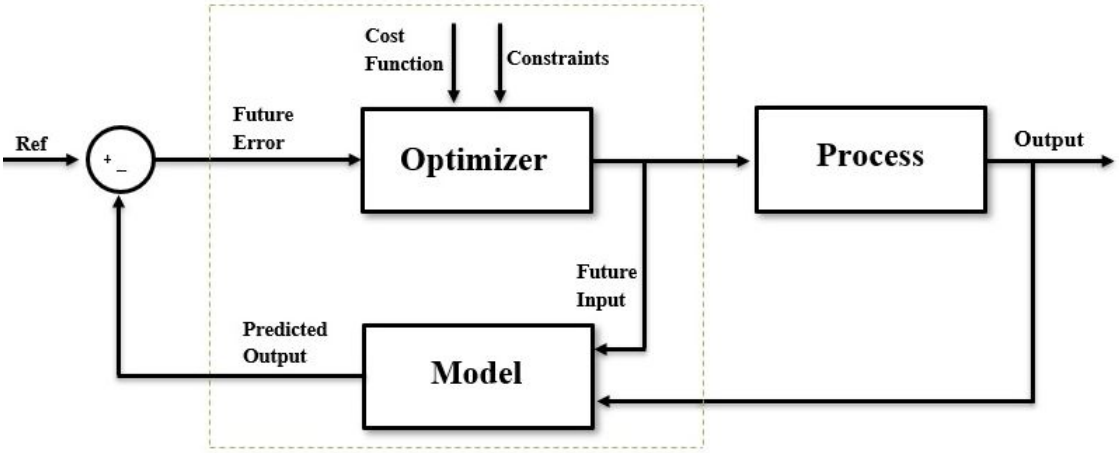


Figure 3.2: Predictive control strategy.

3.2.3 Predictive model

The model must be discrete because the predictive control is a numerical control strategy.

Discrete state model :

$$\begin{cases} x(i+1/k) = Ax(i/k) + Bu(i/k) \\ y(k) = Cx(i/k) \end{cases} \tag{3.1}$$

Where $x \in R^n$ is the state space's vector, $u \in R^m$ is the system's vector of inputs, and $y \in R^L$ is the system's vector of outputs.The MPC controller should also respect constraints on control variables as well as the constrained outputs $z_c \in R^p$

$$\begin{gathered} u_{min} \leq u \leq u_{max}, \\ \Delta u_{min} \leq \Delta u = u_k - u_{k-1} \leq \Delta u_{max} \\ x_{min} \leq x \leq x_{max} \end{gathered} \tag{3.2}$$

3.2.4 Optimal Control law

The optimal control problem, which is at the core of the MPC algorithm, will now be formulated. The cost function "J" penalizes deviations between the ordered predicted outputs 'y' and the reference trajectory 'r' Consider the quadratic cost function below:

$$J(k) = \sum_{K=1}^{N-1} \parallel \ y(i/k) - r(i/k) \parallel_Q^2 + \parallel u(i/k) \parallel_R^2 \tag{3.3}$$

where $y(i/k)$ are the predicted controlled outputs at time k and $u(i/k)$ are the predicted control . N is the prediction horizon ,the matrices $Q \geq 0$ and $R > 0$ are weighting matrices, which are assumed to be constant over the prediction horizon [10] .
The cost function (3.3) may be rewritten as:

$$J(k) = \parallel \ Y(i/k) - \Gamma(i/k) \parallel_Q^2 + \parallel U(i/k) \parallel_R^2 \tag{3.4}$$

Where :

$$\begin{aligned}
Y(k) &= \begin{bmatrix} \hat{y}(k \mid k) \\ \vdots \\ \hat{y}\,(k+N-1 \mid k) \end{bmatrix} \quad \Gamma(k) = \begin{bmatrix} r(k \mid k) \\ \vdots \\ r\,(k+N-1 \mid k) \end{bmatrix} \\
U(k) &= \begin{bmatrix} u(k \mid k) \\ \vdots \\ u\,(k+N-1 \mid k) \end{bmatrix} \quad Q = \begin{bmatrix} Q & 0 & \dots & 0 \\ 0 & Q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q \end{bmatrix} \\
R &= \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R \end{bmatrix}
\end{aligned} \tag{3.5}$$

By deriving the prediction expressions, we can write:

$$Y(k) = \alpha x(k) + \gamma U(k) \tag{3.6}$$

Where:

$Y(k) = [y_{1/k}\ y_{2/k} \cdots\ y_{N/k}]^T$ and $U(k) = [u_{0/K}\ u_{1/k} \cdots u_{N-1/k}]^T$

$$\alpha = \begin{bmatrix} C \\ CA \\ CA^2 \\ . \\ . \\ CA^{N-1} \end{bmatrix}, \gamma = \begin{bmatrix} CB & 0 & 0 \\ CAB + CB & \ddots & \vdots \\ \vdots & \ddots & 0 \\ C\sum_{i=1}^{Nu-2} A^i B & \cdots & 0 \\ \vdots & \ddots & \vdots \\ C\sum_{i=1}^{N-2} A^i B & \cdots & C\sum_{i=1}^{N-1} A^i B \end{bmatrix} \tag{3.7}$$

To obtain the control law, we replace the predictor equation (3.6) in the cost function:

$$J(k) = u^T H u + u^T F + E^T Q E \tag{3.8}$$

Where:

$$F = 2\gamma^T Q E(k) \tag{3.9}$$

$$H = \gamma^T Q \gamma + R \tag{3.10}$$

Also, E = $\Gamma(k) - \alpha x(k)$

Quadratic programming (QP) is used to solve the problem of minimizing the cost function equation 3.8. The optimisation problem is convex if H is positive definite, and the solution can be expressed in a closed form. The assumption that $Q \geq 0$ and $R > 0$ leads to the positive definiteness of H [10].

The solution is given by :

$$U = \frac{1}{2} H^{-1} G \tag{3.11}$$

3.2.5 Constraints

Let us now impose limitations on the constrained and control variables, z_c and U. Linear constraints can be represented in general as:

$$\begin{aligned} Ou(k) &\leq o \\ Gz_c(k) &\leq g \end{aligned} \tag{3.12}$$

This formulation allows for very general constraints, but in the following, only the constraints specified by 3.2 will be considered. Using 3.2, we obtain

$$O = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & & & \\ 0 & 1 & & \vdots \\ & -1 & & \\ \vdots & & \ddots & 0 \\ & & & 1 \\ 0 & \cdots & 0 & -1 \end{bmatrix}, G = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & & & \\ 0 & 1 & & \vdots \\ & -1 & & \\ \vdots & & \ddots & 0 \\ & & & 1 \\ 0 & \cdots & 0 & -1 \end{bmatrix} \tag{3.13}$$

$$o = \begin{bmatrix} u_{Max} \\ -u_{Min} \\ \vdots \\ u_{Max} \\ -u_{Min} \end{bmatrix}, \; g = \begin{bmatrix} x_{Max} \\ -x_{Min} \\ \vdots \\ x_{Max} \\ -x_{Min} \end{bmatrix} \tag{3.14}$$

We rewrite the above constraints in terms of $u(k)$. This gives

$$\begin{bmatrix} O \\ G\gamma \end{bmatrix} u \leq \begin{bmatrix} f \\ -G(\alpha_x(k)) + g \end{bmatrix} \tag{3.15}$$

As we can see, the left side of the inequality is not dependent on k, and could be calculated off-line. The right side depends on the last control signal and the present estimation of the state vector, and should thus be evaluated at each sample.

The optimization problem can now be rewritten using 3.3 and 3.15

$$\begin{aligned} minJ(k) &= u^T Hu - u^T G + E^T QE \\ subject\ to\ \Omega u &\leq f \end{aligned} \tag{3.16}$$

3.3 Control applied in quadrotor (MPC)

Since the quad-copter is a multi-input multi-output (MIMO) system with 12 state variables, 4 inputs, after linearization, the matrix A is 12×12, B is 12×4, as presented in equations (2.34) (2.35). This system complicates the MPC control algorithm. As a result, the initial step towards adopting decentralized control to control the MIMO system was decoupling. For decoupling, we used two separate methods:

- The first is the decoupling inspired by the project [11]
- The second option we chose, which was to decoupling each variable separately from the others(SISO).

3.3.1 The first decentralized control method

We use a sequential design for decoupling, first designing the controllers for the Outer loop as shown in Fig.3.3, then designing the controllers for the Inner loop as a cascade design.

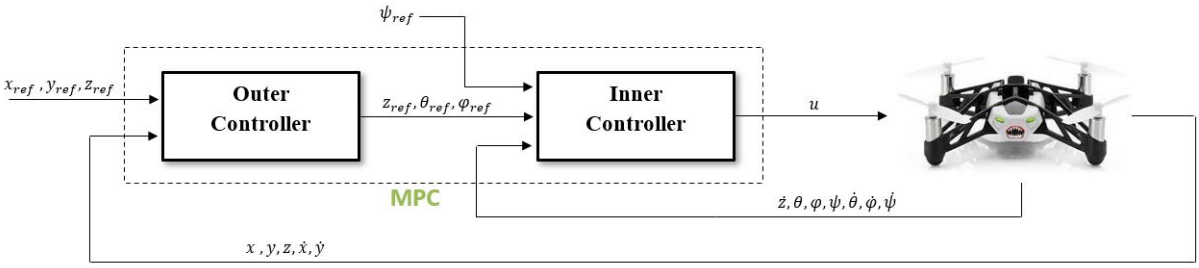


Figure 3.3: Control structure.

The references and states are sent into the MPC control block, which takes into account the dynamics of the real system and outputs a new roll, pitch, yaw, and thrust instruction for the UAV.

Outer loop Control

The linearized outer loop system can be rewritten in decentralized form as follows:

$$\dot{X}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} X_1 + \begin{bmatrix} 0 & 0 & 0 \\ g & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -g & 0 \\ 0 & 0 & 1 \end{bmatrix} U_1 \tag{3.17}$$

Where $\dot{X}_1 = [\dot{x}\ \ddot{x}\ \dot{y}\ \ddot{y}\ \dot{z}]^T$,$X_1 = [x\ \dot{x}\ y\ \dot{y}\ z]^T$ and $U_1 = [\theta\ \phi\ \dot{z}]^T$

The model is a linear and discrete quadrotor model that was generated offline using MATLAB and discretized with a sample time of $T_s = 0.3$s . There are five states and three inputs in the state-space model. The prediction horizon used is N= 30 and we added some constraints to make sure the system remains in close to the equilibrium point around which we linearized the model, in particular the values are contained in table 3.1. These parameters were set based on empirical data to guarantee that the system remained close enough to the linearized model's equilibrium point.

	Value min	Value max
(x,y)	-5 m	5 m
(z)	-1m	4m
(ϕ, θ)	-10°	10°

Table 3.1: Constraints on the outer loop

The outer-loop reference signal r_1 is the space position (r_x, r_y, r_z). We wanted to penalize the position of (x,y,z) between the actual and the desired linear position and input,The final value chosen for performance $Q_1 = diag([1, 1, 50])$ *and* $R_1 = 100 \times eye(3)$.

