A Comparative Study between Proportional-Derivative Control, Active Disturbance Rejection Control and Linear Active Disturbance Rejection Control For Trajectory Tracking of Quadcopter Unmanned Aerial Vehicle

Muazu Imran Wanka

19044881

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics to the Department of Engineering Design and Mathematics of the Faculty of Engineering at the University of the West of England

University of the West of England, Bristol

October 13, 2020

Abstract

This thesis studies the trajectory tracking control for quadcopter unmanned aerial vehicle (UAV). In order to guarantee the desired trajectory performance in the presence wind gusts, three distinct control algorithms are investigated, which are a proportional-derivative (PD) control strategy, classical active disturbance rejection control (ADRC) and a linear ADRC strategy respectively. This is done to establish the limitations of the PD controller and to establish the capabilities of the latter two control strategies to provide robust trajectory tracking performance. The nonlinear dynamical model of the quadcopter in the presence of wind gusts is firstly investigated by way of Newton-Euler methods. Consequently, the PD controller is designed for the position loop. The classical ADRC control strategy is designed for both the altitude and position loops respectively. Firstly, a tracking differentiator (TD) is used to obtain the differential signal of the desired trajectory. Secondly, an extended state observer (ESO) is leveraged to estimate the UAV quadcopter states such as the position, velocity, angular positions and lumped wind gusts disturbances which is compensated by a nonlinear feedback term in real time to facilitate system robustness. The distinct difference between the LADRC and the classical ADRC strategy is observed in two key differences. The use of a linear ESO (LESO) with linear parameters instead of the classical nonlinear based ESO. Lastly, the use of a PD control based strategy to compensate for the estimated state system dynamics is also leveraged instead of the nonlinear feedback term. Simulation results confirm the limitation of PD control algorithm in two key areas; the inability of the quadcopter UAV to track agile trajectories and the its inability to follow simple trajectories in the presence of modelled wind gust dynamics. Both the classical ADRC and LADRC exhibit conformity while tracking agile trajectories as well as showcasing a better trajectory tracking ability in the presence of wind gusts compared to the PD control strategy.

Contents

List of Figures

1 Project Motivation and Background

According to Abdulmajeed [2019] an Unmanned aerial vehicle (UAV) can be defined as a space-traversing vehicle that flies autonomously or remotely controlled without a human pilot or crew. UAVs have several applications that have proceeded military applications which they were originally designed for. Civilian applications such as but not limited to precision agriculture and aerial photography have been increasingly developed over the past decade Liu et al. [2013]. These UAV missions are facilitated by a series of spatial paths with or without temporal requirements which the UAVs are commanded to fly through Encarnaçao and Pascoal [2001]. These requirements are either trajectory tracking or path following flight patterns. Trajectory tracking entails the UAV being in a specific position in a prescribed time while path following necessitates the UAV to converge to a specific geometric path within a feasible speed profile Aguiar et al. [2005].

As noted in Ambrosino et al. [2009] most common UAV applications missions involve cruising along a pre-defined geometric path. This has formed the basis for more challenging missions such as stand-off tracking as in Frew et al. [2008] , small fixed-wing UAVs are particularly suitable for these sort of applications due to their low-cost and endurance property Yang et al. [2020]. The drawback however, of fixed-wing UAVs arises due to their susceptibility to adverse wind conditions, because as noted in Yang et al. [2020] wind velocity can form a significant percentage of the UAV operational velocity. This is critical because UAV flight control systems are responsible for facilitating mission critical functionalities to facilitate tedious tasks under extreme environmental flight conditions that are otherwise unsafe for piloted operation.

Small unmanned quadcopters are particularly useful for these sort of applications due to their distinct ability to manoeuvre around the environment in 3-dimensions (3D) using onboard sensors and even grippers in some instances. The current scope of the world means that most of the aforementioned activities are currently being carried-out by full size helicopters Abdulmajeed [2019]. Their relatively small nature means that they posses the inherent ability of operating within indoor and constrained environments. This is particularly useful in applications such as search and rescue missions when looking for survivors in fire ridden buildings and nuclear contamination sites and so on. This is made possible by using onboard capabilities such as sensors to create situational awareness within the context of the mission without endangering a physical human being Ambrosino et al. [2009].

An alternative to small unmanned quadcopters are unmanned fixed-wing UAVs which are primarily used for navigating over long distances efficiently due to to factors such as but not limited to energy consumption efficiency. However, as mentioned in Mellinger [2012] and Kotarski et al. [2016] quadcopters have the distinct ability to hover in one place which facilitates several advantages over fixed-wing UAVs including but not limited to the ability to precisely pick-up and deliver payload at an exact position and the ability to navigate otherwise compromising spaces.

It can be argued that ground robots also posses the ability to carry out some of the tasks mentioned above due to their ability to manoeuvre larger payloads, their robust interaction with environmental disturbances such as collision avoidance, in most cases do not require active sensing to manoeuvre the environment and the fact that they do not have to expend energy to maintain flight Mellinger [2012]. However, as mentioned in Mellinger [2012] and Dong et al. [2013] the inherent flying characteristic that quadcopters posses means that they are able to reach environments that ground robots cannot access. Furthermore, ground robot navigation problems such as stair climbing and rough terrain navigation are particularly complex problems for ground robots while these tasks are considered as trivial tasks quadcopters.

2 Literature Review

2.1 Motion Control with Trajectory Tracking

The problem of trajectory tracking has long been investigated and applied for various kinds of autonomous vehicles such as UAVs, autonomous under-water vehicles (AUVs) and so on. These vehicles are required to accurately track inertial frame reference trajectories in 3D space. A case in point is where Hopkins and Xu [2008] investigated the complex problem of position tracking for miniature helicopters using a multi-timescale robust controller. Control technique as they provide a certain degree of robustness to external disturbances such as wind Chowdhary et al. [2014]. Nonlinear control laws need to guarantee stability and other performance metrics to ensure the UAV tracks the desired trajectory under varied environmental conditions Xu et al. [2019]. The combined trajectory tracking and path-following of underactuated autonomous vehicles moving in 2-D and 3-D space was carried out by Aguiar and Hespanha [2007]. This was done using adaptive switching supervisory control together with a nonlinear Lyapunov based control law. The effectiveness of the controller was validated via simulation along several complex paths for hover-crafts and underwater vehicles. Furthermore, research conducted in Kaminer et al. [1998] explored the problem of trajectory guidance using integrated design of control systems for UAV Bluebird where the effectiveness of the strategy was validated via simulation.

The position control of nonholonomic wheeled robots was investigated by Wang et al. [2010] where a sliding-mode control (SMC) structure was implemented. The simulation results demonstrated adequate ability of tracking performance. As also seen in Bhatia et al. [2008] were controllers designed for Dubins vehicle were applied on commercial-off-the-shelf autopilots using Dubins trajectory. The results show that the controller is able to accurately track a desired Dubins trajectory through numerical simulation. The problem of combined trajectory tracking and path-following control was investigated by Encarnaçao and Pascoal [2001] for autonomous marine-craft where the results were validated via simulation using models of autonomous surface-craft and AUVs.

An SMC and a kinodynamic receding controller were both proposed for tracking desired sensor footprint in Jackson et al. [2008] which showcased robustness to disturbances and computational efficiency. The latter controller was found to track the sensor footprint and to be robust to flight constraints and obstacle avoidance, although at a considerable computational cost. Consequently, both controllers displayed the ability of being robust to model uncertainties and wind disturbances. An SMC approach was implemented for UAVs in Healey and Lienard [1993] but was found to be robust to model uncertainties but inaccurate in tight turning manoeuvres. An adaptive control approach in Cao et al. [2007] was used on an off the shelf autopilot to track a desired trajectory where the reparamatization of the the trajectory tracking controller produces a path following controller thus, enabling the tracking of a specified path as a trajectory tracking problem Aguiar et al. [2008].

Most robust control techniques for trajectory tracking such as in Healey and Lienard [1993] and Chowdhary et al. [2014] require exact knowledge of the plant dynamics and significant assumptions of model and disturbance uncertainties due to the apparent complexity of the problem. A study conducted in Liu et al. [2013] used a UAV equipped with an Ardupilot autopilot which was used to implement three separate proportional-derivative-integral (PID) controllers to achieve altitude-hold using elevator, airspeed-hold using the throttle and aileron and using the rudder to facilitate coordinated turn. This enabled the UAV to track the desired trajectory demand from the controller. Work conducted in Ren and Beard [2004] investigated the problem of constrained trajectory tracking control for small fixed-wing UAVs. Analysis of the proposed controller was carried out using Lyapunov techniques while the results were validated via simulation. Trajectory tracking control of a four-wheel differential mobile robot was proposed by Kaminer et al. [1998] where linearized control techniques were implemented to compensate for system uncertainty induced by soil dynamics.

Quadcopter UAV considered for the purposes of this project are light in structure and posses limited power properties which means that they are greatly affected by wind disturbances. A common method of reducing the effects of wind is overlooking the airspeed and focusing on its ground track, this enables the ground velocity and flight course to be used as feedback signals Kaminer et al. [2006] and Nelson et al. [2007]. On-board inertial navigation systems (INS) or more primitively GPS position are used to provide the information required. It is noted that for quadcopter UAVs equipped with low-cost sensors suit, the flight data quality may be inadequate in quality, moreover, the fast nature of this small vessels means that they are highly susceptible to latency in GPS feedback Liu et al.

[2013].

This is why for the purposes of this project, works are carried out under the pretence of assuming smooth airspeed measurement, the original GPS positional data to obtain guidance functionality as well as magnetic heading. An alternative approach of attenuating wind disturbances is explicitly incorporating in control algorithms as in Rysdyk [2006] which requires the exact knowledge of wind conditions. As shown in Chen [2004] and Liu et al. [2013] a nonlinear disturbance observer was implemented to provide an estimate of external wind disturbances which was then incorporated into the trajectory tracking controller which forms a composite controller for trajectory tracking. Adaptive trajectory tracking control was used in Zuo [2011] for the trajectory tracking problem on quadcopters. The proposed controller was designed for both the altitude loop and for position tracking. Analysis of the proposed controller was carried out using Lyapunov techniques while the results were validated via simulation. It is also noted that for these works, steady state or constant windy conditions were considered due to the fact that only these components cause steady state error. Furthermore the disturbance observers also posses the capability of providing estimates for other disturbances such as wind gusts errors which cause abnormal system behaviour Liu et al. [2012].

2.2 Active Disturbance Rejection Control Framework

The ADRC framework proposed by Jinquing Han outlined in Wang et al. [2015] is a model independent control architecture which is inherently error driven. It's best offering is found when control is not considered to be mathematical science, rather to be an experimental one. The fundamental difference between ADRC and some of the aforementioned control algorithms is that they all need exact knowledge of the plant dynamics or assumption of some some system uncertainties such as in Wang et al. [2015]. The primary motivation behind it is centred around the shortcomings of PID control. These problems are characterized by setpoint jumping, noise degradation in derivative control, loss of controller performance in the form of weighted sums and complications of integral control. Four distinct solution are presented which mitigate these issues. A transient profile generator is derived from a simple differential equation, a tracking differentiator that is noise tolerant, nonlinear feedback laws and an ESO for total disturbance estimation and rejection. The ADRC is designed in the form of these four combined PID shortcomings such as in Gong et al. [2012] ADRC is shown to solve four distinct control problems that are particularly challenging to PID. Firstly, three solutions to the time delay problem in control design methods are provided. These are transfer function approximation, predictive output feedback and predictive pseudo input respectively. The latter three are multivariable decoupling control, cascade control and parallel system control.

Several applications of ADRC can be found in literature which suggest ADRC can obtain robust performances of applications with uncertainties and disturbances Dong et al. [2013]. Case in point, Xingling and Honglun [2015] developed an attitude controller for small unmanned helicopters using the ADRC framework. The proposed approach was shown to be robust to uncertainties under control constraint. Furthermore, Madoński et al. [2014] implemented ADRC on a robotic-enhanced limb for rehabilitation training. This was shown to be effective for decoupled system modelling uncertainties thus providing a robust system performance. Finally, work carried out by Wang et al. [2015] designed three separate ADRC controllers for roll, pitch and speed control respectively for small fixed-wing UAVs. The unmodelled control dynamics were estimated using ESO which facilitated the robustness of each of the controllers. As shown in Li et al. [2013] where ADRC together with an SMC framework was implemented for an underactuated ship used for path following purposes in the presence of disturbances such as ocean currents and constant direction wind. It was established that the control framework was robust and eliminated the cross-track error caused by the disturbances mentioned above.

2.3 Active Disturbance Rejection Control for Quadcopters

As outlined in Zhang et al. [2019] where an SMC based ADRC structure was applied for the trajectory tracking of an underactuated six degree of freedom (6DOF) quadcopter in the presence of environmental disturbances. This makes use of an ESO to estimate the system uncertainties which was then compensated via the SMC. The stability of the closed-loop was carried out via Lyapunov based system analysis methods. Furthermore, Abdulmajeed [2019] proposed ADRC to facilitate the cancellation of undesirable dynamics such as aerodynamic disturbances and dynamic coupling using a PD based controller for quadcopter altitude. The results were proved via setpoint tracking plots which exhibited the fact the controller accurately tracks the setpoint. Precise trajectory tracking was achieved for UAVs using a combination of an ADRC based control strategy and nonlinear path-following guidance law by Zhang et al. [2015]. This was achieved by using a nonlinear guidance law for the trajectory guidance loop while the linear based ADRC structure was used to design the altitude loop, the results were consequently validated via numerical simulations.

Cascade based ADRC structure applied for quadcopter UAV which was compared to a conventional PID based cascade control according to Xu et al. [2019]. The results were validated via numerical simulation which showed that the the cascade ADRC performed better than the cascade PID. Work carried out by Xu et al. [2020] investigated the trajectory tracking of a UAV when considering system uncertainties and coupling factor based on ADRC. The proposed control structure was validated via simulation which exhibited favourable performance. Finally, a linear based ADRC structure was applied applied for stability control of quadcopters in the presence of wind disturbances in Ding and Wang [2018]. Finally, the use of a linear ESO was implemented to estimate wind induced uncertainties which were compensated by a PD controller. The results were proved via setpoint tracking plots which exhibited the fact the controller accurately tracks the setpoint while a method of parameter tuning as also introduced.

3 Project Scope

3.1 Project Objectives

The objective of the quadcopter trajectory tracking problem is to find a control law such that it enables the quadcopter to accurately track a desired geometrical path. As such the aim of this project is to develop and establish an effective robust control algorithm to facilitate the UAV trajectory tracking objective in the presence of wind gusts uncertainties. The objectives of this project are six-fold and are highlighted below as follows:

(1) To study and incorporate the nonlinear UAV quadcopter dynamics under the influence of wind gusts.

(2) To design and establish the limitation of a PD control strategy to provide trajectory tracking objectives under the influence of wind gusts and high speed agile manoeuvres.

(3) To study, design and implement a controller based on the classical nonlinear ADRC framework and it's perspective potential for solving the quadcopter trajectory tracking problem under the influence of wind gusts.

(4) To study, design and implement a controller based on the linear ADRC framework and it's perspective potential for solving the quadcopter trajectory tracking problem under the influence of wind gusts.

(5) To carry out extensive quantitative analysis and validation via MATLAB/Simulink to showcase the shortcomings of the PD based control strategy and the ability of the ADRC based strategies to mitigate these shortcomings.

(6) To carry out simulations to analyse the PD controller, nonlinear ADRC controller and linear ADRC controller respectively to facilitate pre-requisite project objectives.

3.2 Project Outline

This thesis is structured as follows: important findings and information obtained from literature will be highlighted and elaborated upon, together with the theoretical concepts associated with quadcopter trajectory tracking control. This will include but not limited to quadcopter characteristics and benefits, trajectory tracking control methods and an overview of the ADRC framework. The following chapter would investigate the modelling of the nonlinear quadcopter system dynamics under the influence of wind gusts. The following chapter would investigate controller design to facilitate closed-loop system performance to achieve trajectory tracking objectives. Controller design concepts of the PD control strategy would be provided together with an in-depth controller synthesis of the linear and nonlinear ADRC framework respectively. Numerical validation on the effectiveness of the all three control algorithms will be showcased through software simulation. Finally, discussion points and concluding remarks will be provided as well as an investigation into further works. It is noted that the timeline and all activities for this project are outlined in 43.

4 System Modelling

4.1 Quadcopter Dynamics

The quadcopter UAV is an underactuated system which is characterized by 6DOF mobility, four control inputs such that $U_i(i = 1, 2, 3, 4)$, inherent nonlinearity and so on. Highlighted in 1 is the coordinate system and free-body diagram of a 6DOF quadrotor as outlined in Mellinger [2012] and Zhang et al. [2019]. The world frame is depicted by O_e which is represented by axis X_e, Y_e, and Z_e. The body fixed frame O_b which is attached to the centre of mass of the quadcopter is depicted by X_b, Y_b, and Z_b. The forward direction of the quadcopter is donated by X_b while Z_b is perpendicular to the direction of the lift forces generated by the four rotors such that $F_i(i = 1, 2, 3, 4)$.

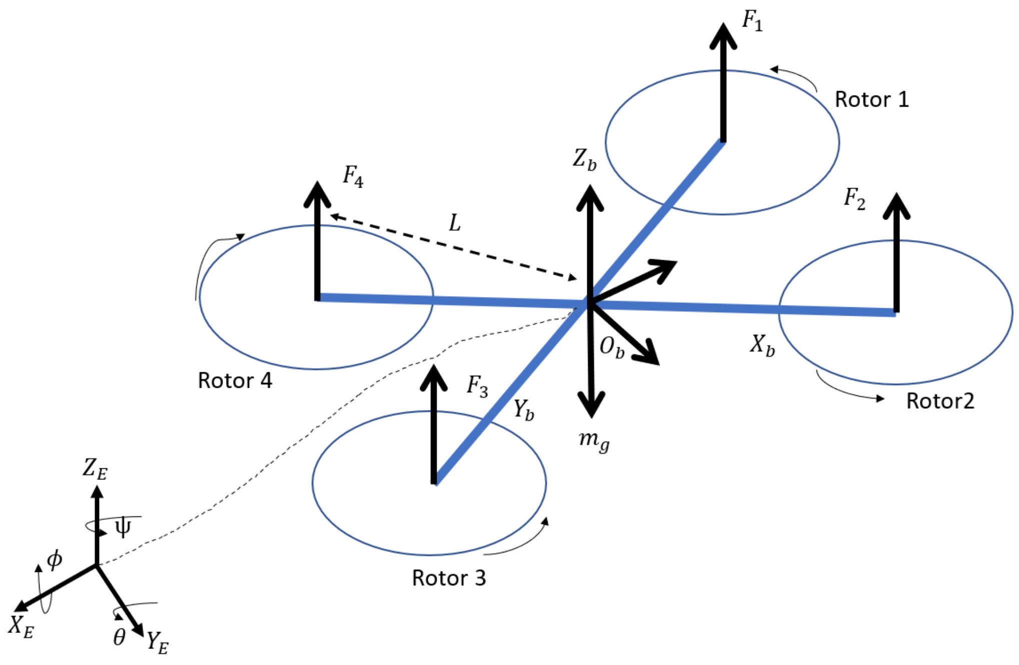


Figure 1: coordinate system with forces and moments acting on quadcopter frame

Furthermore, a moment is produced by each rotor, the first rotor is on the positive X_b, the second rotor is on the positive Y_b, the third on the negative X_b axis and the fourth is on the negative Y_b. The position loop of the quadcopter is defined by x, y and z respectively while the altitude loop is defined by ϕ, θ, ψ which are roll, pitch and yaw angles respectively. The mass of the quadcopter is denoted as m, L denotes the length between the centre of the aircraft and the rotor, I_{xx}, I_{yy} and I_{zz} are used to denote the rotational inertia in the

x, y and z axis respectively and g is used to denote the acceleration due to gravity. An assumption can be made that the quadcopter UAV is a rigid symmetrical body Mellinger [2012] and Zhang et al. [2019]. As such the mathematical model of the quadcopter UAV system dynamics can be expressed can be expressed using euler angles in six channels in 1.

$$\begin{aligned}
\ddot{x} &= \frac{1}{m}(F_x - k_f\dot{x}) \\
\ddot{y} &= \frac{1}{m}(F_y - k_f\dot{y}) \\
\ddot{z} &= \frac{1}{m}(F_z - mg - k_f\dot{z}) \\
\ddot{\phi} &= \frac{l(-F_1 - F_2 + F_3 + F_4)}{I}_{xx} + \frac{I_{yy} - I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} \\
\ddot{\theta} &= \frac{l(-F_1 + F_2 + F_3 - F_4)}{I_y} + \frac{I_{zz} - I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} \\
\ddot{\psi} &= \frac{(-F_1 + F_2 - F_3 + F_4)}{I_z} + \frac{I_{xx} - I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi}
\end{aligned} \tag{1}$$

Where:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = (F_1 + F_2 + F_3 + F_4) \begin{bmatrix} cos\psi\ sin\theta\ cos\phi + sin\psi\ sin\phi \\ sin\psi\ sin\theta\ cos\phi + sin\phi\ cos\psi \\ cos\theta\ cos\phi \end{bmatrix} \tag{2}$$

It is necessary to simplify the quadcopter flight system by introducing virtual control variables U_1, U_2, U_3 and U_4 which makes the quadcopter an underactuated system with four inputs and six channels. A relationship between these virtual control inputs and lift forces F_1, F_2, F_3, F_4 is depicted in 3 as shown in Zhang et al. [2019]. This enables the dynamic equation of the quadcopter to be expressed in 4 by combining 1, 2 and 3.

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ 0 & -k_tl & 0 & k_tl \\ -k_tl & 0 & k_tl & 0 \\ -k_m & k_m & -k_m & k_m \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} \tag{3}$$

Where k_t denotes the thrust coefficient, $w_i(i = 1, 2, 3, 4)$ denotes the angular velocity and k_m represents the antitorque coefficient.

$$
\begin{aligned}
\ddot{x} &= U1(cos\psi sin\theta cos\phi + sin\psi sin\phi) - \frac{k_f \dot{x}}{m} \\
\ddot{y} &= U1(sin\psi sin\theta cos\phi + cos\psi sin\phi) - \frac{k_f \dot{y}}{m} \\
\ddot{z} &= U1(cos\theta cos\phi - g) - \frac{k_f \dot{z}}{m} \\
\ddot{\phi} &= \frac{U_2}{I_x} + \frac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} \\
\ddot{\theta} &= \frac{U_3}{I_y} + \frac{I_z - I_x}{I_y}\dot{\phi}\dot{\psi} \\
\ddot{\psi} &= \frac{U_4}{I_z} + \frac{I_x - I_y}{I_z}\dot{\phi}\dot{\theta}
\end{aligned} \tag{4}
$$

It is noted that the position loop contains only one position input U_1, which enables the selection of intermediary values θ and ϕ to control the position loop indirectly as in 5.

$$
\begin{cases}
U_x = U_1(cos\psi sin\theta cos\phi + sin\psi sin\phi) \\
U_y = U_1(sin\psi sin\theta cos\phi + cos\psi cos\phi) \\
U_z = U_1(cos\phi cos\theta)
\end{cases} \tag{5}
$$

Solving 5 yields 6, 7 and 8 respectively.

$$
U_1 = \frac{U_z}{cos\phi cos\theta} \tag{6}
$$

$$
\theta = arctan(\frac{U_x cos\psi + U_y sin\psi}{U_z}) \tag{7}
$$

$$
\phi = arctan(cos\theta \frac{U_x sin\psi + U_y cos\psi}{U_z}) \tag{8}
$$

4.2 System Modelling with Wind Gust

Highlighted in 2 below Ding and Wang [2018] is the analytic depiction of wind gust dynamics acting on the quadcopter during realistic outdoor flight. When the aircraft is subjected to external wind disturbances it tends to yaw away from the desired trajectory hence, creating additional lateral air acting on the quadrotor propeller Ding and Wang [2018] and Dong et al. [2013].

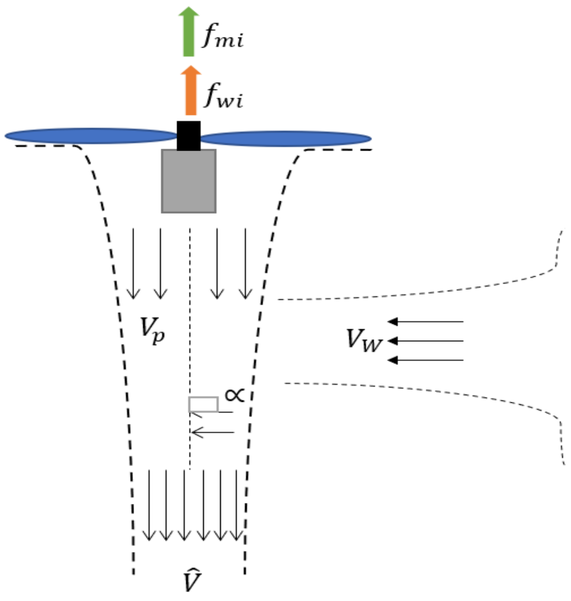


Figure 2: depiction of main and lateral thrusts

The vertical force which induces thrust is denoted by $f_{m_i} = k_t w_i^2$. This enables the total thrust to then be expressed as $f_{T_i} = f_{m_i} + f_{w_i}(i = 1, 2, 3, 4)$ in 9, where f_{w_i} represents the additional lateral forces induced by wind gusts.

$$f_{T_i} = 2\rho A\hat{V}V_p \tag{9}$$

Where ρ denotes air density, A represents the area of the propeller, V_p is propeller speed induced by wind and $\hat{V}$ is the total velocity of the rotor induced by wind, the relationship between these terms is outlined in 10.

$$\hat{V} = [(V_w cos\alpha + V_p)^2 + (V_w sin\alpha)^2]^{1/2} \tag{10}$$

The angle between the propeller axis and lateral wind gusts is donated by $\alpha = 90°$ because the lateral wind gusts act perpendicularly to the propeller axis. The velocity of the additional lateral wind gusts is donated as V_w which yields f_{w_i} in 11.

$$f_{w_i} = 2\rho AV_p^2(1 + \frac{V_w^2}{V_p^2})^{1/2} - f_{m_i} \tag{11}$$

Furthermore, aerodynamic drag m_{drag} can be defined as a function of the lateral wind gusts as $m_{drag} = \frac{\rho V_w^2}{2} = k_{drag}V_w^2$. Where k_{drag} is a positive constant > 0 which depends on several factors such as ρ, shape of propeller blade and so on. The additional torques acting on the quadcopter propellers due to lateral wind gusts can be shown in 12

$$\begin{bmatrix} m_{w_\phi} \\ m_{w_\theta} \\ m_{w_\psi} \end{bmatrix} = \begin{bmatrix} (f_{w_4} - f_{w_2})\, l \\ (f_{w_3} - f_{w_1})\, l \\ \sum_{i=1}^{4} m_{drag} \end{bmatrix} \tag{12}$$

Therefore combining 10, 11 and 12 the dynamics of the quadcopter can be rewritten according 13

$$\begin{aligned}
\ddot{x} &= U_1(cos\psi sin\theta cos\phi + sin\psi sin\phi) - \frac{k_f\dot{x}}{m} + \sum_{i=1}^{4}\frac{f_{w_i}}{m} \\
\ddot{y} &= U_1(sin\psi sin\theta cos\phi + cos\psi sin\phi) - \frac{k_f\dot{y}}{m} + \sum_{i=1}^{4}\frac{f_{w_i}}{m} \\
\ddot{z} &= U_1(cos\theta cos\phi - g) - \frac{k_f\dot{z}}{m} + \sum_{i=1}^{4}\frac{f_{w_i}}{m} \\
\ddot{\phi} &= \frac{U_2}{I_x} + \frac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} + \frac{m_{w_\phi}}{I_x} \\
\ddot{\theta} &= \frac{U_3}{I_y} + \frac{I_z - I_x}{I_y}\dot{\phi}\dot{\psi} + \frac{m_{w_\theta}}{I_y} \\
\ddot{\psi} &= \frac{U_4}{I_z} + \frac{I_x - I_y}{I_z}\dot{\phi}\dot{\theta} + \frac{m_{w_\psi}}{I_z}
\end{aligned} \tag{13}$$

4.3 Active Disturbance Rejection Controller Structure

Highlighted in 3 below is the block diagram of the ADRC topology which would form the basis for controller design in this thesis. The basis of this topology is formed by addressing the issues associated with PID based control strategies which was introduced by Jinquing Han in 2009. These solutions are presented in the form of four distinct solutions which are Tracking Differentiator, Nonlinear feedback Combination and an Extended State Observer Han [2009].

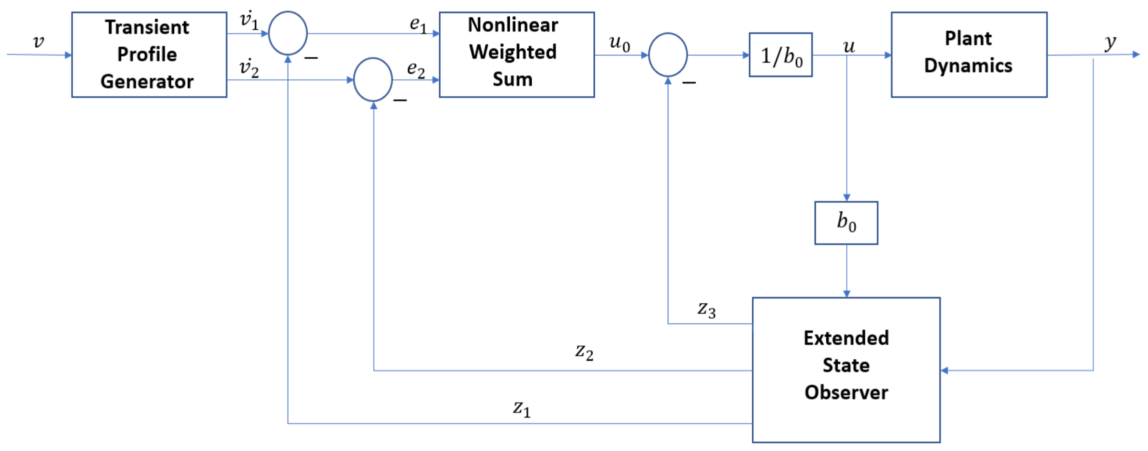


Figure 3: adrc topology

(A) Tracking Differentiator

To solve the issue of setpoint jumping a transient profile is implemented that which the plant can track. It is a matter of fact that for double integral plant as in 14.

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = u \end{cases} \tag{14}$$

where the $|u| \leq r$ and the desired value for x_1 is defined by v such that the time optimal solution for u is outlined in 15.

$$u = -r\text{sign}(x_1 - v + \frac{x_2|x_2|}{2r} \tag{15}$$

From 15 the transient profile generator is derived from the differential equation in 16 where the desired trajectory of transient profile generator is defined as v_1 while it's derivative is denoted as v_2. The parameter denoted as r is selected to speed up or slow down the transient profile as necessary.

$$\begin{cases} \dot{v_1} = v_2 \\ \dot{v_2} = -r\text{sign}(v_1 - v + \frac{v_2|v_2|}{2r} \end{cases} \tag{16}$$

It is however noted that the continuous-time optimal solution mentioned above introduces significant issues for discrete-time implementations, as such the double integral plant in 14 is transformed into a discrete-time solution in 17.

$$\begin{cases} v_1 = v_1 + hv_2 \\ v_2 = v_2 + hu, \quad |u| \leq \mathrm{r} \end{cases} \tag{17}$$

Therefore, the time-optimal solution which guarantees the fastest convergence of v_1 to v is defined in 18.

$$u = fhan(v_1 - v, v_2, r_0, h_0) \tag{18}$$

Where h is the sampling period and controller parameters are defined as r_0 and h_0 and the nonlinear function $fhan(v_1 - v, v_2, r_0, h_0)$ is defined below in 19.

$$fhan = -r_0(\frac{a}{d} - sign(a))s_a - r_0 sign(a), \tag{19}$$

Where:

$$\begin{cases} d = h_0 r_0^2, \qquad a_0 = h_0 v_2, \qquad y = v_1 + a_0 \\ a_1 = \sqrt{d(d + 8|y|)} \\ a_2 = a_0 + sign(y)(a_1 - d)/2 \\ s_y = (sign(y + d) - sign(y - d)/2 \\ a = (a_0 + y - a_2)s_y + a_2 \\ s_a = (sign(a + d) - sign(a - d)/2 \end{cases} \tag{20}$$

(B) Nonlinear Feedback Combination

ADRC employs the following nonlinear function to enable the tracking error to reach steady state quickly in finite time as shown in 21

$$fal\,(e,\ \alpha,\ \delta)\ = \begin{cases} \frac{e}{\delta^{1-a}} & |x| \leq \delta \\ |e|^{\alpha} sign(e), & |x| \geq \delta \end{cases} \tag{21}$$

The important role of the nonlinear functions fal and $fhan$ for the ADRC framework is clear to see. Furthermore, when nonlinear feedback is in the form of 22 with $\alpha < 1$ such that α mitigates the steady-state error significantly to the point that the integral term I in the PID controller can be omitted due to its shortcomings Li et al.

[2013] and manifesting as a PD controller. A case in point is an extreme scenario where $\alpha = 0$ which behaves like a bang-bang controller Han [2009].

$$u = |e|^{\alpha} sign(e) \tag{22}$$

(C) Total Disturbance Estimation and Rejection via ESO

This investigates the concept of total disturbance estimation and rejection which is applicable to most multi-input-multi-output (MIMO) time-varying systems in the type outlined in 23 where for the sake of brevity a generic single-input-single-output (SISO) system is depicted.

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = f(x_1, x_2, w(t), t) + bu \\ y = x_1 \end{cases} \tag{23}$$

Where y represents the system output to be tracked and controlled, the system input is depicted by u and the multivariable function $f(x_1, x_2, w(t), t)$ contains the the system states and system uncertainties Han [2009] which does not need to be explicitly known. This therein, represents the uniqueness of ADRC as $F(t) = f(x_1, x_2, w(t), t)$ "total disturbances" and can be overcome by a control signal. This enables the traditional problem of system identification such as in Yang et al. [2020] to be transformed to a disturbance rejection problem. As such $F(t)$ is treated as an additional variable such that $x_3 = F(t)$ and $\dot{F}(t) = G(t)$, this transforms the system in 23 to an always observable system in 24.