Inner loop Control

The decentralized version of the linearized inner loop system is as follows:

$$\dot{X}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} X_2 + \begin{bmatrix} 3.5 & 3.5 & 3.5 & 3.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0.56 & 0 & -0.56 \\ -0.56 & 0 & 0.56 & 0 \\ 0.7333 & -0.7333 & 0.7333 & -0.7333 \end{bmatrix} U_2 \tag{3.18}$$

Where $\dot{X}_2 = [\ddot{z}\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\ \ddot{\phi}\ \ddot{\theta}\ \ddot{\psi}]^T$,$X_2 = [\dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T$ and $U_2 = [\omega_1^2\ \omega_2^2\ \omega_3^2\ \omega_4^2]^T$

The inner-loop quadrotor model is subject to constraints on the maximum angle, maximum angle velocity as well as maximum velocity in the z direction – these constraints mainly ensure a validity of the linearized model and have been specified for you in the following table 3.2. And $T_s = 0.1$s sample time was used to discretize the system. The state-space model has four inputs and seven states. We use N= 10 as the prediction horizon.

	Value min	Value max
(ϕ, θ)	-10°	10°
$(\dot{z})$	-1m/s	1m/s
$(\dot{\phi}, \dot{\theta})$	-15°	15°
$(\dot{\psi})$	-60°	60°
(U_2)	0	1

Table 3.2: Constraints on the inner loop

The inner-loop reference signal is $r_2 = (r_{\dot{z}}\ ,\ r_\phi,\ r_\theta,\ r_\psi)$, with the first three components $(r_{\dot{z}},\ r_\phi,\ r_\theta)$ controlled by the outer loop controller and r_ψ (yaw angle) controlled by the outside. Take a look at the control structure in Fig.3.3. The scaling parameters for the Q and R matrices are $Q_2 = diag([50, 150, 150, 0.01])$ *and* $R_2 = 0.1 \times eye(4)$.

3.3.2 The second method for decentralized control

The controller for every variable can be designed independently of the others, making each variable loop a SISO system. We can change the settings of each variable independently in this scenario [12].

The first components pitch (θ), roll (ϕ) are governed by the "outer loop1" and "outer loop2" controllers, in order, The outer-loops reference signal is the state space position (r_x, r_y). The "Inner loop1" controller generates U_1 actuator signals, As for the reference signal (r_θ) is the same as the output value of the outer loop1. while the "Inner loop2" controller generate U_2 and the output value of the "outer loop2" is the same as the reference signal (r_ϕ) in the inner loop2. For the "Inner oop4" controller generate U_4 whereas the reference (r_ψ) comes from outside. The system's required thrust U_3 is generated by the "Inner loop3" controller. The design of each of the controllers will be detailed next.

This Fig.3.4 describes the previously discussed decentralized SISO control.

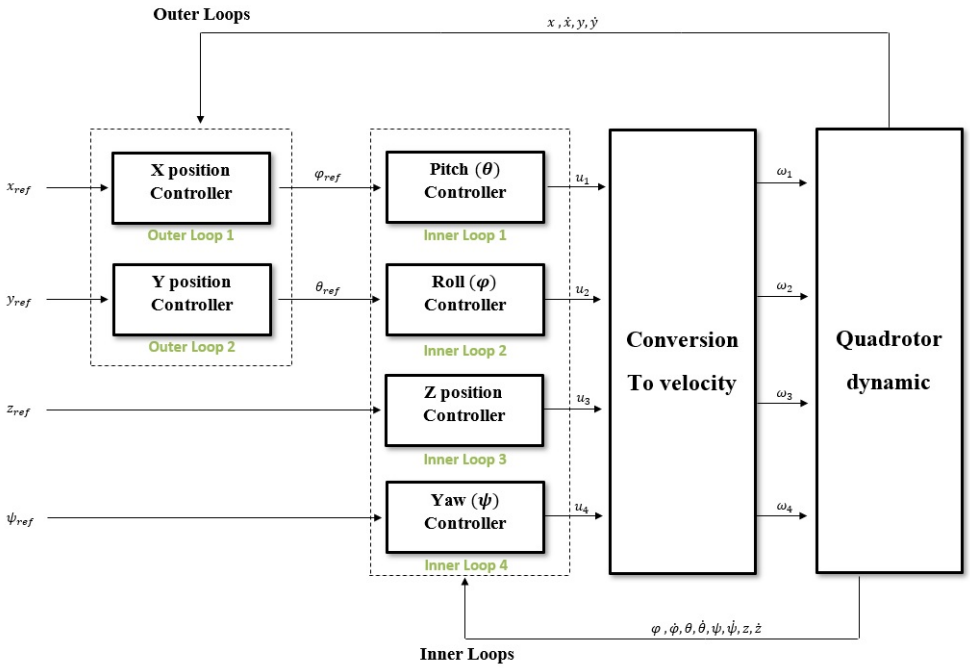


Figure 3.4: Structure of decentralized control

The linearized multi-variable system can be rewritten in decentralized form as follows:

- Outer loop1

$$\dot{X} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} X + \begin{bmatrix} 0 \\ -g \end{bmatrix} \Theta_1 \tag{3.19}$$

where $\dot{X} = [\dot{x}\ \ddot{x}]^T, X = [x_1\ x_2]^T$

- Outer loop2

$$\dot{Y} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} Y + \begin{bmatrix} 0 \\ g \end{bmatrix} \Phi_1 \tag{3.20}$$

where $\dot{Y} = [\dot{y}\ \ddot{y}]^T, Y = [y_1\ y_2]^T$

- inner loop1

$$\dot{\Theta} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \Theta + \begin{bmatrix} 0 \\ \frac{1}{I_y} \end{bmatrix} u_3 \tag{3.21}$$

where $\dot{\Theta} = [\dot{\theta}\ \ddot{\theta}]^T, \Theta = [\theta_1\ \theta_2]^T$

- inner loop2

$$\dot{\Phi} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \Phi + \begin{bmatrix} 0 \\ \frac{1}{I_x} \end{bmatrix} u_2 \tag{3.22}$$

where $\dot{\Phi} = [\dot{\phi}\ \ddot{\phi}]^T, \Phi = [\phi_1\ \phi_2]^T$

- inner loop3

$$\dot{\Psi} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \Psi + \begin{bmatrix} 0 \\ \frac{1}{I_z} \end{bmatrix} u_4 \tag{3.23}$$

where $\dot{\Psi} = [\dot{\psi}\ \ddot{\psi}]^T, \Psi = [\psi_1\ \psi_2]^T$

- inner loop4

$$\dot{Z} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} Z + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u_1 \tag{3.24}$$

where $\dot{Z} = [\dot{z}\ \ddot{z}]^T, Z = [z_1\ z_2]^T$

The sampling time $T_s = 0.3$ was used to discrete the system in the Outer loops, and the prediction horizon was set to N = 30.and to discretize the system in the Inner loops, we used $T_s = 0.1$ sample time and N= 10 as the prediction horizon.
The scaling parameters for the Q and R matrices are:

- In Outer loop 1,2 $Q_{x,y} = 10; R_{x,y} = 1000$

- In inner loop 1,2 $Q_{\theta,\phi} = 1000; R_{\theta,\phi} = 0.01$

- In Inner loop 3 $Q_{\psi} = 10; R_{\psi} = 0.1$

- In Inner loop 4 $Q_z = 500; R_z = 0.01$

The constraints applied in the system are :

	Value min	Value max
(x,y)	-5 m	5 m
$(\dot{x}, \dot{y})$	-2 m/s	2 m/s
(ϕ, θ)	-10°	10°
$(\dot{\phi}, \dot{\theta})$	-15°	-15°
(z)	-1 m	4 m
$(\dot{z})$	-1 m/s	1m/s
$(\dot{\psi})$	-60°	60°
(U_1, U_2)	-5.6	5.6
(U_3)	0	$4.k_f$
(U_4)	$-2.k_m$	$2.k_m$

Table 3.3: Constraints on the outer and inner loops

3.4 Simulation Results

These simulations show the proposed path tracking method. Using the predictive controller with no disturbance. Once the constraints and objective function has been generated, we can solve the optimization problem using the Yalmip toolbox. The values of the reference signal can also be modified, and it can be sinusoidal, step, or ramp input, depending on the requirements. We used the first reference path which is step a and the second is a circle.

The parameters of the quad-rotor used in simulation:

parameters	Description	Value
m	mass	8 Kg
l	arm length	0.2
g	gravity acceleration	9.8 m/s^2
I_x	inertia around x-axis	10 Kg.s^2
I_y	inertia around y-axis	10 Kg.s^2
I_z	inertia around z-axis	15 Kg.s^2
k_f	thrust factor	28 N.s^2
k_m	drag factor	11 N.m.s^2

Table 3.4: System parameters of quadrotor

3.4.1 Simulation 1 without disturbance

This Fig.3.5 of MATLAB shows the MPC control structure 1 of the quad-rotor.

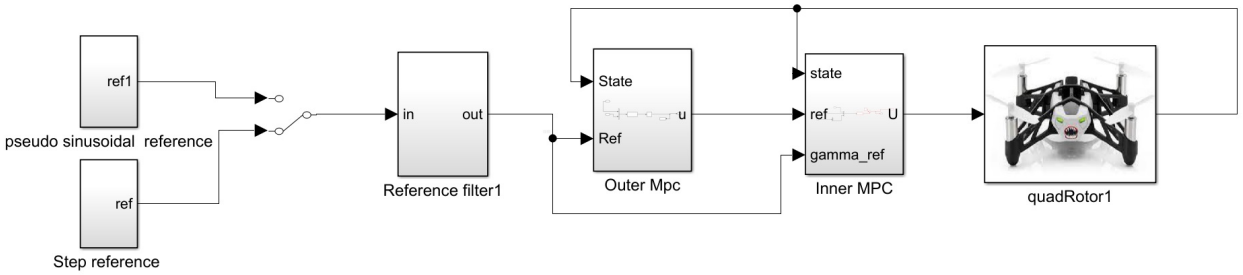
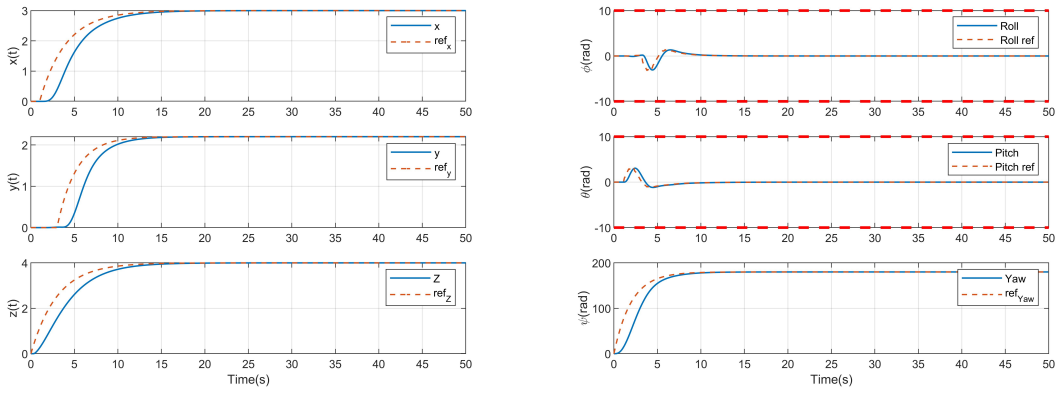


Figure 3.5: MPC MATLAB Simulink bloc diagram.