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = x_3 + bu \\ \dot{x_3} = G(t) \\ y = x_1 \end{cases} \tag{24}$$

The ESO is constructed in discrete form in 25 as in Xu et al. [2020].

$$\begin{cases} e = z_1 - y \\ fe = fal(e, 0.5, \delta), \qquad f_{e1} = fal(e, 0.25, \delta) \\ z_1 = z_1 + hz_2 - \beta_{01} e \\ z_2 = z_2 + h(z_3 + bu) - \beta_{02} fe \\ z_3 = z_3 - \beta_{03} fe_1 \end{cases} \tag{25}$$

The observer gains $\beta_0 1$, $\beta_0 2$ and $\beta_0 3$ in 25 can be selected according to Han [2009] in 26.

$$\beta_0 1 = 1 \qquad \beta_0 2 = \frac{1}{2h^0.5} \qquad \beta_0 3 = \frac{2}{5^2 h^{1.2}} \tag{26}$$

It is noted that the output y and input u act as input to the ESO as outlined in 3. The output of the ESO provides provides the estimate of $F(t) = f(x_1, x_2, w(t), t)$. The control law is formed as $\frac{u_0 - F(t)}{b}$ which enables the system in 23 to be transformed to a cascade double integral form in 27.

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = u_0 \\ y = x_1 \end{cases} \tag{27}$$

Where u_0 is a controller such as a PD controller which is a function of the tracking error and its derivative which actively compensates the for $F(t)$.

5 Controller Synthesis

5.1 Active Disturbance Rejection Controller Synthesis

An ADRC scheme as shown in 4 is developed for the UAV quadcopter according to the mathematical quadcopter model outlined in the previous section of this document, similar architectures are used in Gong et al. [2012], Xu et al. [2020] and Zhang et al. [2019]. As shown in the previous section the quadcopter can be determined to have six distinct channels which are divided into two loops. There two independent channels which are the altitude z channel and yaw ψ channel. There are two cascade subsystems which are the $x - \theta$ and $y - \phi$ channels. Initially, the reference trajectories x_d, y_d and z_d are used as desired positional inputs of the tracking differentiator depicted as TD_1, TD_2, TD_3 and TD_4. The outputs of the tracking differentiator are the desired positions and their derivatives which are leveraged as inputs for the PD controllers $NLSEF_1$, $NLSEF_2$ and $NLSEF_4$. The ADRC based nonlinear feedback is implemented on the translational x, y and z positions initially, which produces the virtual control inputs U_x, U_y, and U_z respectively. The commanded force of the quadcopter $U1$ and the desired θ_d and ϕ_d values are calculated according to equations 4, 5 and 6 respectively. Furthermore, the input to the nonlinear feedback $NLSEF_3$, $NLSEF_6$ and $NLSEF_5$ are desired θ_d, ϕ_d and ψ_d values which are Euler angle references. The commanded moments U_2, U_3, and U_4 are obtained to achieve UAV stabilization. Furthermore, the positional an altitude outputs of the UAV are leveraged as feedback signals into the the ESO $ES0_1$, $ES0_2$, $ES0_3$, $ES0_4$, $ES0_5$ and $ES0_6$ which are used to estimate the disturbances acting on the UAV. Finally the nonlinear feedback is used to actively compensate for the estimated disturbances by the ESO.

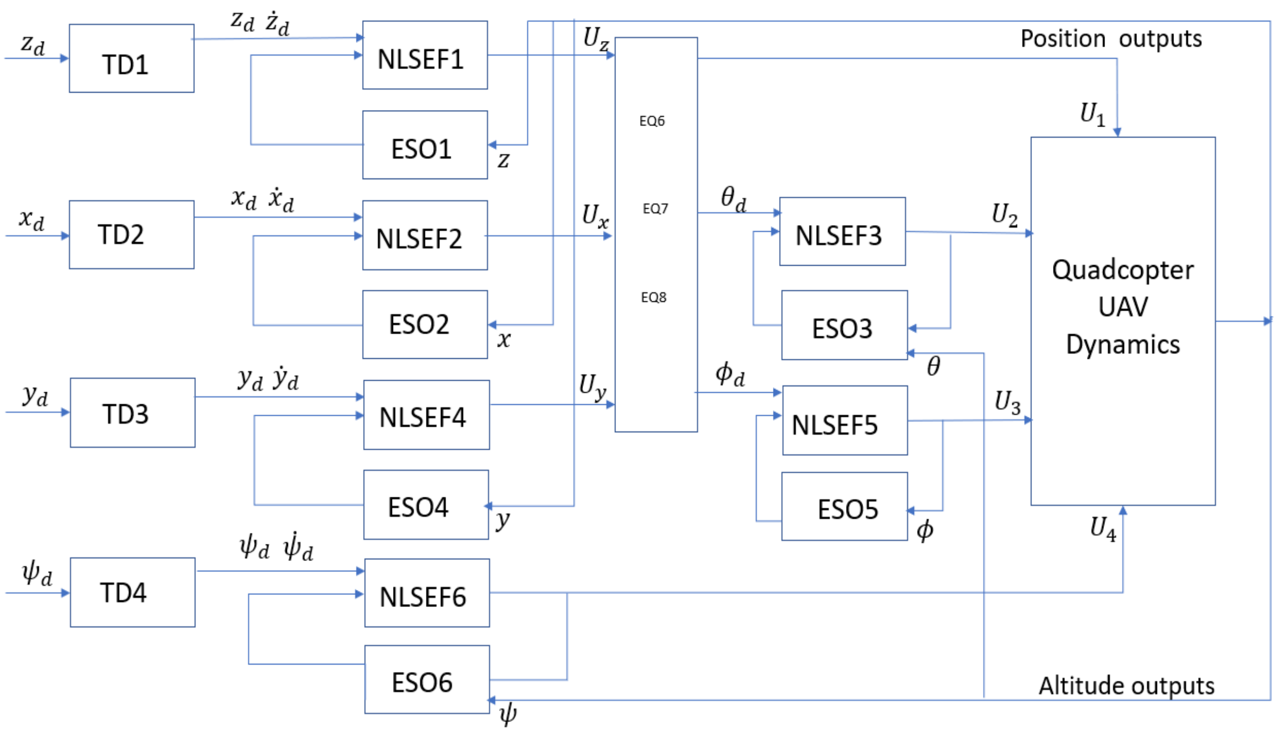


Figure 4: quadcopter adrc framework

5.1.1 Tracking Differentiator Design

As already established in the first section of this document, the quadrotor system is an underactuated system which entails of four control objectives. As such the TD is designed for the altitude z, x position, y position and yaw angle ψ. Below the design algorithms are outlined for the four channels mentioned above.

(A) Altitude Channel

The altitude channel TD is given below according to 28.

$$\begin{cases} \dot{v_{z1}} = v_{z2} \\ \dot{v_{z2}} = fhan(v_{z1} - z_d, v_{z2}, r, h) \end{cases} \tag{28}$$

Where v_{z1} is used to represent the tracking signal for the desired altitude height z_d, v_{z2} is the differential signal of v_{z1} which tracks the desired altitude velocity $\dot{z}_d$. The convergence speed of the signal is donated by r, h is the filtering factor and $fhan$ is a nonlinear function which denoted the optimal rapid control function outlined in 29.

$$\begin{cases} d = rh^2, \qquad a_0 = hv_{z2}, \qquad y = (v_{z1} - z_d) + a_0 \\ a_1 = \sqrt{d(d + 8|y|)} \\ a_2 = a_0 + sign(y)(a_1 - d)/2 \\ s_y = (sign(y + d) - sign(y - d)/2 \\ a = (a_0 + y - a_2)s_y + a_2 \\ s_a = (sign(a + d) - sign(a - d)/2 \\ fhan = -r(\frac{a}{d} - sign(a))s_a - rsign(a) \end{cases} \tag{29}$$

(B) X Channel

The x-channel TD is given below according to 30.

$$\begin{cases} \dot{v_{x1}} = v_{x2} \\ \dot{v_{x2}} = fhan(v_{x1} - x_d, v_{x2}, r, h) \end{cases} \tag{30}$$

Where v_{x1} is used to represent the tracking signal for the desired x position x_d, v_{x2} is the differential signal of v_{x1} which tracks the desired x velocity $\dot{x_d}$. The convergence speed of the signal is donated by r, h is the filtering factor and $fhan$ is a nonlinear function which denoted the optimal rapid control function outlined in 31.

$$\begin{cases} d = rh^2, \qquad a_0 = hv_{x2}, \qquad y = (v_{x1} - x_d) + a_0 \\ a_1 = \sqrt{d(d + 8|y|)} \\ a_2 = a_0 + sign(y)(a_1 - d)/2 \\ s_y = (sign(y + d) - sign(y - d)/2 \\ a = (a_0 + y - a_2)s_y + a_2 \\ s_a = (sign(a + d) - sign(a - d)/2 \\ fhan = -r(\frac{a}{d} - sign(a))s_a - rsign(a) \end{cases} \tag{31}$$

(C) Y Channel

The y-channel channel TD is given below according to 32.

$$\begin{cases} \dot{v_{y1}} = v_{y2} \\ \dot{v_{y2}} = fhan(v_{y1} - y_d, v_{y2}, r, h) \end{cases} \tag{32}$$

Where v_{y1} is used to represent the tracking signal for the desired y position y_d, v_{y2} is the differential signal of v_{y1} which tracks the desired y velocity $\dot{y_d}$. The convergence speed of the signal is donated by r, h is the filtering factor and $fhan$ is a nonlinear function which denoted the optimal rapid control function outlined in 33.

$$\begin{cases} d = rh^2, \qquad a_0 = hv_{y2}, \qquad y = (v_{y1} - y_d) + a_0 \\ a_1 = \sqrt{d(d + 8|y|)} \\ a_2 = a_0 + sign(y)(a_1 - d)/2 \\ s_y = (sign(y + d) - sign(y - d)/2 \\ a = (a_0 + y - a_2)s_y + a_2 \\ s_a = (sign(a + d) - sign(a - d)/2 \\ fhan = -r(\frac{a}{d} - sign(a))s_a - rsign(a) \end{cases} \tag{33}$$

(D) Yaw Channel

The yaw angle channel TD is given below according to 34.

$$\begin{cases} \dot{v_{\psi 1}} = v_{\psi 2} \\ \dot{v_{\psi 2}} = fhan(v_{\psi 1} - y_d, v_{\psi 2}, r, h) \end{cases} \tag{34}$$

Where $v_{\psi 1}$ is used to represent the tracking signal for the desired yaw angle ψ_d, $v_{\psi 2}$ is the differential signal of $v_{\psi 1}$ which tracks the desired yaw rate $\dot{\psi_d}$. The convergence speed of the signal is donated by r, h is the filtering factor and $fhan$ is a nonlinear function which denoted the optimal rapid control function outlined in 35.

$$\begin{cases} d = rh^2, \qquad a_0 = hv_{\psi 2}, \qquad y = (v_{\psi 1} - \psi_d) + a_0 \\ a_1 = \sqrt{d(d + 8|y|)} \\ a_2 = a_0 + sign(y)(a_1 - d)/2 \\ s_y = (sign(y + d) - sign(y - d)/2 \\ a = (a_0 + y - a_2)s_y + a_2 \\ s_a = (sign(a + d) - sign(a - d)/2 \\ fhan = -r(\frac{a}{d} - sign(a))s_a - rsign(a) \end{cases} \tag{35}$$

5.1.2 Extended State Observer Design

A state observer denoted as an ESO in discrete form is leveraged for both the altitude loop and positional loop as shown in 4 for x, y, z, ϕ, θ and ψ respectively below.

(A) Altitude Channel

Outlined in 36 below is the ESO design for the altitude channel. The states of the altitude channel are represented as z_{z1}, z_{z2} and z_{z3} which denote the altitude position, altitude velocity and lumped disturbances acting on the altitude respectively as shown in 13. The estimation error of the ESO on the altitude is denoted as e_z, u_1 is used to denote the the commanded force input of the altitude and the observer gains are donated as $\beta_0 1$, $\beta_0 2$ and $\beta_0 3$ respectively. Finally, fal is a nonlinear feedback function which is essential in ADRC synthesis and h is used to denote the sampling period.

$$\begin{cases} e_z = z_{z1} - y_z \\ fe = fal(e_z, 0.5, h), \qquad fe1 = fal(e_z, 0.25, h) \\ z_{z1} = z_{z1} + hz_{z2} - \beta_{01} e_z \\ z_{z2} = z_{z2} + h(z_{z3} + b_0 u_1) - \beta_{02} fe \\ z_{z3} = z_{z3} - \beta_{03} fe_1 \end{cases} \tag{36}$$

(B) X Channel

Outlined in 37 below is the ESO design for the x-channel. The states of the x-position are represented as z_{x1}, z_{x2} and z_{x3} which denote the x-position, x-velocity and lumped disturbances acting on x-channel respectively as shown in 13. The estimation error of the ESO on the altitude is denoted as e_x, u_1 is used to denote the the commanded force input of the x-position and the observer gains are donated as $\beta_0 1$, $\beta_0 2$ and $\beta_0 3$ respectively. Finally, fal is a nonlinear feedback function which is essential in ADRC synthesis and h is used to denote sampling period.

$$
\begin{cases}
e_x = z_{x1} - y_x \\
fe = fal(e_x, 0.5, h), \qquad fe1 = fal(e_x, 0.25, h) \\
z_{x1} = z_{x1} + hz_{x2} - \beta_{01}e_x \\
z_{x2} = z_{x2} + h(z_{x3} + b_0u_1) - \beta_{02}fe \\
z_{x3} = z_{x3} - \beta_{03}fe_1
\end{cases}
\tag{37}
$$

(C) Y Channel

Outlined in 38 below is the ESO design for the y-channel. The states of the y-position are represented as z_{y1}, z_{y2} and z_{y3} which denote the x-position, y-velocity and lumped disturbances acting on y-position as shown in 13. The estimation error of the ESO on the altitude is denoted as e_y, u_1 is used to denote the the commanded force input of the y-position and the observer gains are donated as β_01, β_02 and β_03 respectively. Finally, fal is a nonlinear feedback function which is essential in ADRC synthesis and h is used to denote sampling period.

$$
\begin{cases}
e_y = z_{y1} - y_y \\
fe = fal(e_y, 0.5, h), \qquad fe1 = fal(e_y, 0.25, h) \\
z_{y1} = z_{y1} + hz_{y2} - \beta_{01}e_y \\
z_{y2} = z_{y2} + h(z_{y3} + b_0u_1) - \beta_{02}fe \\
z_{y3} = z_{y3} - \beta_{03}fe_1
\end{cases}
\tag{38}
$$

(D) Pitch Channel

Outlined in 39 below is the ESO design for the pitch angle channel. The states of the pitch angle channel are represented as $z_{\theta 1}$, $z_{\theta 2}$ and $z_{\theta 3}$ which denote the pitch angle, pitch angular rate and lumped disturbances acting on the pitch channel as shown in 13. The estimation error of the ESO on the pitch channel is denoted as e_θ, u_3 is used to denote the the commanded momentum input influenced by the pitch angle while the observer gains are donated as β_01, β_02 and β_03 respectively. Finally, fal is a nonlinear feedback function which is essential in ADRC synthesis and h is used to denote sampling period.

$$\begin{cases} e_\theta = z_\theta - y_\theta \\ fe = fal(e_\theta, 0.5, h), \qquad fe1 = fal(e_\theta, 0.25, h) \\ z_{\theta 1} = z_{\theta 1} + hz_{\theta 2} - \beta_{01} e_\theta \\ z_{\theta 2} = z_{\theta 2} + h(z_{\theta 3} + b_0 u_2) - \beta_{02} fe \\ z_{\theta 3} = z_{\theta 3} - \beta_{03} fe_1 \end{cases} \tag{39}$$

(E) Roll Channel

Outlined in 40 below is the ESO design for the roll angle channel. The states of the roll angle channel are represented as $z_{\phi 1}$, $z_{\phi 2}$ and $z_{\phi 3}$ which denote the roll angle, roll angular rate and lumped disturbances acting on the roll channel as shown in 13. The estimation error of the ESO on the roll channel is denoted as e_ϕ, u_2 is used to denote the the commanded momentum input influenced by the roll angle while the observer gains are donated as $\beta_0 1$, $\beta_0 2$ and $\beta_0 3$ respectively. Finally, fal is a nonlinear feedback function which is essential in ADRC synthesis and h is used to denote sampling period.

$$\begin{cases} e_\phi = z_{\phi 1} - y_\phi \\ fe = fal(e_\phi, 0.5, h), \qquad fe1 = fal(e_\phi, 0.25, h) \\ z_{\phi 1} = z_{\phi 1} + hz_{\phi 2} - \beta_{01} e_\phi \\ z_{\phi 2} = z_{\phi 2} + h(z_{\phi 3} + b_0 u_2) - \beta_{02} fe \\ z_{\phi 3} = z_{\phi 3} - \beta_{03} fe_1 \end{cases} \tag{40}$$

(F) Yaw Channel

Outlined in 41 below is the ESO design for the yaw angle channel. The states of the yaw angle channel are represented as $z_{\psi 1}$, $z_{\psi 2}$ and $z_{\psi 3}$ which denote the yaw angle, yaw angular rate and lumped disturbances acting on the yaw channel as shown in 13. The estimation error of the ESO on the yaw channel is denoted as e_ψ, u_4 is used to denote the the commanded momentum input influenced by the yaw angle while the observer gains are donated as $\beta_0 1$, $\beta_0 2$ and $\beta_0 3$ respectively. Finally, fal is a nonlinear feedback function which is essential in ADRC synthesis and h is used to denote sampling period.

$$\begin{cases} e_{\psi} = z_{\psi 1} - y_{\psi} \\ fe = fal(e_{\psi}, 0.5, h), \qquad fe1 = fal(e_{\psi}, 0.25, h) \\ z_{\psi 1} = z_{\psi 1} + hz_{\psi 2} - \beta_{01} e_{\psi} \\ z_{\psi 2} = z_{\psi 2} + h(z_{\psi 3} + b_0 u_4) - \beta_{02} fe \\ z_{\psi 3} = z_{\psi 3} - \beta_{03} fe_1 \end{cases} \tag{41}$$

5.1.3 Nonlinear State Error Feedback

The Nonlinear State Error Feedback (NLSEF) is used to actively compensate for the estimation error in the ESO stated in the earlier parts of this section. Just as above the NLSEF will be designed for all six individual channels as stated above.