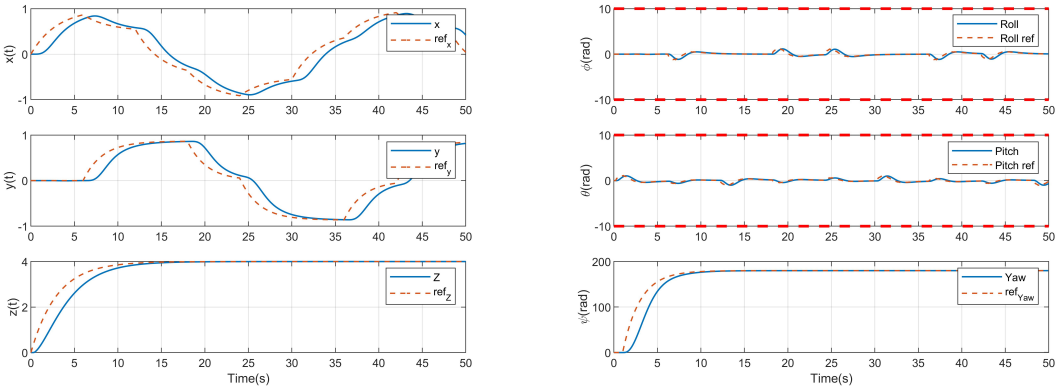
We will now simulate the system using the first decoupling, the parameters of which are explained in section 3.3.1 .



(a) The positions (x ,y, z)

(b) The angulars (ϕ, θ, ψ)

Figure 3.6: Simulation results with step reference



(a) The position (x ,y, z) (b) The angulars (ϕ, θ, ψ)

Figure 3.7: Simulation results with pseudo sinusoidal reference

The outputs of positions and angles are shown in Figures Fig.3.6,3.7. It is clear that it reaches the reference point, but with a noticeable overshoot.

3.4.2 Simulation 2 without disturbance

This Fig.3.8 of MATLAB shows the MPC control structure 2 of quadrotor.

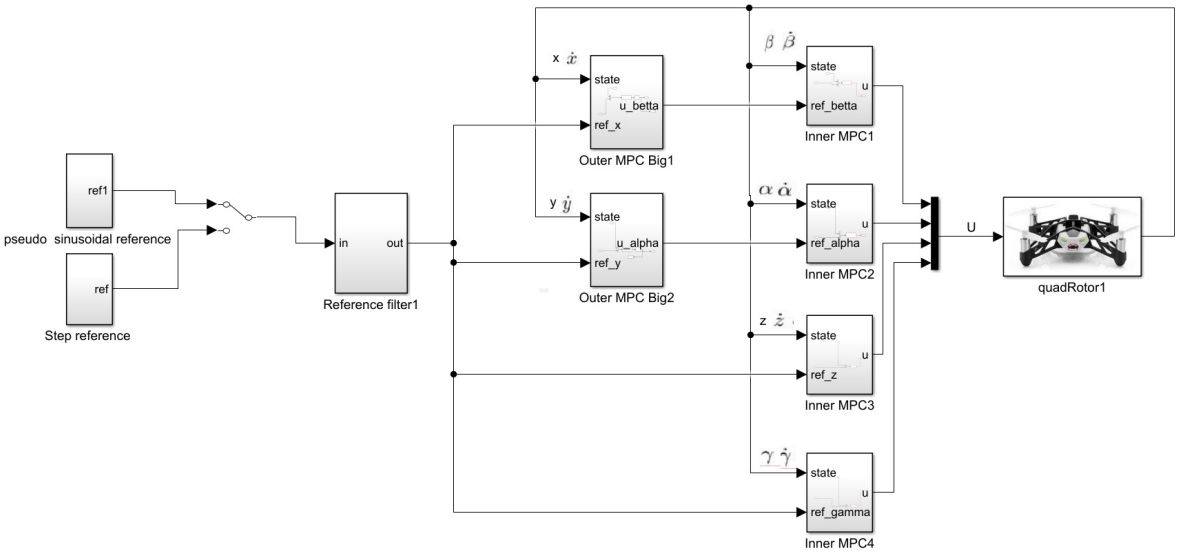
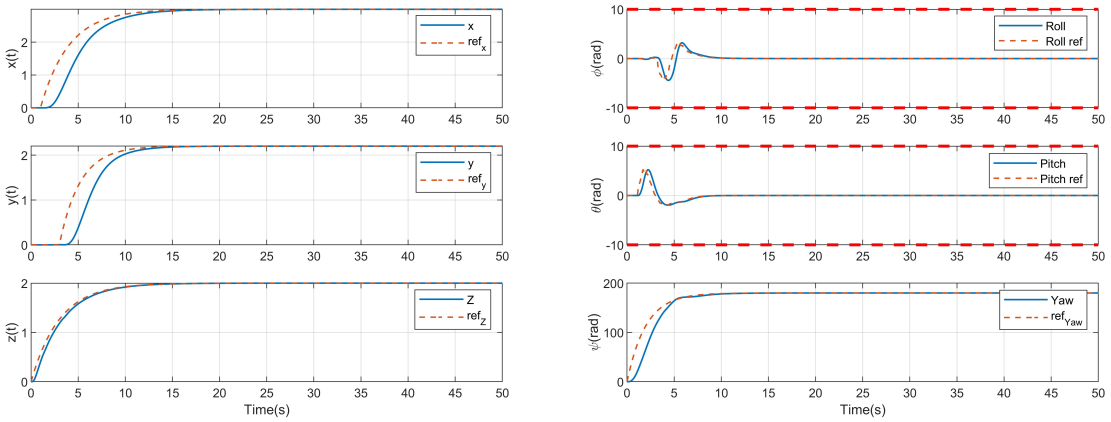


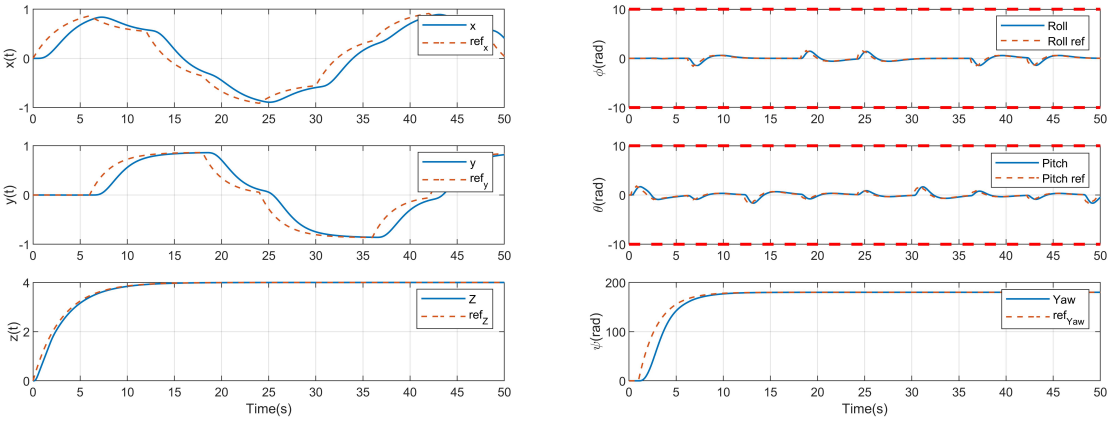
Figure 3.8: MPC MATLAB Simulink bloc diagram 2

We will now simulate the system using the second decoupling,the parameters of which are explained in section 3.3.2



(a) the position (x ,y, z) (b) the angular (ϕ, θ, ψ)

Figure 3.9: Simulation results with step reference



(a) the position (x ,y, z) (b) the angular (ϕ, θ, ψ)

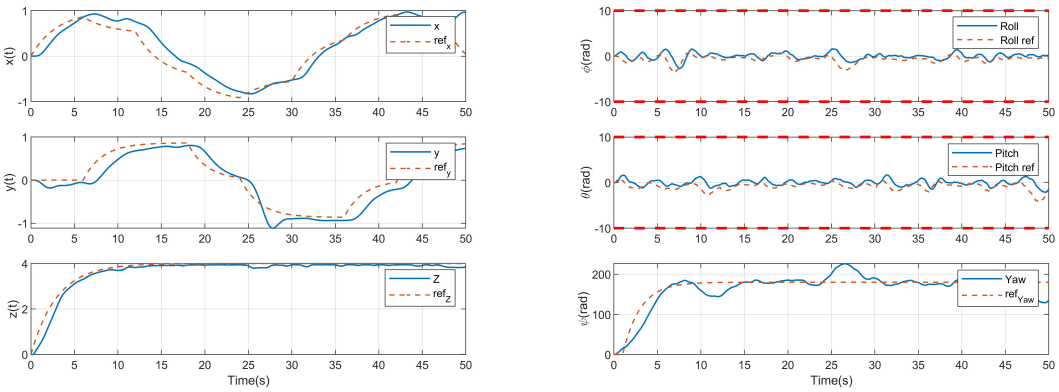
Figure 3.10: Simulation results with pseudo sinusoidal reference

The positions and angles are represented in the figures above 3.10,3.9. Follows the target reference with good tracking, although with a small overshoot.

3.4.3 Simulation results with disturbance

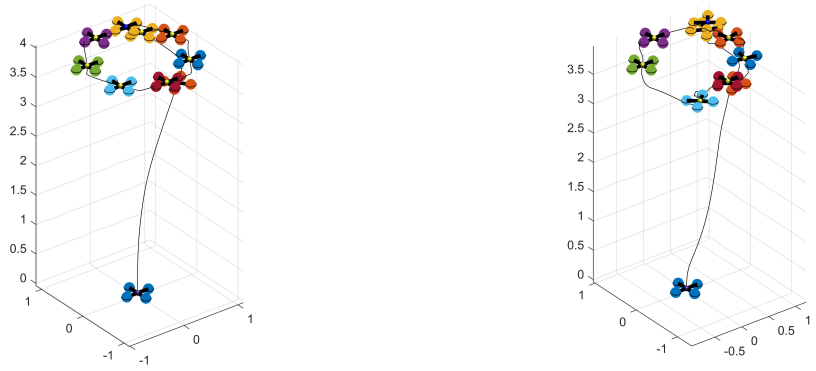
Now we check if the quadrotor follows the reference path by applying disturbance to the drone model only, that is, adding random disturbances to the quadrotor's positional and angular acceleration generated by the wind.

After that, we can run the simulation 1 and 2 with disturbances.



(a) the position (x ,y, z) (b) the angular (ϕ, θ, ψ)

Figure 3.11: Simulation results with pseudo sinusoidal reference



(a) Without disturbance (b) With disturbance

Figure 3.12: Circular trajectory in 3D

Although the control technique yields a good path following the performance, it does not perform well under disturbances as shown in Fig.3.12, To increase the predictive controller's ability to resist disturbances by adding a disturbance estimator.