(A) Altitude Channel

Outlined in 42 below is the NLSEF design for the altitude channel. The control action u_z is used to compensate for the commanded thrust U_1 of the quadcopter. The main tuning parameters for the controller are denoted as r, c, h_1 and b_0 respectively. The amplification coefficient is denoted as r, the damping coefficient is denoted as c, the aggressiveness of the control loop is represented by the precision coefficient denoted by h_1 which is a multiple of the sampling period h stated above and b_0 represents an approximation of the coefficient of parametric plant parameter b.

$$\begin{cases} e_{z1} = v_{z1} - z_{z1}, \qquad e_{z2} = v_{z2} - z_{z2} \\ u_z = \frac{-fhan(e_{z1}, ce_{z1}, r, h_1) + z_{z3}}{b_0} \end{cases} \tag{42}$$

(B) X Channel

Stated in 43 below is the NLSEF design for the x-channel. The control action u_x is used to compensate for the commanded thrust U_1 of the quadcopter. The main tuning parameters for the controller are denoted as r, c, h_1 and b_0 respectively. The amplification coefficient is denoted as r, the damping coefficient is denoted as c, the aggressiveness of the control loop is represented by the precision coefficient denoted by h_1 which is a multiple of the sampling period h stated above and b_0 represents an approximation of the coefficient of parametric plant parameter b.

$$\begin{cases} e_{x1} = v_{x1} - z_{x1}, \qquad e_{x2} = v_{x2} - z_{x2} \\ u_x = \frac{-fhan(e_{x1}, ce_{x1}, r, h_1) + z_{x3}}{b_0} \end{cases} \tag{43}$$

(C) Y Channel

Outlined in 44 below is the NLSEF design for the y-channel. The control action u_y is used to compensate for the commanded thrust U_1 of the quadcopter. The main tuning parameters for the controller are denoted as r, c, h_1 and b_0 respectively. The amplification coefficient is denoted as r, the damping coefficient is denoted as c, the aggressiveness of the control loop is represented by the precision coefficient denoted by h_1 which is a multiple of the sampling period h stated above and b_0 represents an approximation of the coefficient of parametric plant parameter b.

$$\begin{cases} e_{y1} = v_{y1} - z_{y1}, \qquad e_{y2} = v_{y2} - z_{y2} \\ u_y = \frac{-fhan(e_{y1}, ce_{y1}, r, h_1) + z_{y3}}{b_0} \end{cases} \tag{44}$$

(D) Roll Channel

Stated in 45 below is the NLSEF design for the roll angle channel. The virtual control u_2 is used to compensate for UAV stabilization momentum. The main tuning parameters for the controller are denoted as r, c, h_1 and b_0 respectively. The amplification coefficient is denoted as r, the damping coefficient is denoted as c, the aggressiveness of the control loop is represented by the precision coefficient denoted by h_1 which is a multiple of the sampling period h stated above and b_0 represents an approximation of the coefficient of parametric plant parameter b.

$$\begin{cases} e_{\phi1} = v_{\phi1} - z_{\phi1}, \qquad e_{\phi2} = v_{\phi\phi2} - z_{\phi2} \\ u_3 = \frac{-fhan(e_{\phi1}, ce_{\phi1}, r, h_1) + z_{\phi3}}{b_0} \end{cases} \tag{45}$$

(E) Pitch Channel

Highlighted in 46 below is the NLSEF design for the pitch angle channel. The virtual control u_3 is used to compensate for UAV stabilization momentum. The main tuning parameters for the controller are denoted as r, c, h_1 and b_0 respectively. The amplification coefficient is denoted as r, the damping coefficient is denoted as c, the aggressiveness of the control loop is represented by the precision coefficient denoted by h_1 which is a multiple of the sampling period h stated above and b_0 represents an approximation of the coefficient of parametric plant parameter b.

$$\begin{cases} e_{\theta 1} = v_{\theta 1} - z_{\theta 1}, \qquad e_{\theta 2} = v_{\theta 2} - z_{\theta 2} \\ u_3 = \frac{-fhan(e_{\theta 1}, ce_{\theta 1}, r, h_1) + z_{\theta 3}}{b_0} \end{cases} \tag{46}$$

(F) Yaw Channel

Stated in 47 below is the NLSEF design for the yaw angle channel. The virtual control u_4 is used to compensate for UAV stabilization momentum. The main tuning parameters for the controller are denoted as r, c, h_1 and b_0 respectively. The amplification coefficient is denoted as r, the damping coefficient is denoted as c, the aggressiveness of the control loop is represented by the precision coefficient denoted by h_1 which is a multiple of the sampling period h stated above and b_0 represents an approximation of the coefficient of parametric plant parameter b.

$$\begin{cases} e_{\psi 1} = v_{\psi 1} - z_{\psi 1}, \qquad e_{\psi 2} = v_{\psi 2} - z_{\psi 2} \\ u_4 = \frac{-fhan(e_{\psi 1}, ce_{\psi 1}, r, h_1) + z_{\psi 3}}{b_0} \end{cases} \tag{47}$$

5.2 Linear Active Disturbance Rejection Control

The LADRC structure for the quadcopter for each input-output channel in **??** can be represented by a second-order derivative relation as in 23 which leverages an unknown system disturbance state.

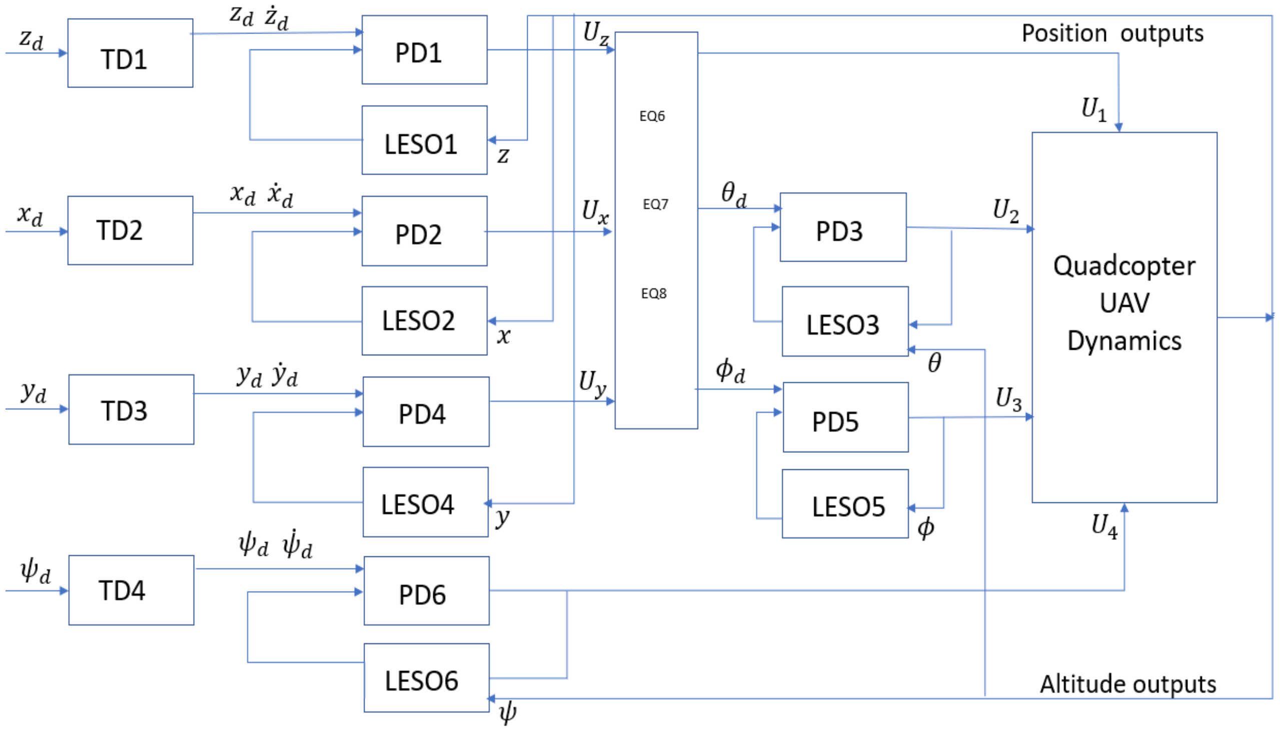


Figure 5: quadcopter linear adrc framework

The LADRC consists of a Linear extended-state observer (LESO) which as opposed to the nonlinear structure of the ESO used earlier for the classical ADRC structure. Finally, an ADRC based PD control structure would be used to ensure the convergence of the LESO as opposed to the NLSEF used above. The unknown disturbance state is estimated with the aid of the LESO based state space model in the form of 48. A similar approach was carried out in Li et al. [2013] and Ding and Wang [2018].

$$\begin{aligned}\dot{x} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \underline{x} + \begin{bmatrix} 0 \\ b_0 \\ 0 \end{bmatrix} \underline{u} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \underline{h} \\ \text{y} &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \underline{x}\end{aligned} \tag{48}$$

5.2.1 Second-order System Dynamics

All the individual sub-system channels are denoted using LADRC below.

(A) Altitude Channel

Highlighted in 49 below is the LADRC system for the altitude z channel, where z_1 is used to denote UAV altitude height, z_2 is used to denote the velocity, z_3 is an added augmented state while its derivative h is an unknown disturbance state. The virtual control input for obtaining commanded thrust force U_1 is giving as u_z and y is used to represent the output of the system.

$$\begin{cases} \dot{z_1} = z_2 \\ \dot{z_2} = z_3 + bu_z \\ \dot{z_3} = h \\ y = z_1 \end{cases} \tag{49}$$

(B) X Channel

Highlighted in 50 below is the LADRC system for the x channel, where x_1 is used to denote quadcopter x position component, x_2 is used to denote the velocity component of x, x_3 is an added augmented state while its derivative h is an unknown system disturbance state. The virtual control input for obtaining commanded thrust force U_1 is giving as u_x and y is used to represent the output of the system.

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = x_3 + bu_x \\ \dot{x_3} = h \\ y = x_1 \end{cases} \tag{50}$$

(C) Y Channel

Highlighted in 51 below is the LADRC system for the y channel, where y_1 is used to denote quadcopter y position component, y_2 is used to denote the velocity component of y, y_3 is an added augmented state while its derivative h is an unknown system disturbance state. The virtual control input for obtaining commanded thrust force U_1 is giving as u_y and y is used to represent the output of the system.

$$\begin{cases} \dot{y_1} = y_2 \\ \dot{y_2} = y_3 + bu_y \\ \dot{y_3} = h \\ y = y_1 \end{cases} \tag{51}$$

(D) Roll Channel

Highlighted in 52 below is the LADRC system for the ϕ channel, where ϕ_1 is used to denote the roll component of the quadcopter dynamics, ϕ_2 is used to denote the roll angular rate of the quadcopter, ϕ_3 is an added augmented state while its derivative h is an unknown system disturbance state. The virtual control input for quadcopter moment stability is given as u_phi and y is used to represent the output of the system.

$$\begin{cases} \dot{\phi_1} = \phi_2 \\ \dot{\phi_2} = \phi_3 + bu_\phi \\ \dot{\phi_3} = h \\ y = \phi_1 \end{cases} \tag{52}$$

(E) Pitch

Highlighted in 53 below is the LADRC system for the θ channel, where θ_1 is used to denote the pitch component of the quadcopter dynamics, θ_2 is used to denote the pitch angular rate of the quadcopter, θ_3 is an added augmented state while its derivative h is an unknown system disturbance state. The virtual control input for quadcopter moment stability is given as u_θ and y is used to represent the output of the system.

$$\begin{cases} \dot{\theta_1} = \theta_2 \\ \dot{\theta_2} = \theta_3 + bu_\theta \\ \dot{\theta_3} = h \\ y = \theta_1 \end{cases} \tag{53}$$

(F) Yaw

Highlighted in 54 below is the LADRC system for the ψ channel, where ψ_1 is used to denote the yaw component of the quadcopter dynamics, ψ_2 is used to denote the yaw angular rate of the quadcopter, ψ_3 is an added augmented state while its derivative h is an unknown system disturbance state. The virtual control input for quadcopter moment stability is given as u_ψ and y is used to represent the output of the system.

$$\begin{cases} \dot{\psi}_1 = \psi_2 \\ \dot{\psi}_2 = \psi_3 + bu_\psi \\ \dot{\psi}_3 = h \\ y = \psi_1 \end{cases} \tag{54}$$

5.2.2 Linear Extended State Observer

A LESO is constructed for 49, 50, 51, 52, 53 and 54 to estimate parametric system states and uncertainties as these are now states in extended state model. Below the structure of the LESO for all the individual system channels is provided.

(A) Altitude Channel

Stated in 55 is the LESO design structure for the z altitude channel. From 49, the inputs to the LESO are obtained as u_z and y respectively while $\hat{z} = [\hat{z}_1 \quad \hat{z}_2 \quad \hat{z}_3]^T$ are such that they provide the estimate for the states outlined in 49.

$$\begin{cases} \dot{\hat{z}}_1 = \hat{z}_2 - L_1(\hat{z}_1 - z_1) \\ \dot{\hat{z}}_2 = \hat{z}_2 - L_2(\hat{z}_1 - z_1) + bu_z \\ \dot{\hat{z}}_3 = -L_3(\hat{z}_1 - z_1) \end{cases} \tag{55}$$

The observer gain parameters are such that $L_i, i = 1, 2, 3$ which need to be chosen to facilitate the convergence of the LESO. For the sake of simplicity tuning the observer within this document would be similar to the approach used Li et al. [2013]. Where the poles of the observer are placed at $-w_0$ where w_0 is the bandwidth of the observer. Therefore, the characteristic polynomial of 55 in the Laplace domain is outlined in 56 below which yields the observer gains such that $L = [3w_0 \quad 3w_0^2 \quad w_0^3]$.

An observer with a larger bandwidth would be characterized by a more accurate estimation error although it is sensitive to noise. It is noted that the bandwidth of the observer is chosen as trade-off between tracking performance of the observer and the elimination of noise. Upon tuning the observer the system states as yielded as output.

$$s^3 + L_1 s^2 + L_2 s + L_3 = (s + w_0)^3 \tag{56}$$

Finally, it is worth noting that observer gain parameters would only be discussed for the altitude channel for the sake of brevity as the design procedure for choosing observer gains is the same for each channel of the quadcopter.

(B) X Channel

Stated in 57 is the LESO design structure for the x channel. From 50, the inputs to the LESO are obtained as u_x and y respectively while $\hat{x} = [\hat{x_1} \quad \hat{x_2} \quad \hat{x_3}]^T$ are such that they provide the estimate for the states outlined in 50.

$$\begin{cases} \dot{\hat{x_1}} = \hat{x_2} - L_1(\hat{x_1} - x_1) \\ \dot{\hat{x_2}} = \hat{x_2} - L_2(\hat{x_1} - x_1) + bu_x \\ \dot{\hat{x_3}} = -L_3(\hat{x_1} - x_1) \end{cases} \tag{57}$$

(C) Y Channel

Shown in 58 is the LESO design structure for the y channel. From 51, the inputs to the LESO are obtained as u_y and y respectively while $\hat{y} = [\hat{y_1} \quad \hat{y_2} \quad \hat{y_3}]^T$ are such that they provide the estimate for the states outlined in 51.

$$\begin{cases} \dot{\hat{y_1}} = \hat{y_2} - L_1(\hat{y_1} - y_1) \\ \dot{\hat{y_2}} = \hat{y_2} - L_2(\hat{y_1} - y_1) + bu_y \\ \dot{\hat{y_3}} = -L_3(\hat{y_1} - y_1) \end{cases} \tag{58}$$

(D) Roll Channel

Outlined in 59 is the LESO design structure for the ϕ roll channel. From 52, the inputs to the LESO are obtained as u_ϕ and y respectively while $\hat{\phi} = [\hat{\phi_1} \quad \hat{\phi_2} \quad \hat{\phi_3}]^T$ are such that they provide the estimate for the states outlined in 52.

$$\begin{cases} \dot{\hat{\phi}}_1 = \hat{\phi}_2 - L_1(\hat{\phi}_1 - \phi_1) \\ \dot{\hat{\phi}}_2 = \hat{\phi}_2 - L_2(\hat{\phi}_1 - \phi_1) + bu_\phi \\ \dot{\hat{\phi}}_3 = -L_3(\hat{\phi}_1 - \phi_1) \end{cases} \tag{59}$$

(E) Pitch Channel

Stated in 60 is the LESO design structure for the θ pitch channel. From 53, the inputs to the LESO are obtained as u_z and y respectively while $\hat{\theta} = [\hat{\theta}_1 \quad \hat{\theta}_2 \quad \hat{\theta}_3]^T$ are such that they provide the estimate for the states outlined in 53.

$$\begin{cases} \dot{\hat{\theta}}_1 = \hat{\theta}_2 - L_1(\hat{\theta}_1 - \theta_1) \\ \dot{\hat{\theta}}_2 = \hat{\theta}_2 - L_2(\hat{\theta}_1 - \theta_1) + bu_\theta \\ \dot{\hat{\theta}}_3 = -L_3(\hat{\theta}_1 - \theta_1) \end{cases} \tag{60}$$

(F) Yaw Channel

Stated in 61 is the LESO design structure for the ψ yaw channel. From 54, the inputs to the LESO are obtained as u_z and y respectively while $\hat{\psi} = [\hat{\psi}_1 \quad \hat{\psi}_2 \quad \hat{\psi}_3]^T$ are such that they provide the estimate for the states outlined in 54.

$$\begin{cases} \dot{\hat{\psi}}_1 = \hat{\psi}_2 - L_1(\hat{\psi}_1 - \psi_1) \\ \dot{\hat{\psi}}_2 = \hat{\psi}_2 - L_2(\hat{\psi}_1 - \psi_1) + bu_\psi \\ \dot{\hat{\psi}}_3 = -L_3(\hat{\psi}_1 - \psi_1) \end{cases} \tag{61}$$

5.2.3 ADRC Based PD Controller Design

The ADRC based PD control law is proposed for the purpose of actively compensating for the effects of the unknown parametric disturbances mentioned above. The controller gain parameters are chosen that $s^2 + k_d s + k_p$ is Hurwitz. Where k_p and k_d are controller parameters. The PD control structure for all six channels is presented below. The PD controller gains are tuned iteratively according to Ziegler Nichols method of PID gain tuning as shown in Copeland [2008].

(A) Altitude Channel

The PD based ADRC control law for the altitude channel u_z is outlined 62. The estimation error e_z and its differential $\dot{e_z}$ is denoted as $e_z = \hat{z_1} - z_d$ and $\dot{e_z} = \dot{\hat{z_1}} - \dot{z_d}$ respectively.

$$u_{0z} = -k_p e_z - k_d \dot{e_z} \tag{62}$$

As such the PD control law for the altitude channel is defined in 63 which drives the error dynamics $e_z(t) \rightarrow 0$ as $t \rightarrow \infty$.

$$u_z = \frac{u_{0z} - z_3}{b_0} \tag{63}$$

(B) X Channel

The PD based ADRC control law for the x channel u_x is outlined 64. The estimation error e_x and its differential $\dot{e_x}$ is denoted as $e_x = \hat{x_1} - x_d$ and $\dot{e_x} = \dot{\hat{x_1}} - \dot{x_d}$ respectively.

$$u_{0x} = -k_p e_x - k_d \dot{e_x} \tag{64}$$

As such the PD control law for the x channel is defined in 65 which drives the error dynamics $e_x(t) \rightarrow 0$ as $t \rightarrow \infty$.

$$u_x = \frac{u_{0x} - x_3}{b_0} \tag{65}$$

(C) Y Channel

The PD based ADRC control law for the y channel u_y is outlined 66. The estimation error e_y and its differential $\dot{e_y}$ is denoted as $e_y = \hat{y_1} - y_d$ and $\dot{e_y} = \dot{\hat{y_1}} - \dot{y_d}$ respectively.

$$u_{0y} = -k_p e_y - k_d \dot{e_y} \tag{66}$$

As such the PD control law for the y channel is defined in 67 which drives the error dynamics $e_y(t) \rightarrow 0$ as $t \rightarrow \infty$.

$$u_y = \frac{u_{0y} - y_3}{b_0} \tag{67}$$

(D) Roll Channel

The PD based ADRC control law for the roll channel u_ϕ is outlined 68. The estimation error e_ϕ and its differential $\dot{e_\phi}$ is denoted as $e_\phi = \hat{\phi}_1 - \phi_d$ and $\dot{e_\phi} = \dot{\hat{\phi}}_1 - \dot{\phi}_d$ respectively.

$$u_{0\phi} = -k_p e_\phi - k_d \dot{e_\phi} \tag{68}$$

As such the PD control law for the roll channel is defined in 69 which drives the error dynamics $e_\phi(t) \to 0$ as $t \to \infty$.

$$u_\phi = \frac{u_{0\phi} - \phi_3}{b_0} \tag{69}$$

(E) Pitch Channel

The PD based ADRC control law for the pitch channel u_θ is outlined 70. The estimation error e_θ and its differential $\dot{e_\theta}$ is denoted as $e_\theta = \hat{\theta}_1 - \psi_d$ and $\dot{e_\theta} = \dot{\hat{\theta}}_1 - \dot{\theta}_d$ respectively.

$$u_{0\theta} = -k_p e_\theta - k_d \dot{e_\theta} \tag{70}$$

As such the PD control law for the pitch channel is defined in 71 which drives the error dynamics $e_\theta(t) \to 0$ as $t \to \infty$.

$$u_\theta = \frac{u_{0\theta} - \theta_3}{b_0} \tag{71}$$

(F) Yaw Channel

The PD based ADRC control law for the yaw channel u_ψ is outlined 72. The estimation error e_ψ and its differential $\dot{e_\psi}$ is denoted as $e_\psi = \hat{\psi}_1 - \psi_d$ and $\dot{e_\psi} = \dot{\hat{\psi}}_1 - \dot{\psi}_d$ respectively.

$$u_{0\psi} = -k_p e_\psi - k_d \dot{e_\psi} \tag{72}$$

As such the PD control law for the yaw channel is defined in 73 which drives the error dynamics $e_\psi(t) \to 0$ as $t \to \infty$.

$$u_\psi = \frac{u_{0\psi} - \psi_3}{b_0} \tag{73}$$

5.3 Proportional-Derivative Controller Synthesis

The PD control strategy presented in this document is similar to the back-stepping control architecture presented in Mellinger [2012] and Xu et al. [2019] as highlighted in 6. The altitude control loop is used to track trajectories that are close to the nominal hover state of the quadcopter donated by roll angle ϕ, pitch angle θ and yaw angle ψ res. The position control loop is used to estimate the position and velocity of the desired trajectory in 3D. The trajectory controller used to maintain the quadcopter at a desired x_d, y_d, z_d and ψ_d orientation. Within the scope of this section and other sections referencing this section, only position loop control would be evaluated. The positions y, x and z leverage U_1 as control input while the yaw angle ψ is controlled by U_4. It is noted that the desired trajectory is denoted by $p_d = [x_d \quad y_d \quad x_d \quad \psi_d]$.

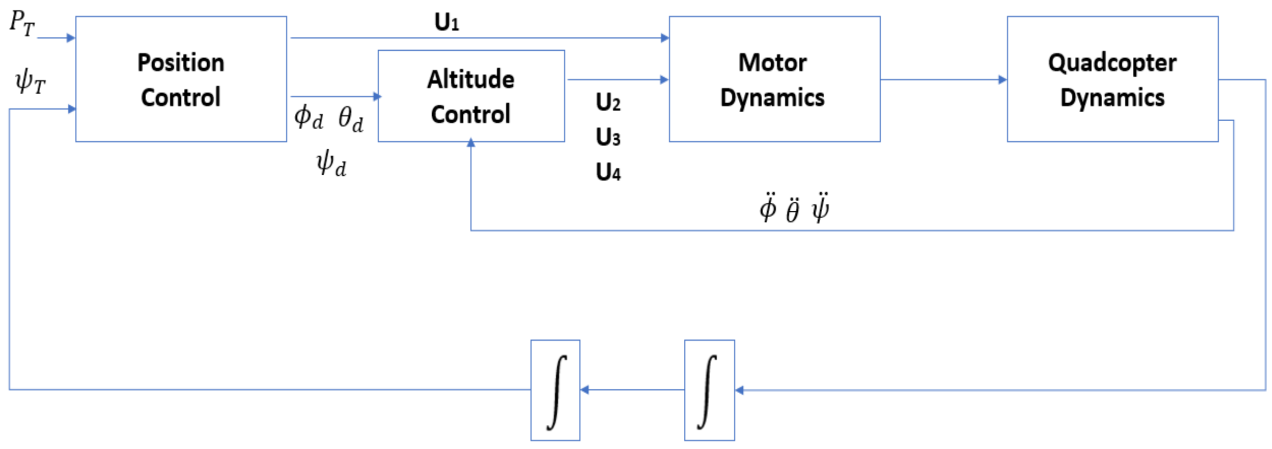


Figure 6: pd control system architecture

The 3D trajectory control objective is to calculate the closest point on the trajectory such that the position error is given as $e_i = p_i - p_{i,d}$ as in Mellinger [2012]. As such like in Xu et al. [2020] the unit tangent vector of the trajectory associated with that particular point be denoted as $\hat{t}$, this enables the desired velocity to be consequently defined as $\dot{p_T}$. Furthermore, this yields the position and velocity errors in in 74 and 75 as e_r and e_v respectively. It is noted that error is only considered in the normal $\hat{n}$ and bi-normal direction $\hat{b}$, therefore, tangential error is ignored.

$$e_r = ((p_T - p)\ .\hat{n})\hat{n} + ((p_T - p)\ .\hat{b})\hat{b} \tag{74}$$

$$e_v = \dot{p_T} - \dot{p} \tag{75}$$

The acceleration commanded output $\ddot{p_d}$ can be calculated from the PD feedback of position error such that $e_i = p_i - p_{i,d}$ as shown in 76. It is noted that an assumption can be made that for near hovering conditions $\psi_d = 0$ this approach is similar to the ones proposed in Xingling and Honglun [2015] and Zhang et al. [2015].

$$\ddot{p_{d,i}} - \ddot{p_i} = k_{p,i}(p_{d,i} - p_i) + k_{d,i}(\dot{p_{d,i}} - p_i) \tag{76}$$

6 Simulation Setting, Results and Discussion

6.1 Simulation Procedure

The numerical simulation for the purposes of trajectory tracking and controller performance for all three proposed controllers is carried out via MATLAB and Simulink. Firstly, an investigation into establishing the limitation of the PD based controller via extended scenario testing is carried out. Furthermore, the capability of both the ADRC based controllers to provide a better performance than the PD controller is investigated within the bounds of chosen testing scenarios. The simulation parameters of the quadcopter are such that: $m = 0.03kg$, $L = 0.046m$, $g = 9.81\frac{m}{s^2}$, $I_{xx}, I_{yy} = 0.00143\frac{kg}{m^2}$ and $I_{zz} = 0.00289\frac{kg}{m^2}$. The model of the motor being used in this document for the purposes of simulation is similar to that outlined in Mellinger [2012].

6.2 Proportional-Derivative Controller

The PD control structure would be evaluated under three distinct test scenarios. Firstly, the limitation of the controller is highlighted by enabling the quadcopter to track a circular type trajectory. This was experimentally carried out such that the velocity of the quadcopter was varied from $2\frac{m}{s} \rightarrow 10\frac{m}{s}$. Furthermore, the quadcopter was made to track a spiral trajectory for velocities ranging from $0.5\frac{m}{s} \rightarrow 3\frac{m}{s}$. The final experiment was conducted for the same circular trajectory in the presence of wind gusts in the quadcopter dynamics for velocities between $2\frac{m}{s} \rightarrow 6\frac{m}{s}$.

6.2.1 Circular Trajectory at Varied Velocities

The PD controller was tested according to a spiral trajectory which can be defined as $x_d = 10cos(0.1t)m$, $y_d = 10sin(0.1t)m$, $z_d = 0.1tm$ and $\psi_d = \frac{\pi}{6}rads$. The initial states of the quadcopter are given such that $[x \quad y \quad z]$ and $[\dot{x} \quad \dot{y} \quad \dot{z}] = [0 \quad 0 \quad 0]$ and $[0 \quad 0 \quad 0]$ respectively.

As shown in 7 below the quadcopter is made to track a 3-D circular trajectory at $2\frac{m}{s}$. As can be seen quadcopter accurately tracks the trajectory of the quadcopter. Outlined in 8a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.018m$ which occurs at $t = 16s$. The maximum error for the y component of trajectory is $\approx 0.033m$ while he maximum error for the z component of trajectory is $\approx 0.031m$ which also occurs at the same time frame as above. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [5 \quad 5 \quad 20]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [5 \quad 5 \quad 10]$.

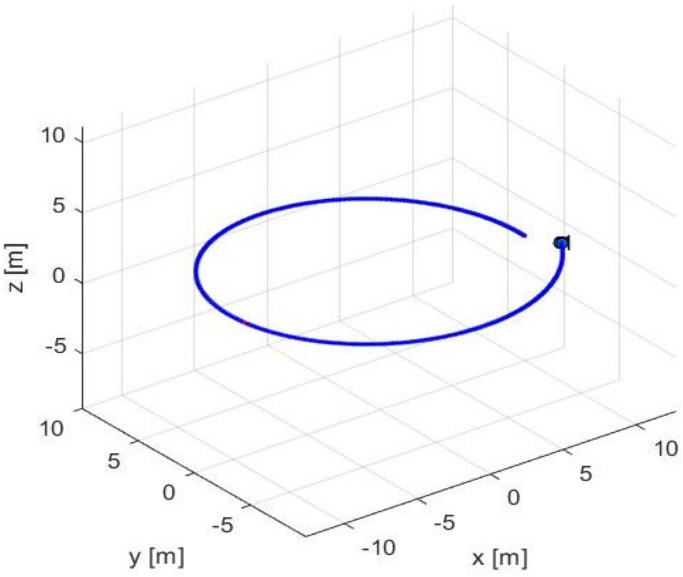
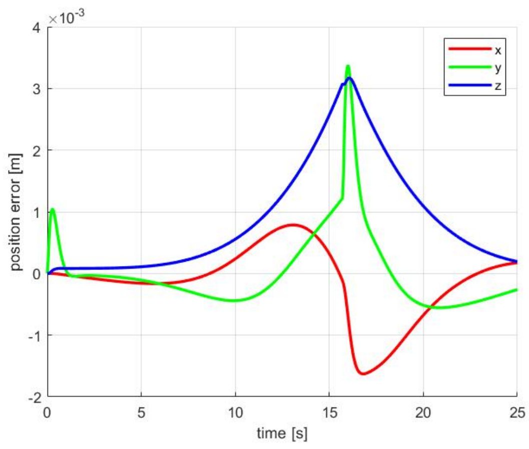
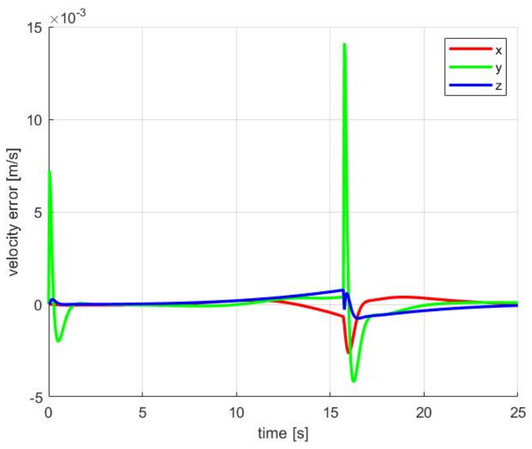


Figure 7: circular trajectory at 2m/s without disturbance

Outlined in 8b is the velocity tracking error between the desired path velocity states $[\dot{x}_d \quad \dot{y}_d \quad \dot{z}_d]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 0.0013\frac{m}{s}$ which occurs at $t = 16s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 0.014\frac{m}{s}$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 0.001\frac{m}{s}$ which also occurs at the same time frame as above.



(a) position error in x, y, z [m]



(b) velocity error in x, y, z [m/s]

Figure 8: trajectory tracking error at 2m/s

As shown in 9 below the quadcopter is made to track a 3-D circular trajectory at $4\frac{m}{s}$. As can be seen the quadcopter accurately tracks the desired trajectory. Outlined in 10a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.045m$ which occurs at $t = 8s$. The maximum error for the y component of trajectory is $\approx 0.011m$ while he maximum error for the z component of trajectory is $\approx 0.045m$ which also occurs at the same time frame as above.