3.5 Disturbance estimator

In practical applications, there are always modeling disturbances present. The MPC formulation described above contains no explicit mechanism to deal with these complications. In order for the controller to be useful in practice, these problems have to be considered. The quadrotor is susceptible to turbulent wind fields during actual flight. Thus, in order to create disturbance, a simple disturbance estimator model is used to generate wind disturbances, which are then added to the design of the predictive control unit.

3.5.1 Design a disturbance estimator

The nonlinear model is subject various disturbances. is a random disturbance with a nonzero mean value. There is also a gravity error (disturbance affecting the $\dot{z}$ component) and multiplicative and additive input disturbance. All these disturbances can be packed into a single disturbance (random) variable $d \in R^8$ with mean d and zero-mean component $\tilde{d}$. After that, the discrete-dynamics can be written as:

$$x(k+1) = Ax(k) + BU(k) + d + \tilde{d}(k) \tag{3.25}$$

Where $x = [z\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T$, $d = [d_1\ d_2 ...\ d_8]^T$ and $U = [u_1\ u_2\ u_3\ u_4]^T$

our goal is to estimate the time-invariant mean d while rejecting the unpredictable $\tilde{d}_k$. To estimate d, design a disturbance estimator (not a Kalman filter because there is no measurement noise). The disturbance can be modeled as a dynamical system:

$$d(k+1) = d(k) \tag{3.26}$$

we create a full-state estimator for the augmented system:

$$\underbrace{\begin{bmatrix} \text{x(k+1)} \\ \text{d(k+1)} \end{bmatrix}}_{X_a(k+1)} = \underbrace{\begin{bmatrix} A_{8\times8} & I_{8\times8} \\ 0_{8\times8} & I_{8\times8} \end{bmatrix}}_{\hat{A}} \underbrace{\begin{bmatrix} x(k) \\ d(k) \end{bmatrix}}_{X_a(k)} + \underbrace{\begin{bmatrix} \text{B}_{8\times4} \\ 0_{8\times4} \end{bmatrix}}_{\hat{B}} U(k),\ y = \underbrace{[I_{8\times8}\ 0_{8\times8}]}_{\hat{C}} \begin{bmatrix} \text{x(k)} \\ \text{d(k)} \end{bmatrix} \tag{3.27}$$

Where A,B are the matrices that result from a discrete system with a simple time Ts=0.1.

The state estimator is chosen as the Luenberger observer of the form:

$$\hat{X}_a(k+1) = A_a\hat{X}_a(k) + B_aU(k) + L(y_a - \hat{y}_a) \tag{3.28}$$

Denoting the observation error as $\tilde{X}_a = X_a - \hat{X}_a$, so the observation error dynamics are given by:

$$\tilde{X}_a(k+1) = A_a\tilde{X}_a(k) + B_aU(k) + LC(X_a(k) - \hat{X}_a(k)) \tag{3.29}$$

$$\tilde{X}_a(k+1) = (A_a - LC)\tilde{X}_a(k) \tag{3.30}$$

The discrete observer gain L must be chosen in such a way that the poles of the matrix $(A_a - LC)$ are inside the unit circle

3.5.2 Simulation with disturbance

In this section, we will simulate our system with disturbance on a quadrotor model, as well as have an estimator, and we will add the estimator's disturbance output to the MPC control.We still use the same structure and parameters that we discussed in Section 3.3.2

This Fig.3.13 of MATLAB diagram shows the simple MPC with disturbance estimator control for the quadrotor.

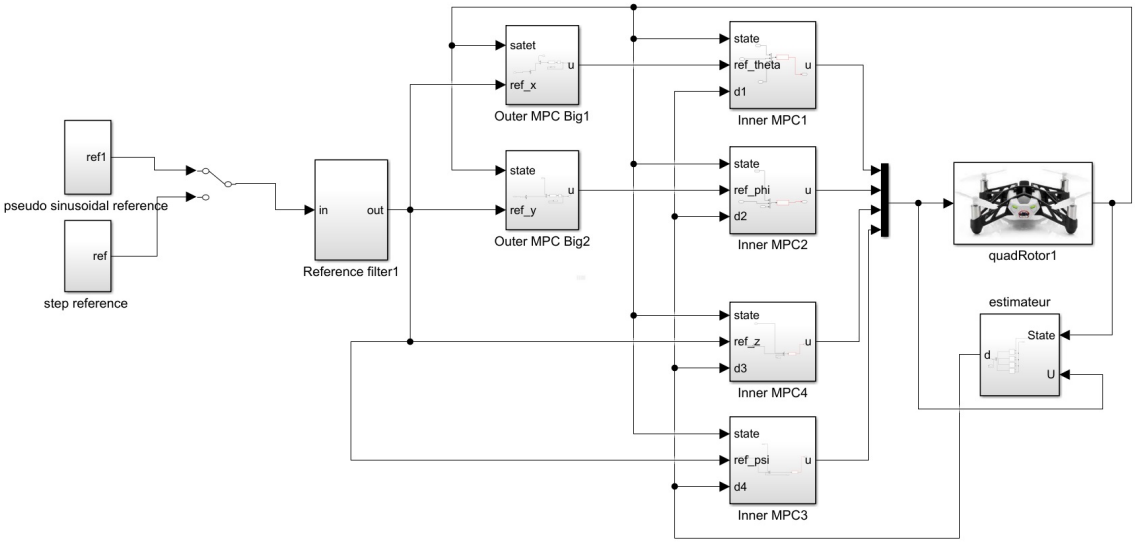
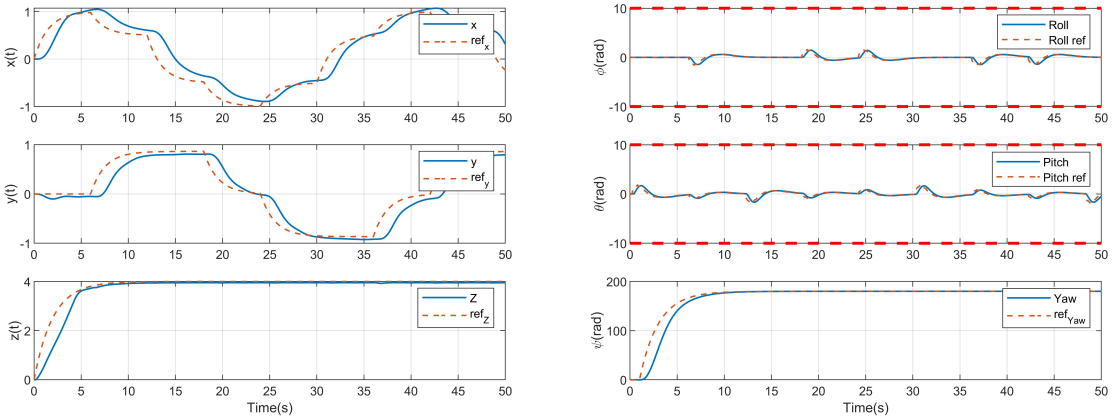


Figure 3.13: MPC MATLAB Simulink bloc diagram with estimator.

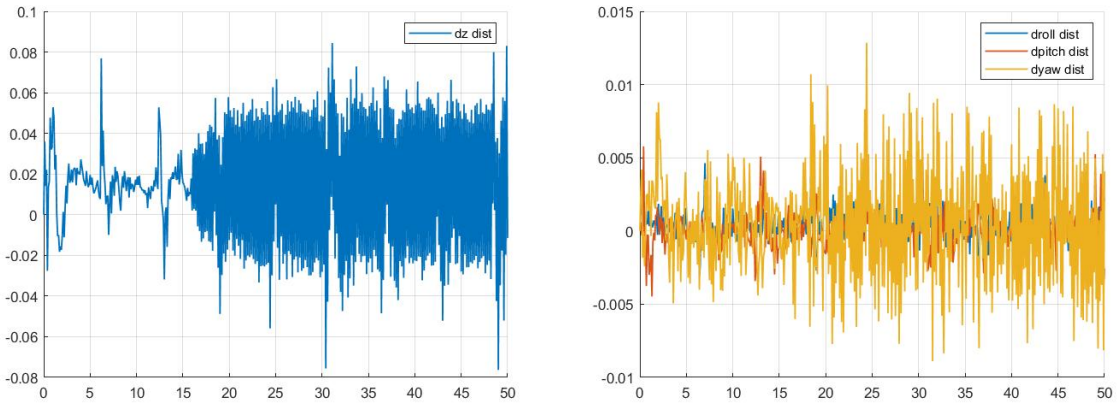
Simulation result

Now we can run the simulation and see how the quadrotor follows the desired path when there are disturbances



(a) The position (x ,y, z) (b) The angular (ϕ, θ, ψ)

Figure 3.14: Simulation results with pseudo-sinusoidal reference



(a) Disturbance (z) (b) Disturbance (ϕ, θ, ψ)

Figure 3.15: Simulation results of the disturbance.

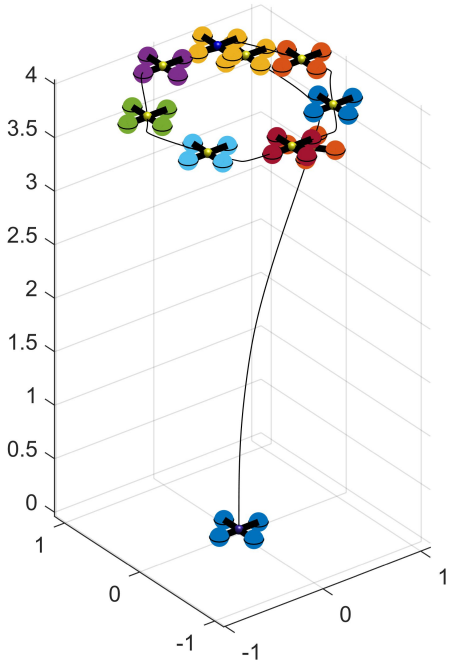


Figure 3.16: Circular trajectory in 3D with estimator

As seen in Fig.3.14, both positions and angles follow the desired path with a little deviation, proving that the quadrotor follows the desired path despite the presence of disturbances. This is because when the disturbance is added to the MPC, it optimizes the cost function, which in turn gives the optimal controller for making the drone resist disturbance.

3.6 Conclusion

In this chapter, we introduced some predictive control concepts such as the control system principle. Next, we talk about the control structure. Finally, we simulate the system and show the use of the Model Predictive Control (MPC) approach for different trajectories (i.e. circle, step trajectory) and the main feature of the MPC controller that optimizes the control of the inputs and outputs while taking into account disturbances and constraints.

Chapter 4

Active Disturbance Rejection Control

4.1 Introduction

In this chapter, we will discuss the concept of active disturbance rejection control (ADRC), which has emerged as an alternative that combines easy applicability known from classical PID-type control methods with the power of modern model-based approaches. The basis for ADRC is an observer who jointly treats actual disturbances and the unknown dynamics, which makes ADRC an attractive choice for practitioners and promises good robustness against process variations due to uncertainties and disturbances.