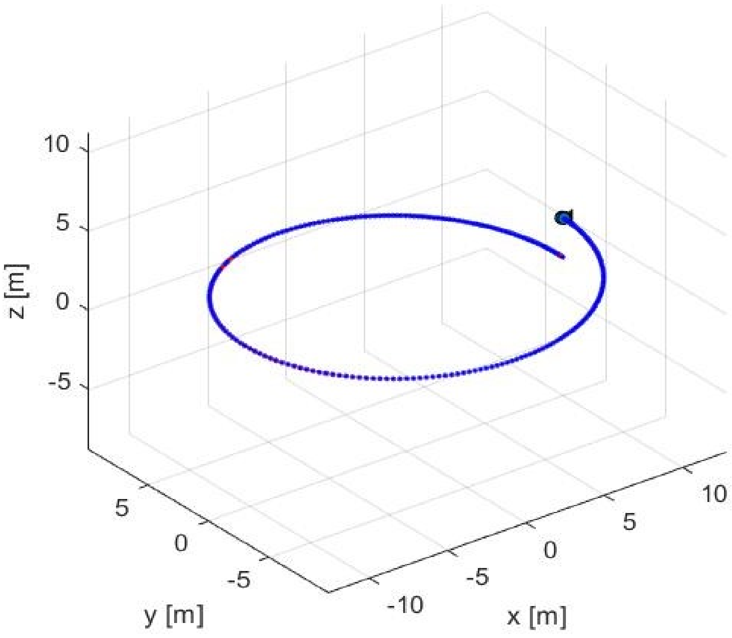
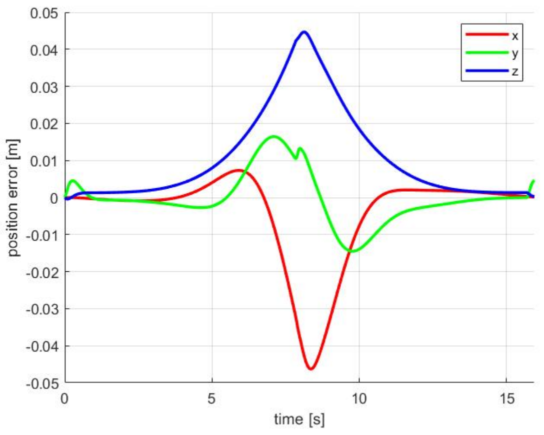
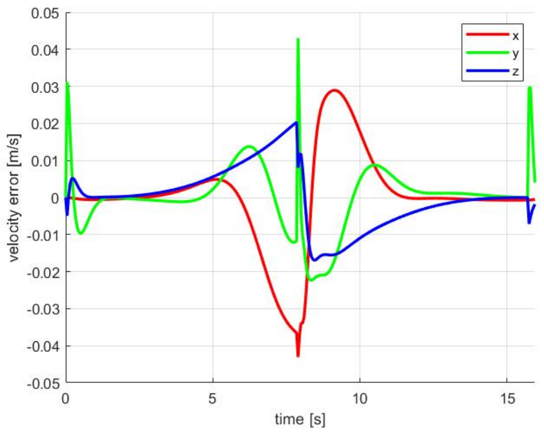


Figure 9: circular trajectory at 4m/s without disturbance

Highlighted in 10b is the velocity tracking error between the desired path velocity states $[\dot{x}_d \quad \dot{y}_d \quad \dot{z}_d]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 0.04\frac{m}{s}$ which occurs at $t = 8s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 0.041\frac{m}{s}$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 0.02\frac{m}{s}$ which also occurs at the same time frame as above. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [5 \quad 5 \quad 18]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [6 \quad 6 \quad 12]$.



(a) position error in x, y, z [m]



(b) velocity error in x, y, z [m/s]

Figure 10: trajectory tracking error at 4m/s

As shown in 11 below the quadcopter is made to track a 3-D circular trajectory at $6\frac{m}{s}$. As can be seen the quadcopter accurately tracks the desired trajectory with minimal error. Outlined in 12a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.25m$ which occurs at $t = 6s$. The maximum error for the y component of trajectory is $\approx 0.1m$ at $5.5s$ while he maximum error for the z component of trajectory is $\approx 0.21m$ which also occurs at $6s$.

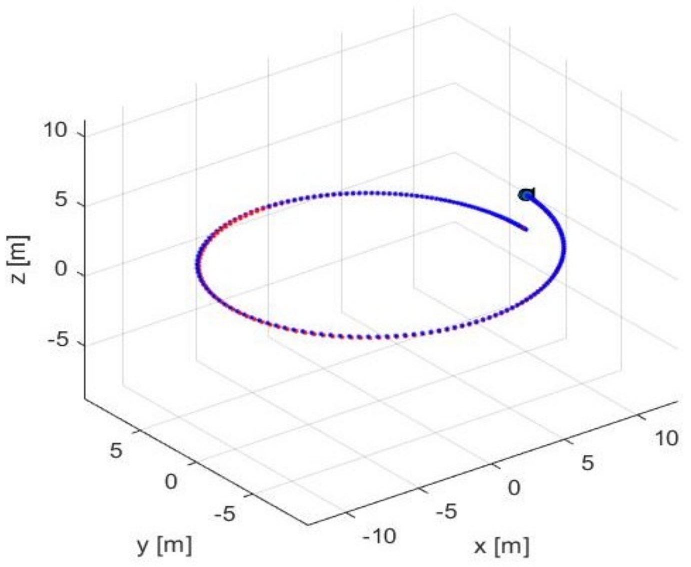
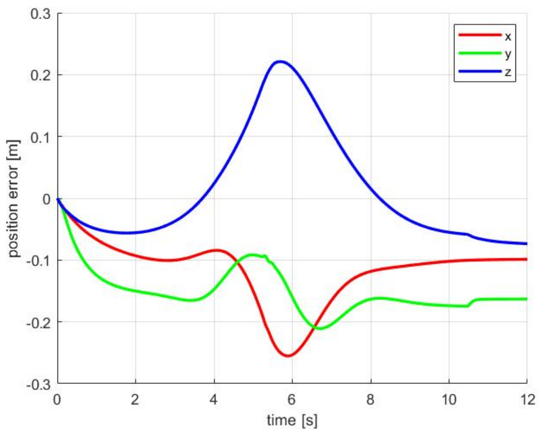
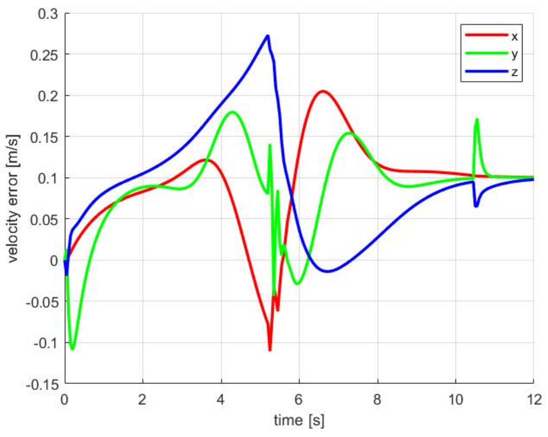


Figure 11: circular trajectory at 6m/s without disturbance

Highlighted in 12b is the velocity tracking error between the desired path velocity states $[\dot{x_d} \quad \dot{y_d} \quad \dot{z_d}]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 0.27\frac{m}{s}$ which occurs at $t = 6s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 0.1\frac{m}{s}$ at $5.8s$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 0.21\frac{m}{s}$ which also occurs at the same time frame as above. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [22 \quad 27 \quad 35]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [5.5 \quad 5.5 \quad 11]$.



(a) position error in x, y, z [m]



(b) velocity error in x, y, z [m/s]

Figure 12: trajectory tracking error at 6m/s

As shown in 13 below the quadcopter is made to track a 3-D circular trajectory at $8\frac{m}{s}$. It is seen that there is a slight mismatch between the desired trajectory and actual trajectory. Outlined in 14a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.045m$ which occurs at $t = 4.1s$. The maximum error for the y component of trajectory is $\approx 0.2m$ at $3.9s$ while he maximum error for the z component of trajectory is $\approx 0.6m$ which also occurs at $t = 4.1s$. The error dynamics converges to steady state at $t = 9s$. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [18 \quad 25 \quad 40]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [5 \quad 5 \quad 9]$.

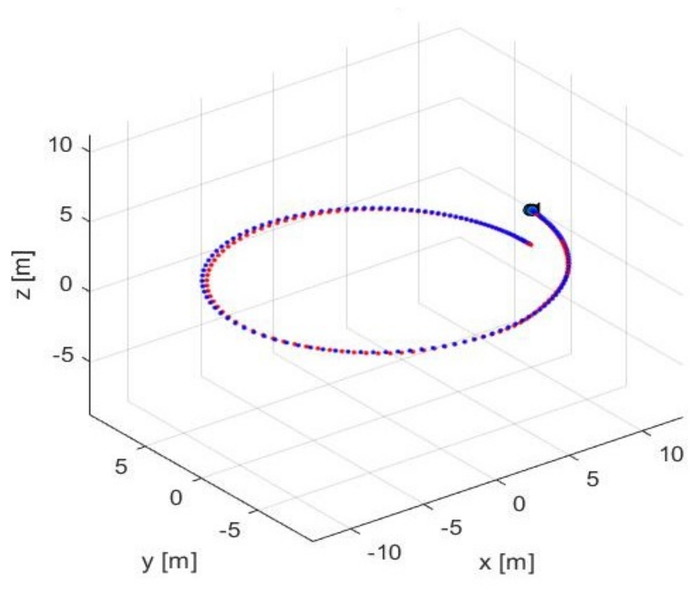
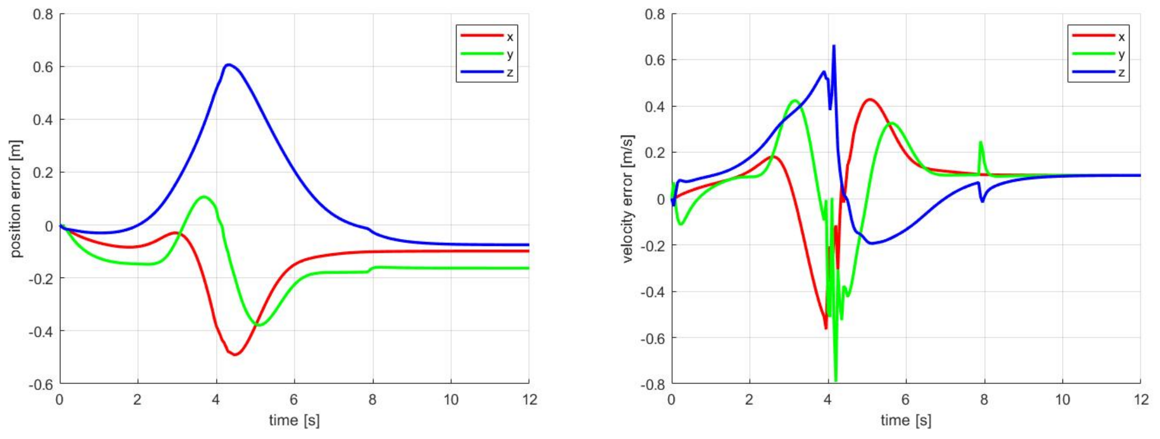


Figure 13: circular trajectory at 8m/s without disturbance

Highlighted in 14b is the velocity tracking error between the desired path velocity states $[\dot{x_d} \quad \dot{y_d} \quad \dot{z_d}]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 0.45\frac{m}{s}$ which occurs at $t = 4s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 0.8\frac{m}{s}$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 0.7\frac{m}{s}$ which also occurs at the same time frame as above. The error dynamics converge at $0.1\frac{m}{s}$.



(a) position error in x, y, z [m]

(b) velocity error in x, y, z [m/s]

Figure 14: trajectory tracking error at 8m/s

As shown in 15 below the quadcopter is made to track a 3-D circular trajectory at $10\frac{m}{s}$. It can be observed the the quadcopter trajectory is significantly off the desired trajectory. Outlined in 16a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.45m$ which occurs at $t = 3.8s$. The maximum error for the y component of trajectory is $\approx 0.25m$ while he maximum error for the z component of trajectory is $\approx 0.6m$ which also occurs at the same time frame as above.

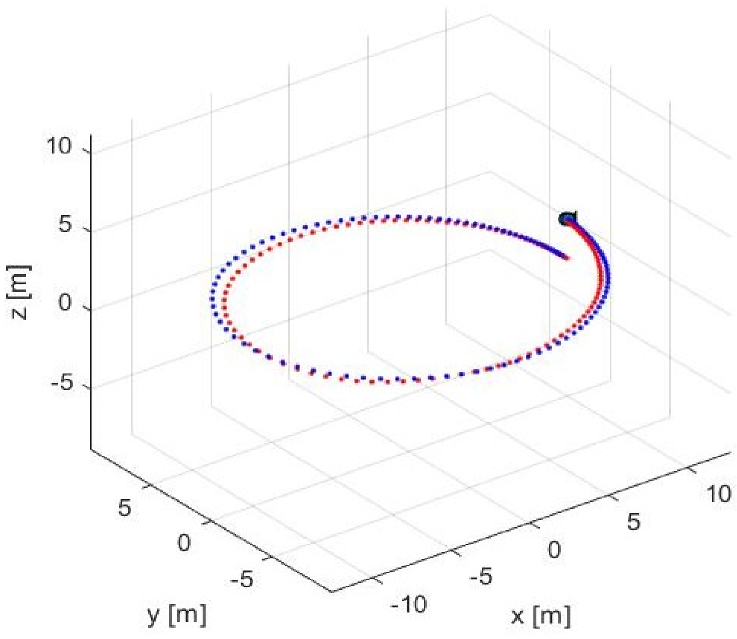
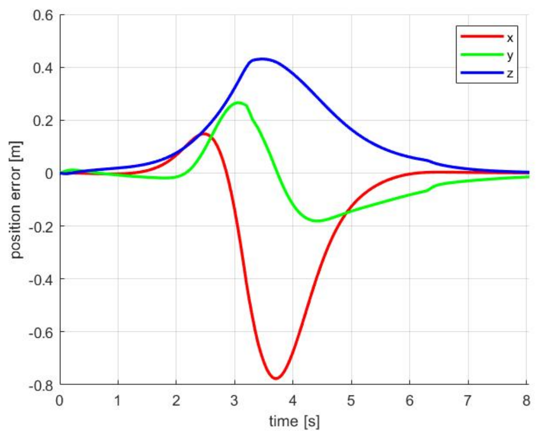
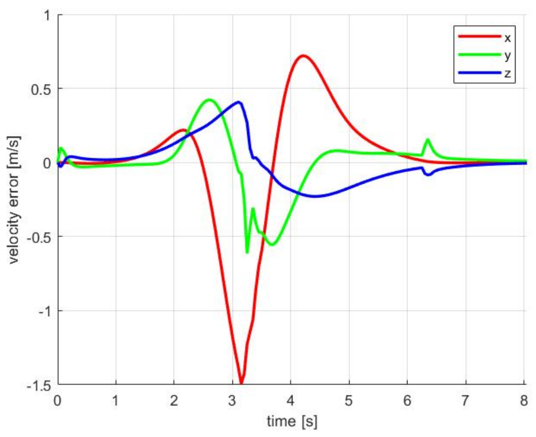


Figure 15: circular trajectory at 10m/s without disturbance

Highlighted in 16b is the velocity tracking error between the desired path velocity states $[\dot{x}_d \quad \dot{y}_d \quad \dot{z}_d]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 0.5\frac{m}{s}$ which occurs at $t = 4s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 0.8\frac{m}{s}$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 0.6\frac{m}{s}$ which also occurs at the same time frame as above. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [10 \quad 10 \quad 30]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [3.5 \quad 3.5 \quad 7.2]$.



(a) position error in x, y, z [m]



(b) velocity error in x, y, z [m/s]

Figure 16: trajectory tracking error at 10m/s

6.2.2 Spiral Trajectory Performance

The PD controller was tested using a spiral manoeuvre which can be defined as $x_d = 10cos(0.1t)$m, $y_d = 10sin(0.1t)$m, $z_d = 0.1t$m and $\psi_d = \frac{\pi}{6}$rads. The initial position position and altitude of quadcopter are given as $[0 \quad 0 \quad 0]$ and $[0 \quad 0 \quad 0]$ respectively. Due to the inability of the PD controller to accurately track the spiral manoeuvre test scenarios, tests were only carried out for modest velocities between $1\frac{m}{s} \rightarrow 3\frac{m}{s}$.

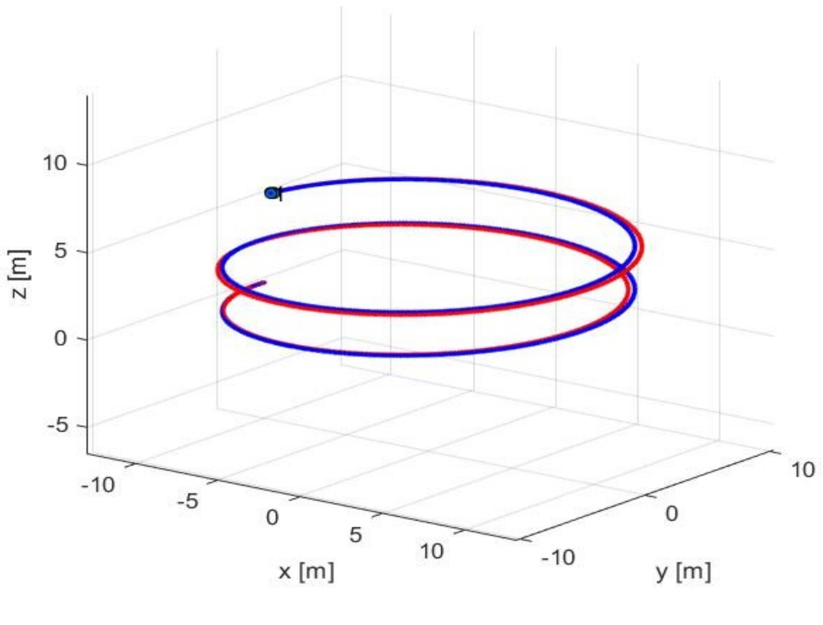
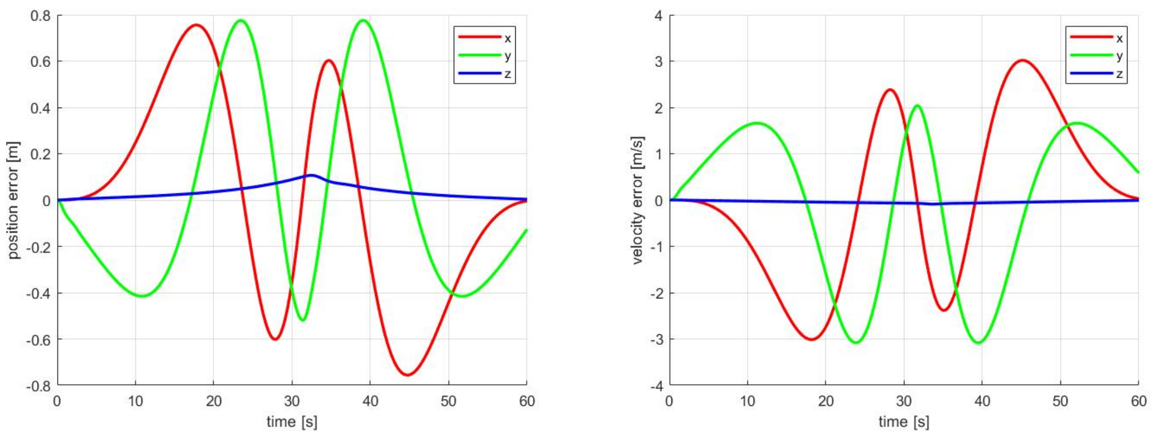


Figure 17: trajectory at 1m/s for spiral trajectory

As shown in 17 below the quadcopter is made to track a 3-D spiral trajectory at $0.5\frac{m}{s}$. A slight mismatch can be observed between the desired and actual trajectory of the quadcopter. Outlined in 18a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.75m$ which occurs at $t = 18s$ and $t = 45s$. The maximum error for the y component of trajectory is $\approx 0.8m$ at $t = 18s$ and $t = 45s$.The maximum error for the z component of trajectory is $\approx 0.1m$ which also occurs at $t = 31s$.



(a) position error in x, y, z [m] (b) velocity error in x, y, z [m/s]

Figure 18: spiral trajectory tracking error at 1m/s

Highlighted in 20b is the velocity tracking error between the desired path velocity states $[\dot{x}_d \quad \dot{y}_d \quad \dot{z}_d]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 3\frac{m}{s}$ which occurs at $t = 45s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 2\frac{m}{s}$ at $t = 30s$ while the maximum error for the $\dot{z}$ component of trajectory is negligible. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [25 \quad 25 \quad 5]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [15 \quad 15 \quad 1]$.

As shown in 19 below the quadcopter is made to track a 3-D spiral trajectory at $1\frac{m}{s}$. An apparent mismatch can be observed between the desired and actual trajectory of the quadcopter. Outlined in 20a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 1.3m$ which occurs at $t = 22s$. The maximum error for the y component of trajectory is $\approx 1.2m$ at $t = 13s$ and $t = 23s$.The maximum error for the z component of trajectory is $\approx 0.5m$ which occurs at $t = 18s$. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [15 \quad 15 \quad 5]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [3 \quad 3 \quad 1]$.

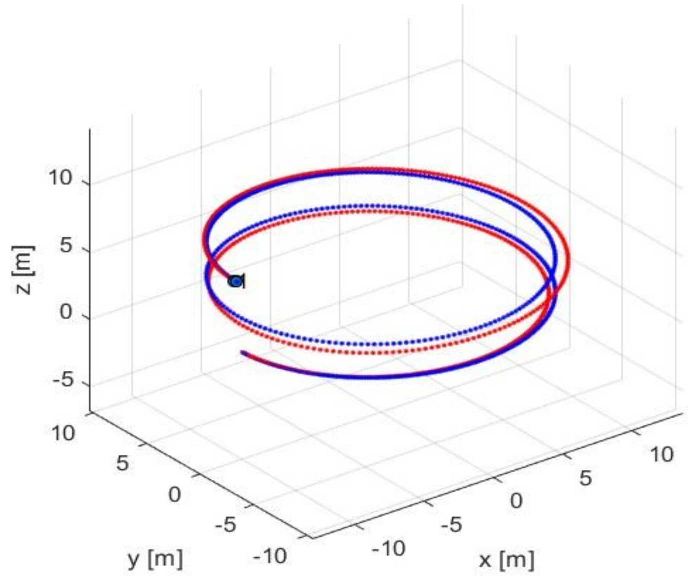
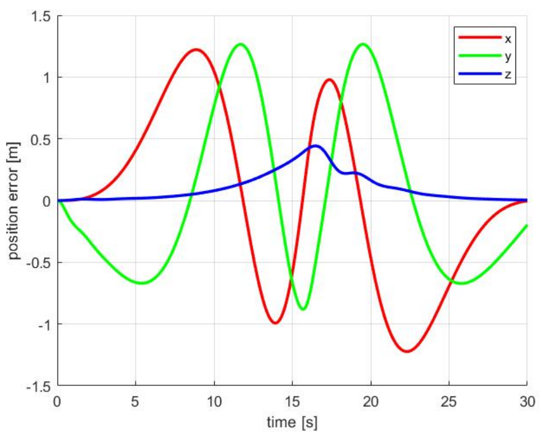
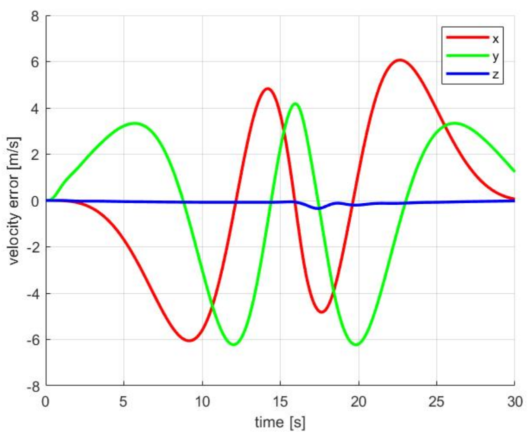


Figure 19: trajectory at 0.5m/s for spiral trajectory

Illustrated in 20b is the velocity tracking error between the desired path velocity states $[\dot{x_d} \quad \dot{y_d} \quad \dot{z_d}]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 6\frac{m}{s}$ which occurs at $t = 23s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 4\frac{m}{s}$ at $t = 18s$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 0.2\frac{m}{s}$ which occurs at $t = 23s$.



(a) position error in x, y, z [m]



(b) velocity error in x, y, z [m/s]

Figure 20: spiral trajectory tracking error at 2m/s

The figure in 21 below illustrates the quadcopter is made to track a 3-D spiral trajectory at $1.5\frac{m}{s}$. It is apparent to see that the quadcopter fails to track the desired trajectory. Outlined in 22a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 2.5m$ which occurs at $t = 7s$. The maximum error for the y component of trajectory is $\approx 2.5m$ at $t = 8s$ and $t = 13s$.The maximum error for the z component of trajectory is $\approx 0.5m$ which occurs at $t = 18s$. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [12 \quad 11 \quad 5.5]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [4.5 \quad 4.5 \quad 0.7]$.

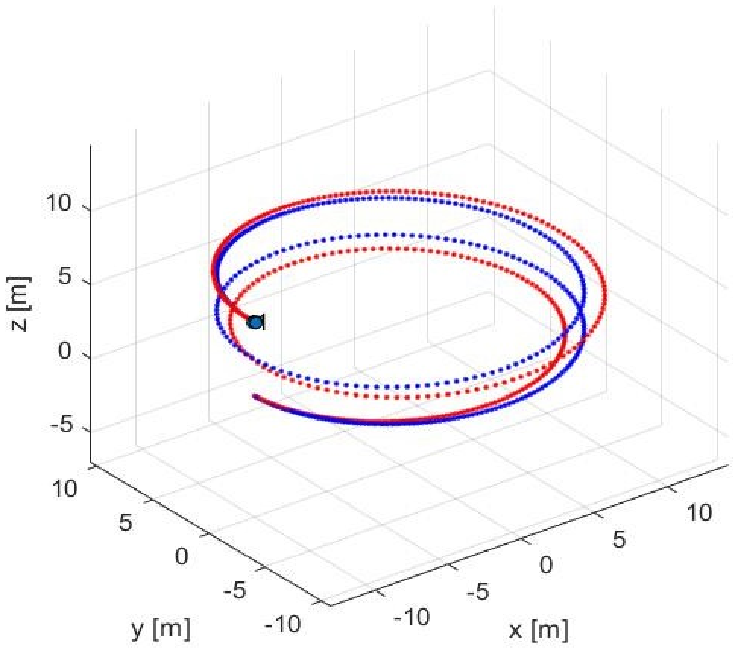
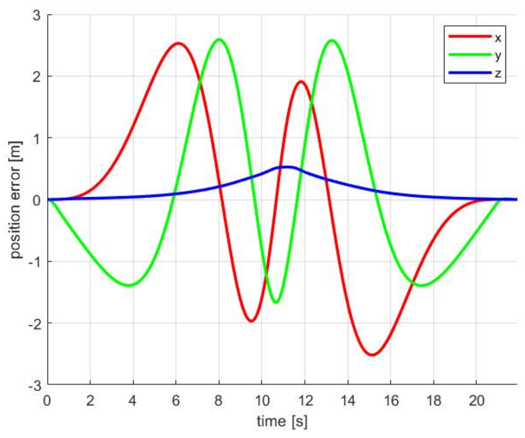
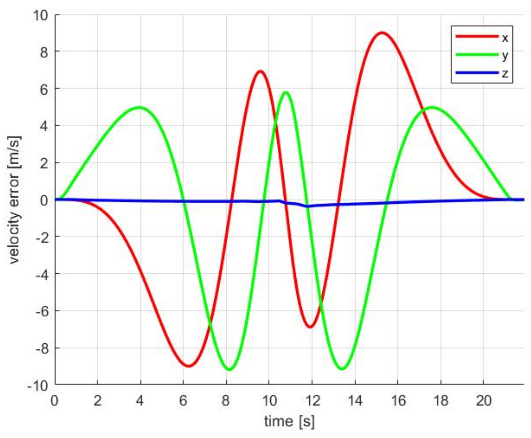


Figure 21: trajectory at 1.5m/s for spiral trajectory

Illustrated in 22b is the velocity tracking error between the desired path velocity state $[\dot{x}_d \quad \dot{y}_d \quad \dot{z}_d]$ and the actual state of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 9\frac{m}{s}$ which occurs at $t = 16s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 9\frac{m}{s}$ at $t = 18s$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 0.1\frac{m}{s}$ which occurs at $t = 13s$.



(a) position error in x, y, z [m]



(b) velocity error in x, y, z [m/s]

Figure 22: spiral trajectory tracking error at 3m/s

6.2.3 External Disturbance Effects on Performance

The robustness of the PD controller to external disturbances acting on the system dynamics in 13 is outlined in this section. The trajectory being used is a circular trajectory defined as $x_d = 10cos(0.5t)m$, $y_d = 10sin(0.5t)m$, $z_d = 1 + 0.1tm$ and $\psi_d = \frac{\pi}{6}rads$. The simulation variables are defined such that $[V_w \quad V_p \quad \rho \quad A \quad L \quad m \quad k_F \quad w_i]$ are attributed to $[1\frac{m}{s} \quad 0.5\frac{m}{s} \quad 1.293\frac{g}{L}\frac{N}{m^2} \quad 0.0016m^2 \quad 0.046m \quad 0.003kg \quad 6.11 \text{ x } 10^{-}8\frac{N}{rpm^2} \quad 2023rpm]$. Due to the lack of robustness of the PD controller to wind gusts, the test were on only carried out for velocities of $2\frac{m}{s} \rightarrow 6\frac{m}{s}$.

The figure in 23 below illustrates the quadcopter tracking a 3-D circular trajectory at $2\frac{m}{s}$ where it can be observed that quadcopter tracks the desired trajectory with minimal error. Outlined in 24a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.09m$ which occurs at $t = 7s$. The maximum error for the y component of trajectory is $\approx 0.11m$ at $t = 8s$.The maximum error for the z component of trajectory is $\approx 0.03m$ which occurs at $t = 13s$. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [4 \quad 4 \quad 18]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [4 \quad 4.5 \quad 7]$.

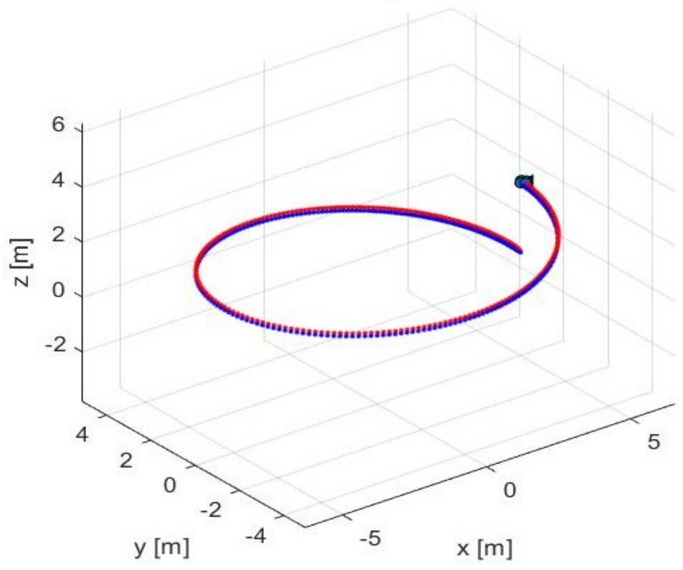
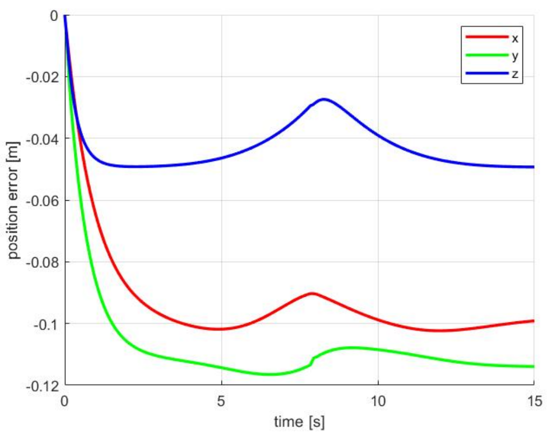
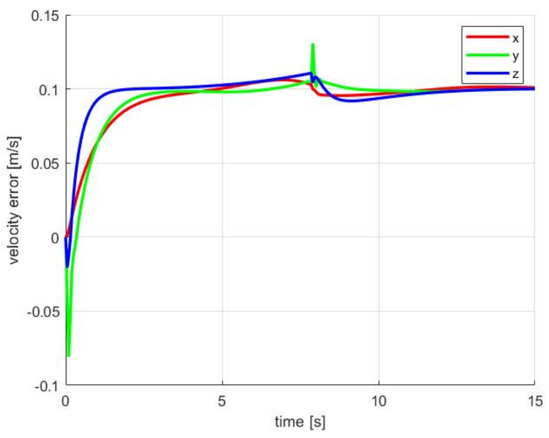


Figure 23: circular trajectory at 2m/s under disturbance

Illustrated in 22b is the velocity tracking error between the desired path velocity states $[\dot{x_d} \quad \dot{y_d} \quad \dot{z_d}]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 0.09\frac{m}{s}$ which occurs at $t = 9s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 0.12\frac{m}{s}$ at $t = 11s$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 0.11\frac{m}{s}$ which occurs at $t = 9s$.



(a) position error in x, y, z [m]



(b) velocity error in x, y, z [m/s]

Figure 24: trajectory tracking error at $2m/s$

The figure in 25 below illustrates the quadcopter below illustrates the quadcopter tracking a 3-D circular trajectory at $4\frac{m}{s}$ where it can be seen that quadcopter tracks the trajectory with minimal error. Outlined in 26a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.2m$ which occurs at $t = 5s$. The maximum error for the y component of trajectory is $\approx 0.1m$ at $t = 5s$.The maximum error for the z component of trajectory is $\approx 0.2m$ which occurs at $t = 4s$. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [11 \quad 11 \quad 21]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [7 \quad 7 \quad 10]$.