4.2 Linear Active Disturbance Rejection Control

The LADRC control is becoming more and more popular, and several authors have treated its description and analysis. To illustrate its concept, we first consider the general model of a non-linear time-varying dynamic system of order n, with a single input u and a single output y (SISO Single Input Single Output), described by the following equation :

$$y^{(n)} = f(y(t), w(t)) + b_0 u(t) \tag{4.1}$$

- $w(t)$: models all external disturbances.

- $f(y^{(n-1)}, y^{(n-2)}), f(y(t), w(t))$: system dynamics and all internal and external disturbances.

$-b_0$:constant parameter

It should be mentioned here that the order of the system and parameter b is known. The ADRC has the advantage of being able to control this type of system by estimating and rejecting disturbances including those depending on the system itself, something that other controllers will not be able to realize [13] .

4.2.1 Extended State Observer- ESO

The basic element of the ADRC control is the Extended State Observer whose principle is described above:

The previous equation 4.1 can be described in state representation as follows:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \dots \\ \dot{x}_n = x_{n+1} + b_0.u \\ \dot{x}_{n+1} = \dot{f} \\ y = x_1 \end{cases} \tag{4.2}$$

Or in the matrix form :

$$\begin{cases} \dot{X} = A.X + B.U + E.\dot{f} \\ \dot{Y} = C.X \end{cases} \tag{4.3}$$

As we note the state vector X contains an additional derivable state x_{n+1}, which models the perturbation.

It is therefore obvious that the observer for this system takes into consideration the additional variable, its form is given by:

$$\begin{cases} \dot{\hat{X}} = A.\hat{X} + B.U + \beta(Y - \hat{Y}) \\ \hat{Y} = C.\hat{X} \end{cases} \tag{4.4}$$

Where :

- $\hat{X} = [\hat{x}_1\ \hat{x}_2\ ..\ \hat{x}_{n+1}]^T$: estimate state vector
- $\beta = [\beta 1\ \beta 2\ ..\ \beta_{n+1}]^T$: vector gains of the observer.

4.2.2 Control law

when $(A - \beta C)$ is asymptotically stable, y(t) and its (n-1) derivatives will be estimated by $\hat{x}_1\ \hat{x}_2\ at\ \hat{x}_n$ and the set of disturbances will be estimated by $\hat{x}_{n+1}$ thus, if we choose the control law:

$$u(t) = \frac{u_0(t) - \hat{f}}{b_0} \tag{4.5}$$

The differential equation of the system becomes:

$$y^{(n)} = f(y(t), w(t)) + u_0(t) - \hat{f} \tag{4.6}$$

If the ESO is properly set up:

$$f(y(t), w(t)) = \hat{f} = \hat{x}_{n+1} \tag{4.7}$$

And consequently :

$$y^{(n)} \approx u_0(t) \tag{4.8}$$

This reduces the system to a set of cascade integrators, and the system can finally be controlled by:

$$u_0(t) = k_1(r(t) - y(t)) + k_2(\dot{r}(t) - \dot{y}(t)) + k_3(\ddot{r}(t) - \ddot{y}(t)) + \dots k_n(r^{(n-1)}(t) - y^{(n-1)}(t)) \tag{4.9}$$

And since y(t) and its (n-1) derivatives will be estimated by $\hat{x}_1$ $\hat{x}_2$ at $\hat{x}_n$ and the set of disturbances will be estimated by $\hat{x}_{n+1}$:

$$u(t) = \frac{k_1(r(t) - \hat{x}_1(t)) + k_2(\dot{r}(t) - \hat{x}_2(t)) + \dots k_n(r^{(n-1)}(t) - \hat{x}_n(t)) - \hat{x}_{n+1}}{b_0} \tag{4.10}$$

Which can be expressed as :

$$u(t) = K_0(\hat{r}(t) - \hat{x}(t)) \tag{4.11}$$

where :

- $\hat{r}(t) = [r(t)\ \dot{r}(t)\ \dots r^{n-1}(t)]^t$: reference vector
- $K_0 = [k_1\ k_2\ \dots k_n\ 1]/b_0$:controller gain vector

4.3 Application to Second-Order Linear ADRC

4.3.1 Second-Order Process

we now consider a second-order process, P(s), with a DC gain, K, damping factor, D, and a time constant, T.

$$P(s) = \frac{y(s)}{u(s)} = \frac{K}{T^2s^2 + 2DT\ s + 1} \Leftrightarrow\ T^2\ddot{y}(t) + 2DT\dot{y}(t) + y(t) = Ku(t) \tag{4.12}$$

As for the second-order case, we add an input disturbance, d(t), abbreviate $b = \frac{K}{T^2}$ and split b into a known and unknown part, $b = b_0 + \Delta b$:

$$\ddot{y}(t) = \underbrace{\left(-\frac{2D}{T}\dot{y}(t) - \frac{1}{T^2}y(t) + \frac{1}{T^2}d(t) + \Delta b u(t)\right)}_{\text{generalized disturbance } f(t)} + b_0u(t) = f(t) + b_0u(t) \tag{4.13}$$

In state representation, this equation can be put in the form:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 + b \cdot u \\ \dot{x}_3 = \dot{f} \\ y = x_1 \end{cases} \tag{4.14}$$

Or in a matrix format:

$$\dot{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{A} x(t) + \underbrace{\begin{bmatrix} 0 \\ b_0 \\ 0 \end{bmatrix}}_{B} u(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{f}(t) \tag{4.15}$$

$$y(t) = \underbrace{\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}_{C} x(t) \tag{4.16}$$

where : $\dot{x}(t) = [\dot{x}_1(t)\ \dot{x}_2(t)\ \dot{x}_3(t)]^T$ and $x(t) = [x_1(t)\ x_2(t)\ x_3(t)]^T$

4.3.2 State Observer

To which we associate a state observer, defined by the following state representation:

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + \beta(y - \hat{y}) \\ \hat{y} = Cx \end{cases} \tag{4.17}$$

where:

$$\dot{\hat{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \hat{x}(t) + \begin{bmatrix} 0 \\ b_0 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} (y(t) - \hat{x}_1(t)) \tag{4.18}$$

$$= \underbrace{\begin{bmatrix} -\beta_1 & 1 & 0 \\ -\beta_2 & 0 & 1 \\ -\beta_3 & 0 & 0 \end{bmatrix}}_{A-LC} \hat{x}(t) + \underbrace{\begin{bmatrix} 0 \\ b_0 \\ 0 \end{bmatrix}}_{B} u(t) + \underbrace{\begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}}_{\beta} y(t)$$

Where: $\hat{x}_1(t) = \hat{y}(t)\ , \hat{x}_2(t) = \dot{\hat{y}}(t)\ , \hat{x}_3(t) = \hat{f}(t)$

The designed LADRC structure for second order time-invariant systems is illustrated in Fig.4.1

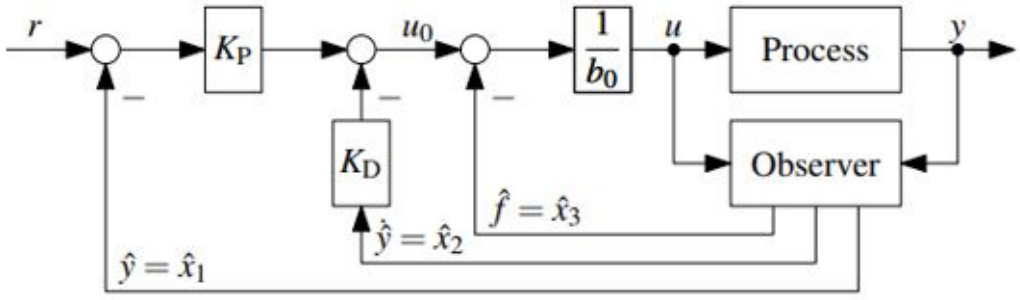


Figure 4.1: Control loop structure with (ADRC) for a second-order process

4.3.3 Control structure

The control form is as described in equation 4.10.

$$u(t) = \frac{u_0(t) - \hat{f}}{b_0} \tag{4.19}$$

Where: $u_0(t) = K_P(r(t) - \hat{y}(t)) - K_D.\dot{\hat{y}}(t)$

$$u(t) = \frac{(K_P(r(t) - \hat{x}_1(t)) - K_D.\hat{x}_2(t)) - \hat{x}_3(t)}{b_0} \tag{4.20}$$

4.3.4 Closed loop dynamics

Choose K_P and K_D, e.g. according to a desired settling time,under ideal conditions, this leads to:

$$\frac{1}{K_p}.\ddot{y}(t) + \frac{K_D}{K_p}.\dot{y}(t) + y(t) = r(t) \tag{4.21}$$

While any second-order dynamics can be set using K_P and K_D, one practical approach is to tune the closed loop to a critically damped behavior and a desired 2% settling time T_r, i.e., choose K_P and K_D to get a negative-real double pole, $s_{1/2}^{CL} = s^{CL}$:

$$K_P = (s^{CL})^2 \; and \; K_D = -2.s^{CL} \quad with \quad s^{CL} \approx -\frac{6}{T_r} \tag{4.22}$$

The observer poles,can be placed using a simple rule:

$$s_{1/2/3}^{ESO} = s^{ESO} \approx (3 \cdots 10).s^{CL} \quad with \quad s^{CL} - \frac{6}{T_r} \tag{4.23}$$

4.3.5 Observer dynamics

After the pole locations have been determined in this manner, the observer gains are calculated using the characteristic polynomial of $(A - \beta C)$:

$$det(sI-(A-\beta C)) = s^3+\beta_1.s^2+\beta_2.s+\beta_3 \stackrel{!}{=} (s-s^{ESO})^3 = s^3-3s^{ESO}.s^2+3.(s^{ESO})^2.s-(s^{ESO})^3$$

The respective solutions for β_1, β_2 *and* β_3 are:

$$\beta_1 = -3.s^{ESO}, \beta_2 = 3.(s^{ESO})^2 \quad \beta_3 = -(s^{ESO})^3$$

4.3.6 Simulation And Result LADRC

Simulation of LADRC with a 2nd order system:

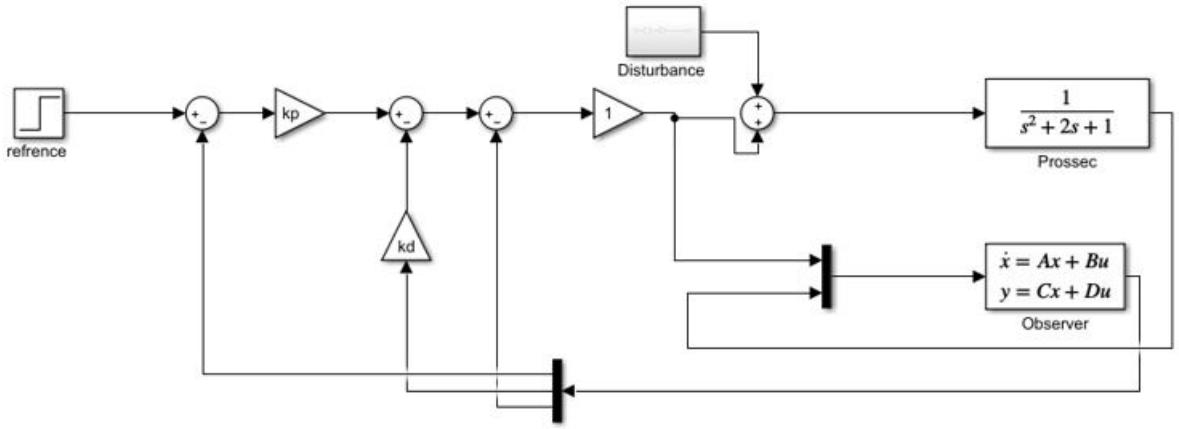
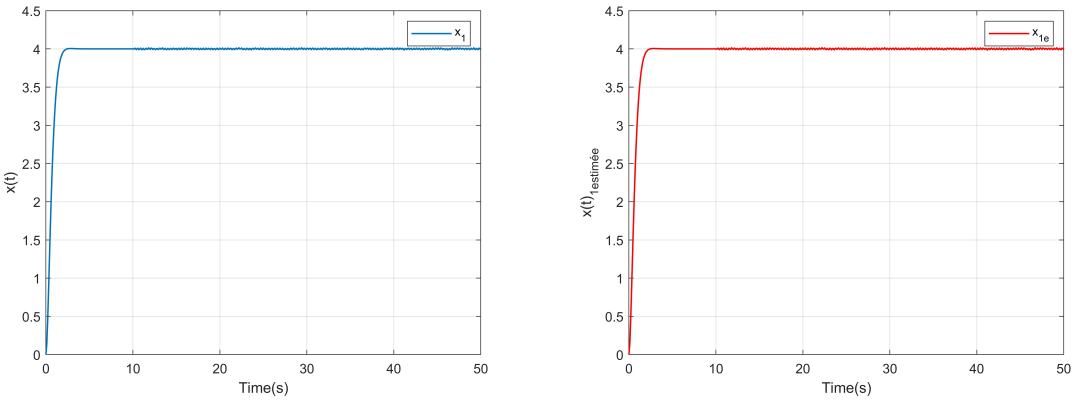


Figure 4.2: Simulink bloc digram of LADRC with a 2nd order

A fixed second-order ADRC controlling second-order process 4.12§ with varying parameters. Nominal process parameters: K = 1, D = 1, T = 1.

ADRC parameters, $b0 = \frac{K}{T^2} = 1$, $T_r = 1$, $s^{ESO} = 10s^{CL}$. The parameters of the PD controller are, following equation.4.22 , set to $K_P = 36$ and $K_D = 12$.

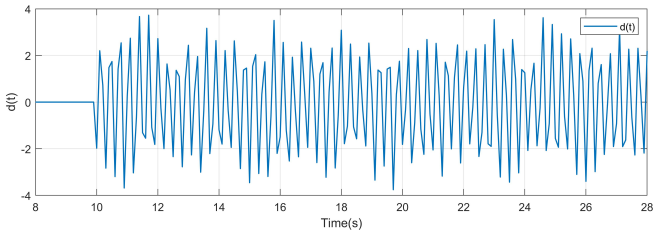
After that can we ran the Simulation and see the result :



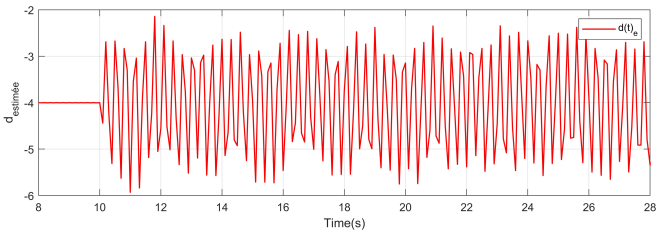
(a) Actual output (b) Estimated outpu

Figure 4.3: (a) Actual and (b) Estimated outputs.

The following figures present respectively the external disturbance applied on the controlled system, and the estimated disturbance:



(a) Actual disturbance



(b) Estimated disturbance

Figure 4.4: (a) Actual and (b) Estimated disturbances.

The estimate of the disturbance becomes closer and closer to reality.

4.4 Discrete Time ADRC

Practical implementations of a controller with a state observer, such as the ADRC technique, will almost certainly be done in discrete time, e.g. using a microcontroller. Because the real linear ADRC control law is based on proportional state feedback, a discrete time version can already be obtained by discretizing the extended state observer, which will be done in this Section. Simulative experiments will be performed to visually analyze the effect of the discretisation process and measurement disturbance on the performance of the control loop[14].

4.4.1 Discretisation of the State Observer

The standard approach for a discrete-time observer is:

$$\hat{x}(k+1) = A_{\mathrm{d}} \cdot \hat{x}(k) + B_{\mathrm{d}} \cdot u(k) + \beta_{\mathrm{p}} \cdot (y(k) - C_{\mathrm{d}} \cdot \hat{x}(k)) \tag{4.24}$$

$A_{d,}B_d$, and C_d are time-discrete versions of the respective matrices in the state space process models equations 4.15 generated via a discretisation procedure. Because there is no matrix D_d in the observer equations, the model is discretisation must provide $D_d = 0$.

When we look at the equation for the estimation error in equation 4.24, we can see that, just like in the continuous case, the matrix determines the dynamics of the error decay.

$$e(k+1) = x(k+1) - \hat{x}(k+1) = (A_{\mathrm{d}} - \beta_{\mathrm{p}} \cdot C_{\mathrm{d}}) \cdot (x(k) - \hat{x}(k)) \tag{4.25}$$

Because observer gains in Lp influence matrix pole placement, they can be chosen so that the estimation settles within a desired time.

This observer strategy is also known as "prediction observer" since the current measurement, y(k), will only be used to correct the estimate in the next time step,$\hat{x}(k+1)$. In order to avoid unnecessary time delays (which may even destabilize the control loop), The core idea is to divide the update equation into two steps, similar to Kalman filtering: a prediction step to predict $\hat{x}(k/k-1)$ based on measurements from the previous

time step, k-1, and a correction step to achieve the final estimate, $\hat{x}(k/k)$, adding the most recent measurement[14], y(k):

$$\begin{aligned} \hat{x}(k \mid k-1) &= A_{\mathrm{d}} \cdot \hat{x}(k-1 \mid k-1) + B_{\mathrm{d}} \cdot u(k-1) \quad \text{(prediction)} \\ \hat{x}(k \mid k) &= \hat{x}(k \mid k-1) + \beta_{\mathrm{c}} \cdot (y(k) - C_{\mathrm{d}} \cdot \hat{x}(k \mid k-1)) \quad \text{(correction)} \end{aligned} \tag{4.26}$$

If we put the prediction into the correction equation and abbreviate $\hat{x}(k/k) = \hat{x}(k)$, we obtain one update equation for the observer:

$$\hat{x}(k) = (A_{\mathrm{d}} - \beta_{\mathrm{c}} \cdot C_{\mathrm{d}} \cdot A_{\mathrm{d}}) \cdot \hat{x}(k-1) + (B_{\mathrm{d}} - \beta_{\mathrm{c}} \cdot C_{\mathrm{d}} \cdot B_{\mathrm{d}}) \cdot u(k-1) + \beta_{\mathrm{c}} \cdot y(k) \tag{4.27}$$

From the estimation error, we can see that, in contrast to the prediction observer, the error dynamics are determined by the matrix (A_d - $\beta_c C_d A_d$) :

$$e(k+1) = x(k+1) - \hat{x}(k+1) = (A_{\mathrm{d}} - L_{\mathrm{C}} \cdot C_{\mathrm{d}} \cdot A_{\mathrm{d}}) \cdot (x(k) - \hat{x}(k)) \tag{4.28}$$

This must be considered while computing the observer gains in β_c, i.e., β_c must be chosen so that the eigenvalues of (A_d - $\beta_c C_d A_d$) match the desired observer pole locations.

4.5 Application to Second-Order Discrete ADRC

4.5.1 Second-Order Process

We will repeat the same process as described in section (4.4.1)

$$P(s) = \frac{K}{T^2 s^2 + 2DTs + 1} \tag{4.29}$$

4.5.2 State Observe

The discrete time observer equation, which has been discussed in section (4.8.1)

$$\hat{x}(k) = \underbrace{(A_{\mathrm{d}} - \beta_{\mathrm{c}} \cdot C_{\mathrm{d}} \cdot A_{\mathrm{d}})}_{A_{ESO}} \cdot \hat{x}(k-1) + \underbrace{(B_{\mathrm{d}} - \beta_{\mathrm{c}} \cdot C_{\mathrm{d}} \cdot B_{\mathrm{d}})}_{B_{ESO}} \cdot u(k-1) + \beta_{\mathrm{c}} \cdot y(k) \tag{4.30}$$

$C_d = C$ and $D_d = D = 0$ remain unchanged in the second example in equation 4.15, and one obtains for A_d and B_d.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ b_0 \\ 0 \end{bmatrix}$$

Can be used ZOH discretisation [15] can be used to derive the discrete time versions of the matrices A, B, and C from the state space process models, which are required for the observer equations.

$$A_{\mathrm{d}} = I + \sum_{i=1}^{\infty} \frac{A^i \cdot T_{\mathrm{s}}^i}{i!}, \quad B_{\mathrm{d}} = \left(\sum_{i=1}^{\infty} \frac{A^{i-1} \cdot T_{\mathrm{s}}^i}{i!} \right) \cdot B, \tag{4.31}$$

$$\text{gives} \quad A_{\mathrm{d}} = \begin{bmatrix} 1 & T_{\mathrm{s}} & T_{\mathrm{s}}^2/2 \\ 0 & 1 & T_{\mathrm{s}} \\ 0 & 0 & 1 \end{bmatrix}, \quad B_{\mathrm{d}} = \begin{bmatrix} b_0 \cdot T_{\mathrm{s}}^2/2 \\ b_0 \cdot T_{\mathrm{s}} \\ 0 \end{bmatrix} \tag{4.32}$$

4.5.3 Observer Dynamics

The observer gain, $\beta_c = [\beta_1 \ \beta_2 \ \beta_3]^T$ is computed to obtain the desired observer dynamics. The desired pole locations can be formulated in the s-plane first, as shown in Sections 4.3.1, and then mapped to the z-plane: $z^{ESO} = e^{s^{ESO}.T_s}$, β_c can be calculated either numerically or analytically. To demonstrate this, we shall develop the equations for both poles for the first-order example with one common pole location, z^{ESO}:

$$\begin{gathered} \det\left(zI - (A_{\mathrm{d}} - \beta_{\mathrm{c}} \cdot C_{\mathrm{d}} \cdot A_{\mathrm{d}})\right) \stackrel{!}{=} \left(z - z^{\mathrm{ESO}}\right)^3 \\ \det \begin{pmatrix} z + \beta_1 - 1 & \beta_1 \cdot T_{\mathrm{s}} - T_{\mathrm{s}} & \beta_1 T_{\mathrm{s}}^2/2 - T_{\mathrm{s}}^2/2 \\ \beta_2 & z + \beta_2 \cdot T_{\mathrm{s}} - 1 & \beta_2 T_{\mathrm{s}}^2/2 - T_{\mathrm{s}} \\ \beta_3 & \beta_3 T_{\mathrm{s}} & z + \beta_3 T_{\mathrm{s}}^2/2 - 1 \end{pmatrix} \end{gathered} \tag{4.33}$$

And therefore:

$$\beta_1 = 1 - \left(z^{\mathrm{ESO}}\right)^3, \quad \beta_2 = \frac{3}{2 \cdot T_{\mathrm{s}}} \cdot \left(1 - z^{\mathrm{ESO}}\right)^2 \cdot \left(1 + z^{\mathrm{ESO}}\right), \quad \beta_3 = \frac{1}{T_{\mathrm{s}}^2} \cdot \left(1 - z^{\mathrm{ESO}}\right)^3 \tag{4.34}$$

The same equations 4.20, 4.22 apply to disecrete ADRC controls, closed loop dynamics.

4.5.4 Simulation And Results Discrete ADRC

Simulation of discrete ADRC with a 2nd order system as an example is given :

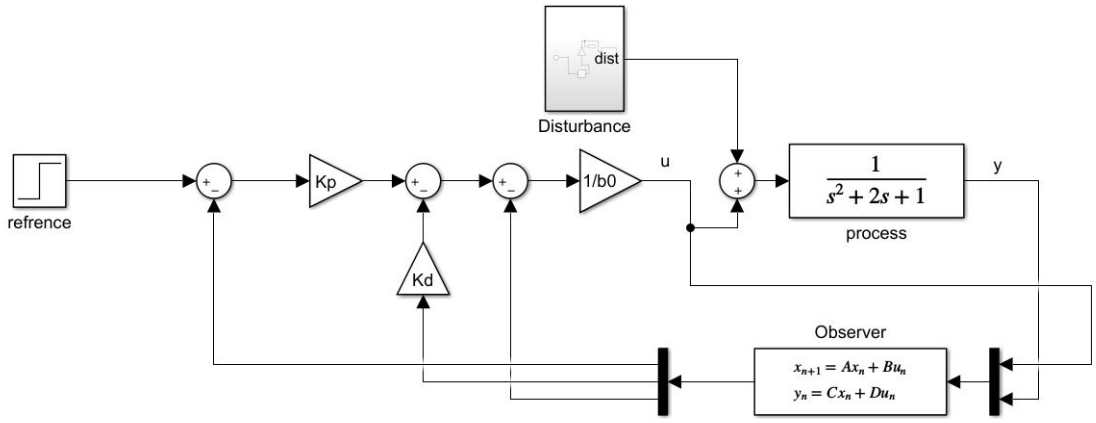
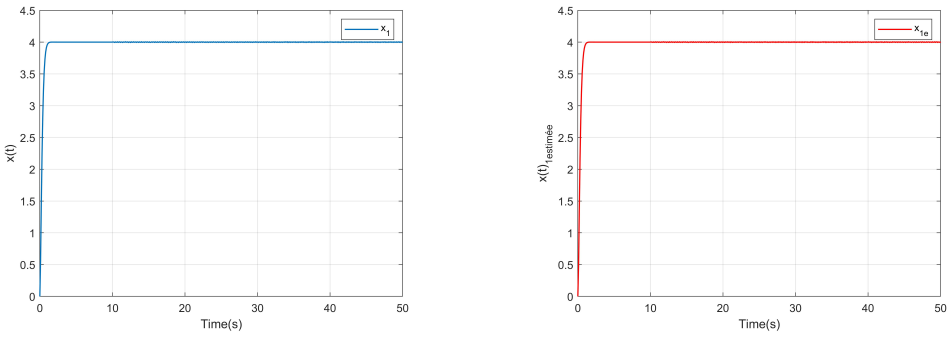


Figure 4.5: Simulink bloc digram of ADRC with a 2nd order

The ADRC will be designed assuming perfect knowledge, $b_0 = \frac{k}{T} = 1$, with $k^{ESO} = 5$, and a settling time, $T_r = 1$. The discretisation will be based on a sampling time $T_s = 0.01$. The parameters of the PD controller are, set to $K_P = 36$ and $K_D = 12$. After that a simulation is conducted to see the results of ADRC with a given disturbance:

Input disturbance was effective from t = 10.

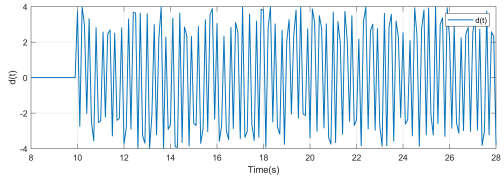


(a) Actual output (b) Estimated output

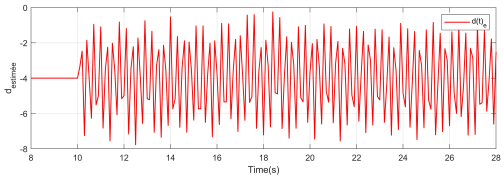
Figure 4.6: Step response and reaction on disturbance.

The simulation results in Fig.4.6 clearly demonstrate the ADRC's ability to reject disturbances.

The following Figures fig.4.7 present respectively the external disturbance applied on the controlled system, and the estimated disturbance.



(a) Actual disturbance



(b) Estimated disturbance

Figure 4.7: (a) Actual and (b) Estimated disturbances.

4.6 MPC With Discrete-Time Observer

We wish to estimate the disturbance to add to the MPC control, for this purpose. In this case, we will use a discrete time observer to estimate the disturbance.which is explained in detail in the section 4.4.1.

The concept is to create an independent discrete time observer for each disturbances We estimate only four disturbances, on quadrotor's positional and angular acceleration($\ddot{z}$ $\ddot{\phi}$ $\ddot{\theta}$ $\ddot{\psi}$) . We will add this discrete time observer in the second system design (Decentralized 2), which was discussed in detail in the previous section (3.4.2). The estimator was designed in this manner:

- The corresponding discrete time observer for disturbance 1 is then written:

$$\hat{X}_\theta(k) = A_{ESO} \cdot \hat{X}_\theta(k-1) + B_{ESO} \cdot u_\theta(k-1) + \beta_c \cdot y(k) \tag{4.35}$$

Where $\hat{X}_\theta$=$[\hat{\theta}\ \hat{\dot{\theta}}\ \hat{d}_\theta]^T$ are the state vectors of the discrete time observer, β_c = $[\beta_1\ \beta_2\ \beta_3]^T$ is the observer gain vector .

The matrices A_{ESO} and B_{ESO} are of the form:

$$A_{ESO} = (A_d - \beta_{\theta c} \cdot C_{\mathrm{d}} \cdot A_d)\,, B_{ESO} = (B_{\theta d} - \beta_{\theta c} \cdot C_{\mathrm{d}} \cdot B_{\theta d}) \tag{4.36}$$

A_d and $B_{\theta d}$ They are matrices A,B after discrete time ($T_s = 0.01$) ,$C_d = [1\ 0\ 0]$, $D_d = 0$.

Where :

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ b_{\theta,\phi,\psi,z} \\ 0 \end{bmatrix}$$

The other discrete time observers takes the same form as explained above, with the following changes:

$$\begin{aligned} \hat{X}_\phi(k) &= [\hat{\phi}\ \hat{\dot{\phi}}\ \hat{d}_\phi]^T, B_{ESO} = (B_{\phi d} - \beta_{\phi c} \cdot C_{\mathrm{d}} \cdot B_{\phi d})\,, u_\phi \\ \hat{X}_\psi(k) &= [\hat{\psi}\ \hat{\dot{\psi}}\ \hat{d}_\psi]^T, B_{ESO} = (B_{\psi d} - \psi_{\theta c} \cdot C_{\mathrm{d}} \cdot B_{\psi d})\,, u_\psi \\ \hat{X}_z(k) &= [\hat{z}\ \hat{\dot{z}}\ \hat{d}_z]^T, B_{ESO} = (B_{zd} - \beta_{zc} \cdot C_{\mathrm{d}} \cdot B_{zd})\,, u_z \end{aligned} \tag{4.37}$$

Then, we add these disturbances (d_i) estimated by a discrete-time observer to the MPC (inner loops), and the model in equation 3.1 after disturbances can be expressed as:

$$x(k+1) = A_i x_i(k) + B_i u_i(k) + d_i \tag{4.38}$$

With i= z ,θ ϕ ψ

The following figure shows a schematic representation of the closed-loop system with the discrete-time observer and MPC as a controller.

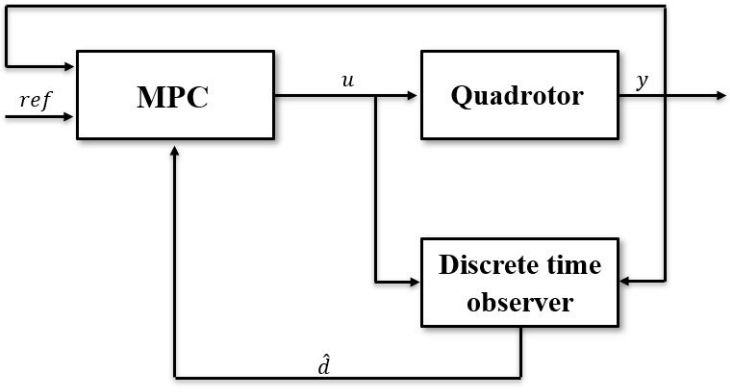


Figure 4.8: Schematic representation of the closed-loop system.

Where, y denotes the outputs, u are the control actions, and d are disturbance estimates.

4.7 Simulation

In this section, The system will be simulated with the discrete-time observers we designed and explained above. And we keep the same constraints as written in the table (3.3) .

The scaling parameters for the Q and R matrices are:

- In Outer loop 1,2 :$Q_{x,y} = 10$; $R_{x,y} = 1000$
- In inner loop 1,2 : $Q_{\theta,\phi} = 1000$; $R_{\theta,\phi} = 0.01$
- In Inner loop 3 $Q_{\psi} = 10; R_{\psi} = 0.1$
- In Inner loop 4 $Q_z = 500; R_z = 0.01$

The Observer Dynamics parameters values:

- $b_{\theta,\phi} = 0.1;\ b_z = 0.125;\ b_{\psi} = 0.0666$
- $T_s = 0.01;\ T_r = 2;\ s^{CL} = -\frac{6}{T_r}$
- $s^{ESO} = 5 \times s^{CL};\ K_d = -2 \times s^{CL};\ K_p = (s^{CL})^2;\ Z^{ESO} = e^{s^{ESO} \times T_s};$
- $\beta_1 = 1 - \left(z^{\mathrm{ESO}}\right)^3, \quad \beta_2 = \frac{3}{2 \cdot T_s} \cdot \left(1 - z^{\mathrm{ESO}}\right)^2 \cdot \left(1 + z^{\mathrm{ESO}}\right),$
  $\beta_3 = \frac{1}{T_s^2} \cdot \left(1 - z^{\mathrm{ESO}}\right)^3$

The observers gain parameters values:

$$\beta_{\theta,\phi\ \psi,z} = \begin{bmatrix} 0.3624 \\ 5.4153 \\ 27.0258 \end{bmatrix} \tag{4.39}$$

´

This Fig. 4.9 of MATLAB diagram shows the simple MPC with discrete-time observers control for the quadrotor.

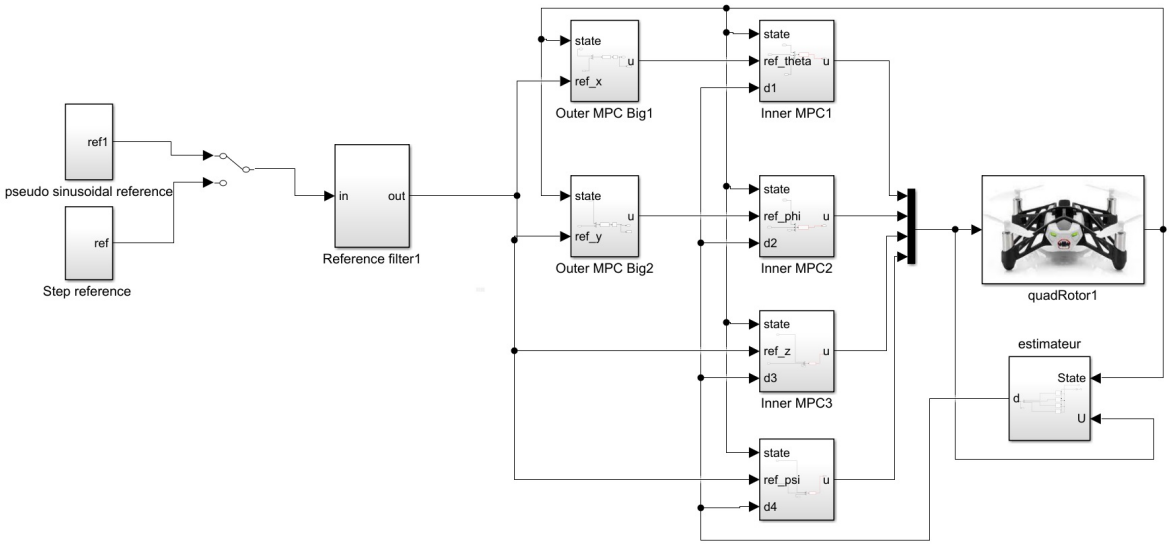


Figure 4.9: MPC MATLAB Simulink bloc diagram with Discrete-Time Observer .

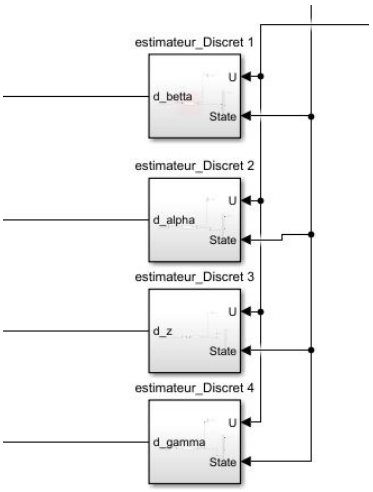
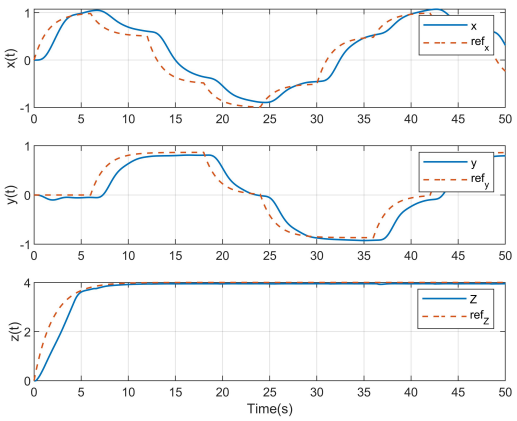


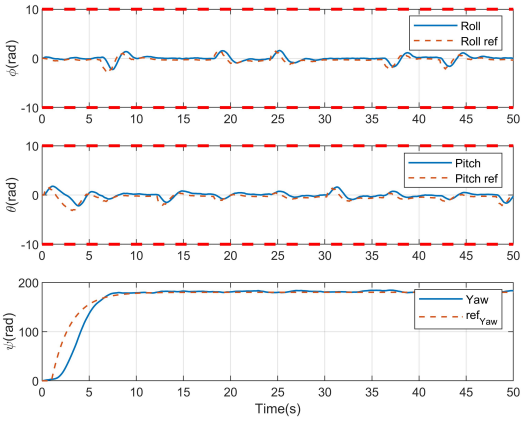
Figure 4.10: Illustration Figure for Discrete-Time Observer .

4.7.1 Simulation result

we can run the simulation and see the result :

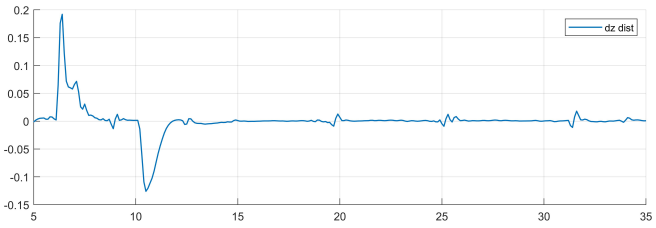


(a) The position (x ,y, z)

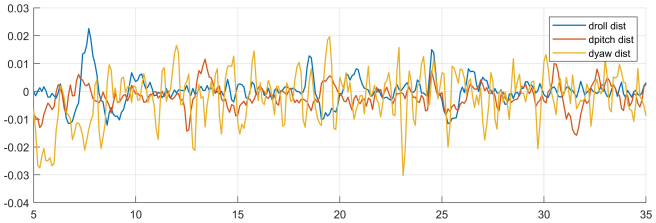


(b) The angular (ϕ, θ, ψ)

Figure 4.11: Simulation results with pseudo-sinusoidal reference.



(a) Disturbance (z)



(b) Disturbance (ϕ, θ, ψ)

Figure 4.12: Simulation results of the disturbance.

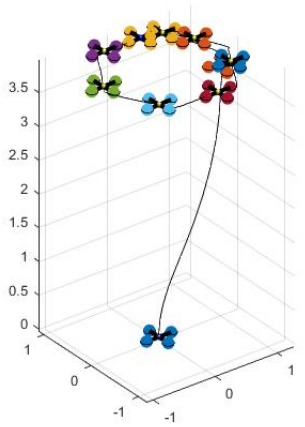


Figure 4.13: Circular trajectory in 3D with ADRC

4.8 Conclusion

In this chapter, we introduce various active disturbance rejection control concepts, such as the control system principle. We also discuss the two types of ADRC: linear ADRC and discrete time ADRC. Finally, we use a discrete-time observer to estimate disturbances that we add to the control MPC and simulate our system.

Chapter 5

General Conclusion

In this master project, the quadrotor trajectory tracking control using MPC was achieved in the presence of unknown external disturbances acting on the quadrotor.

Starting from the physical model of this system (quadcopter) which is MIMO nonlinear coupled system with four inputs and twelve state variables, we have obtained the linearized state-space model around the desired equilibrium point by linearizing the nonlinear model using Jacobian methods and this model was then decentralized to get decoupled linearized SISO systems that can be controlled independently to achieve trajectory tracking.

Model predictive control is an efficient tool that can achieve the tracking objective and minimize the control efforts and at the same time keep the control and state variables constrained at some predefined values. The MPC was applied successfully to the linearized decoupled subsystems of the drone based on two decentralization approaches.

Simulations on the proposed Model Predictive Control (MPC) approach for different trajectories (i.e. circle, step trajectory) was conducted and have proven that this approach is efficient and the trajectory objective was successful even under unknown external disturbances applied to the quadrotor and the constraints was always respected. These unknown disturbances were firstly estimated by using an estimator-based approach.

Finally, we can conclude. The fundamental advantage of the MPC controller over other controllers like PID, LQR, or feedback linearization is that it optimizes control inputs and outputs while taking into consideration disturbances, and constraints. This has been shown to help achieve proper inputs and outputs under certain requirements of the system.

As a recommendation for future work, we propose to work on realistic disturbances as an example the modeling of the wind for the simulations because we need to represent the influence of the wind as accurately as possible. The wind does not have a constant speed but is subject to variations that can be modeled by using particular mathematical tools. After the model allows us to simulate the wind speed and its variations over time, we will present the model allowing us to obtain the forces which are exerted on the quadrotor and its rotors. And we plan to establish a line-tracking algorithm for a mini-drone, which will also be used in real-world applications.

Bibliography

[1] Ye Wang, Andres Ramirez-Jaime, Feng Xu, and Vicenç Puig. Nonlinear Model Predictive Control with Constraint Satisfactions for a Quadcopter. Journal of Physics: Conference Series, 783:012025, January 2017.

[2] Yuan Yuan, Lei Cheng, Zidong Wang, and Chong Sun. Position tracking and attitude control for quadrotors via active disturbance rejection control method. Science China Information Sciences, 62(1):10201, December 2018.

[3] Sawyer, Shaun. Gain-Scheduled Control of a Quadcopter UAV. Master's thesis, UWSpace, 2015.

[4] Haifeng Lyu. Multivariable Control of a Rolling Spider Drone. PhD thesis, University of Rhode Island, Kingston, RI, 2017.

[5] Francesco Sabatino. Quadrotor control: modeling, nonlinearcontrol design, and simulation. 2015.

[6] Aerodynamic Force and Moment | Article about Aerodynamic Force and Moment by The Free Dictionary.

[7] Mohammed Belkheiri, Abdelhamid Rabhi, Ahmed El hajjaji, and C Pegard. Different linearization control techniques for a quadrotor system. pages 1–6, January 2012.

[8] Boumaza Hamza. Commande prédictive approximante. January 2017. Accepted: 2017-01-04T13:26:52Z Publisher: Université Constantine 1.

[9] Mohammed Belkheiri. Commande Avancée (contrôle optimal et prédictif) Master Automatique. 2019. Publisher: University of Amar Telidji Laghouat.

[10] J. Åkesson. MPCtools 1.0: Reference Manual. Institutionen för reglerteknik, Lunds tekniska högskola. Department of Automatic Control, Lund University, 2006.

[11] Colin Jones. Model predictive control course notes. 2020. Publisher: École Polytechnique Fédérale de Lausanne.

[12] Seif-El-Islam Hasseni and Latifa Abdou. Decentralized PID Control by Using GA Optimization Applied to a Quadrotor. Journal of Automation, Mobile Robotics and Intelligent Systems, 12(2):33–44, June 2018.

[13] Amri Lahcen. Contrôle ADRC de l'éolienne utilisant la machine asynchrone à double alimentation. Technical report, July 2020.

[14] Gernot Herbst. A Simulative Study on Active Disturbance Rejection Control (ADRC) as a Control Tool for Practitioners. Electronics, 2(4):246–279, August 2013.

[15] Addison-Wesley - Franklin G.F., Powell J.D., Workman M.L. - Digital Control of Dynamic Systems, 3E .pdf, 1997. publisher : Boston, MA, USA,.