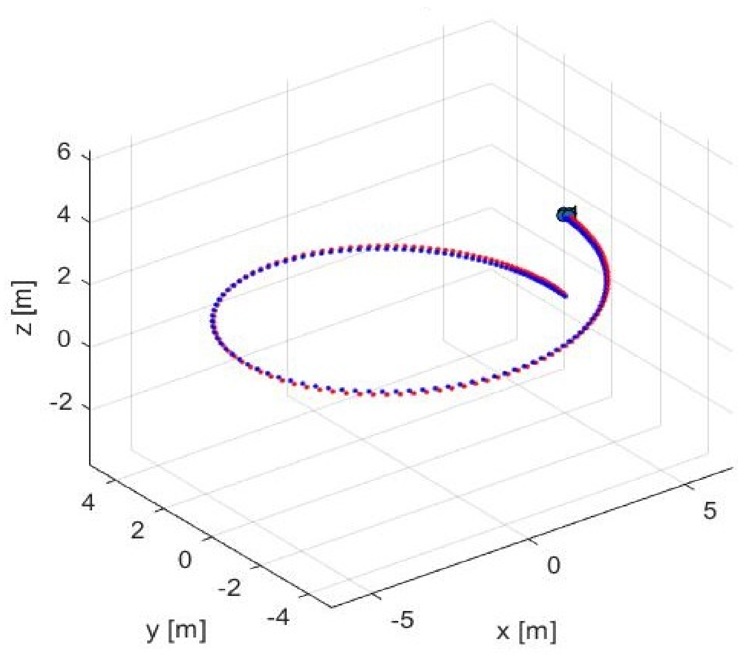
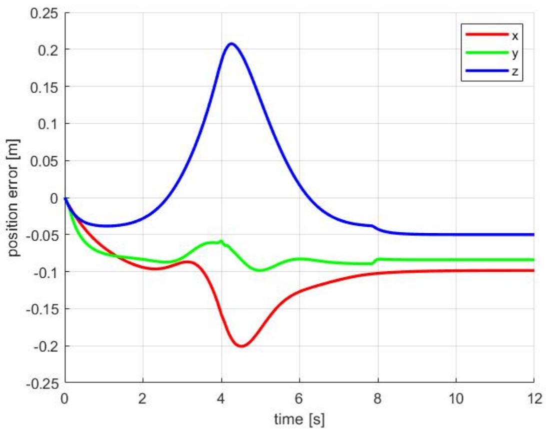


Figure 25: circular trajectory at 4m/s under disturbance

Illustrated in 22b is the velocity tracking error between the desired path velocity states $[\dot{x_d} \quad \dot{y_d} \quad \dot{z_d}]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 0.09\frac{m}{s}$ which occurs at $t = 4s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 0.15\frac{m}{s}$ at $t = 4s$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 0.3\frac{m}{s}$ which occurs at $t = 4s$.


(a) position error in x, y, z [m]

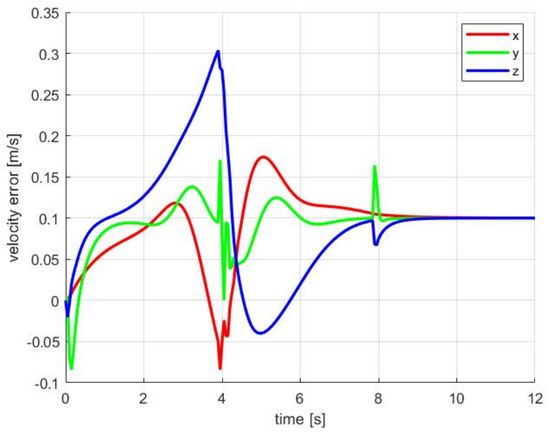

(b) velocity error in x, y, z [m/s]

Figure 26: trajectory tracking error at $4m/s$

The figure in 25 below illustrates the quadcopter below illustrates the quadcopter tracking a 3-D circular trajectory at $6\frac{m}{s}$ where it can be seen that a mismatch occurs between the desired trajectory and the quadcopter trajectory. Outlined in 28a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.8m$ which occurs at $t = 3.5s$. The maximum error for the y component of trajectory is $\approx 0.2m$ at $t = 2.5s$.The maximum error for the z component of trajectory is $\approx 0.7m$ which occurs at $t = 3s$. The controller gains where chosen such that: $[k_{p_x} \quad k_{p_y} \quad k_{p_z}] = [19 \quad 19 \quad 28]$ and $[k_{d_x} \quad k_{d_y} \quad k_{d_z}] = [5.2 \quad 5.7 \quad 7]$.

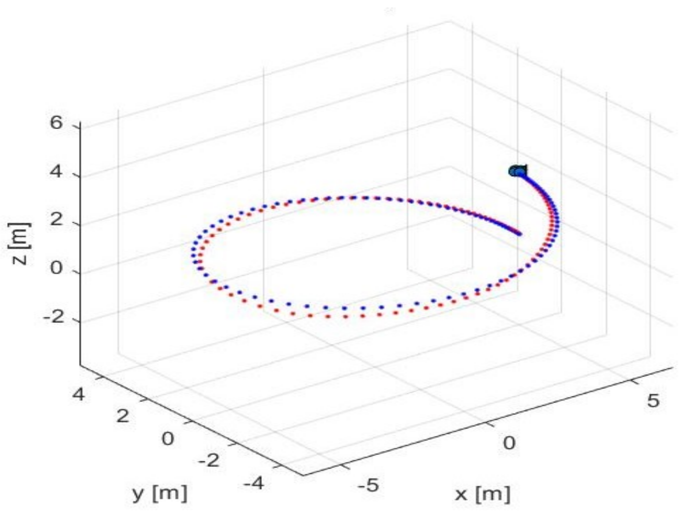
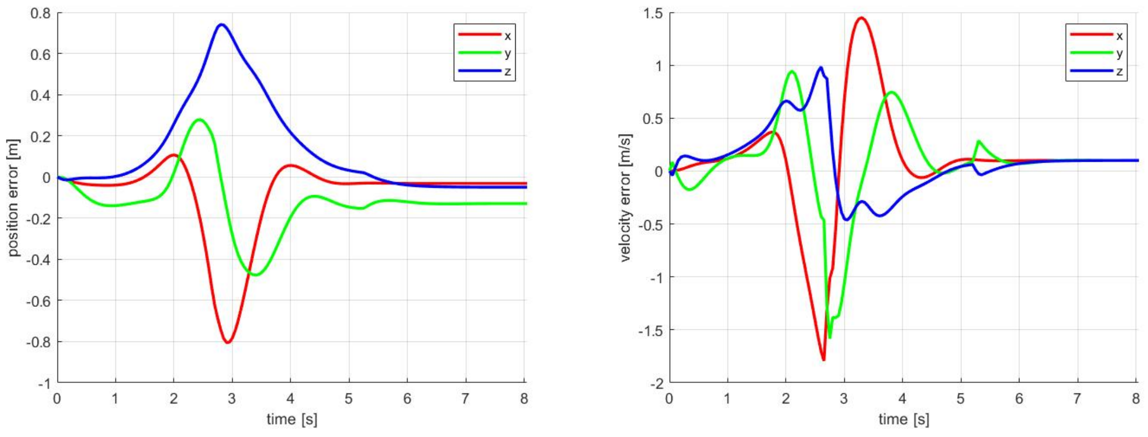


Figure 27: circular trajectory at 6m/s under disturbance

Illustrated in 28b is the velocity tracking error between the desired path velocity states $[\dot{x_d} \quad \dot{y_d} \quad \dot{z_d}]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 1.7\frac{m}{s}$ which occurs at $t = 3.5s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 1\frac{m}{s}$ at $t = 3s$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 1\frac{m}{s}$ which occurs at $t = 3s$.



(a) position error in x, y, z [m] (b) velocity error in x, y, z [m/s]

Figure 28: trajectory tracking error at $6m/s$

6.3 Active Disturbance Rejection Controller

Performance of the classical ADRC structure in 4 for quadcopter trajectory tracking and system dynamics performance is evaluated. The robustness of the controller against parametric uncertainties is investigated. The nonlinear controller variables are selected such that: $[b_0 \quad \delta \quad h_g \quad r \quad h] = [0.83 \quad 0.001 \quad 0.15 \quad 100 \quad 0.01]$. The observer gains of the ESO were chosen such that $[\beta_{01} \quad \beta_{02} \quad \beta_{03}] = [100 \quad 100(\frac{1}{2(h_g^{0.5})}) \quad 100(\frac{2}{25(h_g^{1.2})})]$. The trajectory being used for the purposes of simulation is a spiral trajectory defined as $x_d = 10cos(0.5t)m$, $y_d = 10sin(0.5t)m$, $z_d = 1 + 0.1tm$ and $\psi_d = \frac{\pi}{6}rads$.

The disturbance variables are defined such that $[V_w \quad V_p \quad \rho \quad A \quad L \quad m \quad k_F \quad w_i]$ are paramaterized by $[1\frac{m}{s} \quad 0.5\frac{m}{s} \quad 1.293\frac{g}{L}\frac{N}{m^2} \quad 0.0016m^2 \quad 0.046m \quad 0.003kg \quad 6.11 \text{ x } 10^-8\frac{N}{rpm^2} \quad 2023rpm]$. The response of the altitude loop dynamics ϕ, θ, and ψ as well as virtual control inputs U_1, U_2, U_3 and U_4 responses are presented. MATLAB code and Simulink blocks used for the design of the classical ADRC controller can be found in B and 44. Initial states of the quadcopter are given such that $[x \quad y \quad z]$ and $[\dot{x} \quad \dot{y} \quad \dot{z}] = [0 \quad 0 \quad 0]$ and $[0 \quad 0 \quad 0]$

respectively.

6.3.1 Spiral Trajectory Performance under Disturbances

The figure in 29 below illustrates the quadcopter tracking a 3-D spiral trajectory where it is apparent that the quadcopter tracks the desired trajectory. It is noted that for the figure in 29, the quadcopter dynamics contains no system uncertainties. Outlined in 30a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.61m$ which occurs at $t = 65s$. The maximum error for the y component of trajectory is $\approx 0.5m$ at $t = 35s$ and $t = 65s$.The maximum error for the z component of trajectory is negligible.

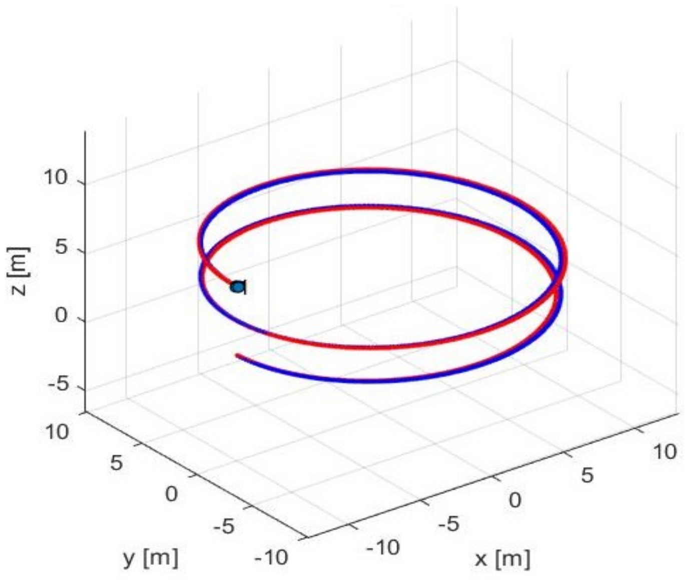
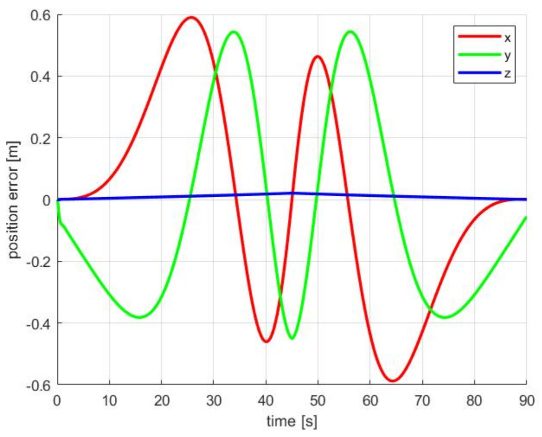
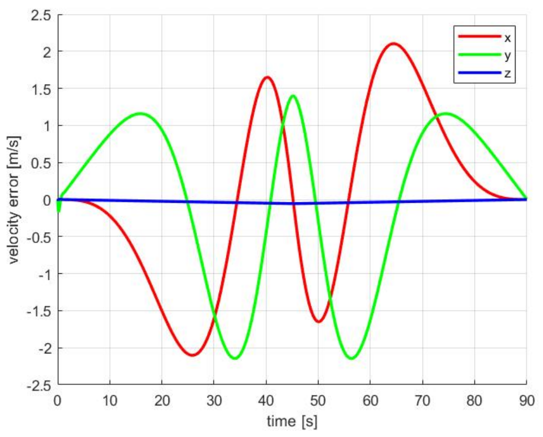


Figure 29: trajectory under disturbance at 3m/s

Illustrated in 30b is the velocity tracking error between the desired path velocity states $[\dot{x_d} \quad \dot{y_d} \quad \dot{z_d}]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 2\frac{m}{s}$ which occurs at $t = 65s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 2\frac{m}{s}$ at $t = 45s$ while the maximum error for the $\dot{z}$ component of trajectory is negligible.



(a) position error x, y, z [m]



(b) velocity error x, y, z [m/s]

The figure in 31 below illustrates the quadcopter which is made to track a 3-D spiral trajectory under the presence of system disturbances. It can be seen that a mismatch is apparent between the desired trajectory and the quadcopter trajectory. Outlined in 32a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.8m$ which occurs at $t = 25s$. The maximum error for the y component of trajectory is $\approx 0.8m$ at $t = 55s$.The maximum error for the z component of trajectory is $\approx 0.3m$ which occurs at $t = 25s$.

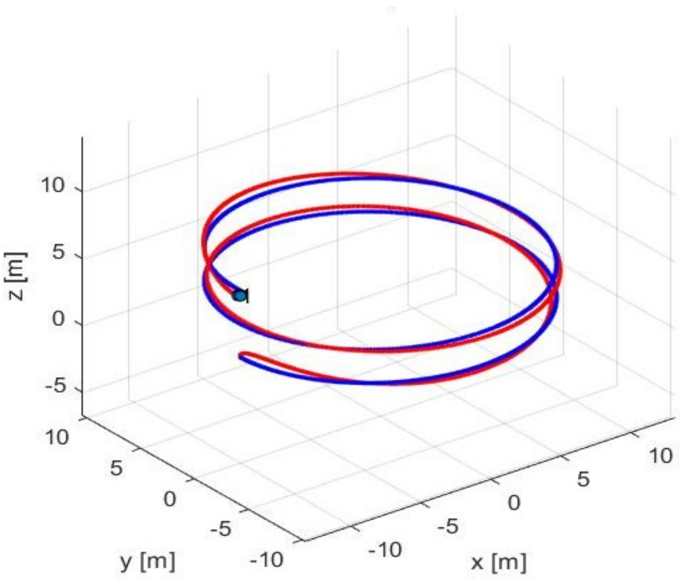
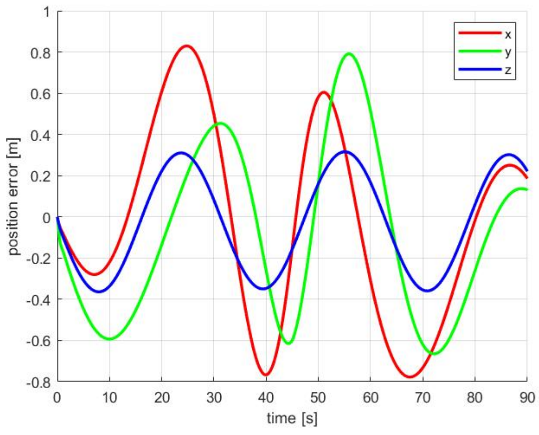


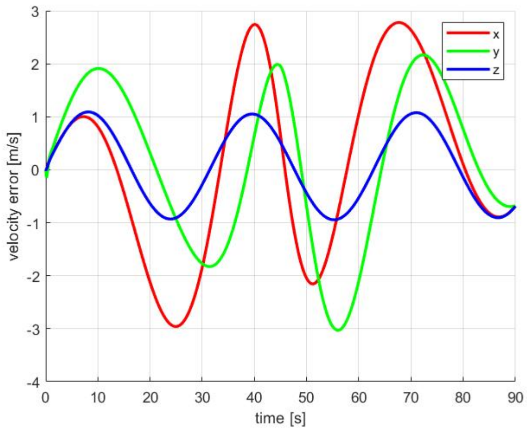
Figure 31: trajectory response at 3m/s

Illustrated in 32b is the velocity tracking error between the desired path velocity states

$[\dot{x}_d \quad \dot{y}_d \quad \dot{z}_d]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 2.5\frac{m}{s}$ which occurs at $t = 40s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 2\frac{m}{s}$ at $t = 50s$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 1\frac{m}{s}$ which occurs at $t = 40s$.



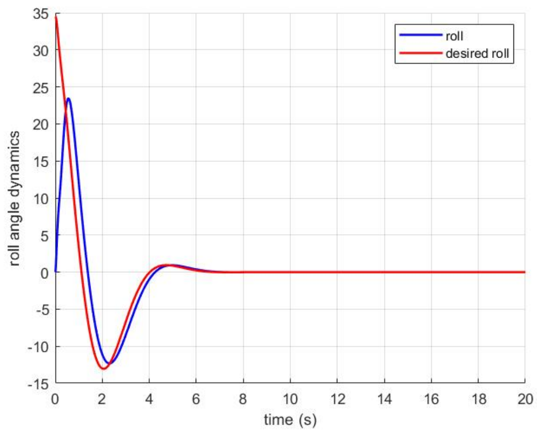
(a) position error x, y, z [m]

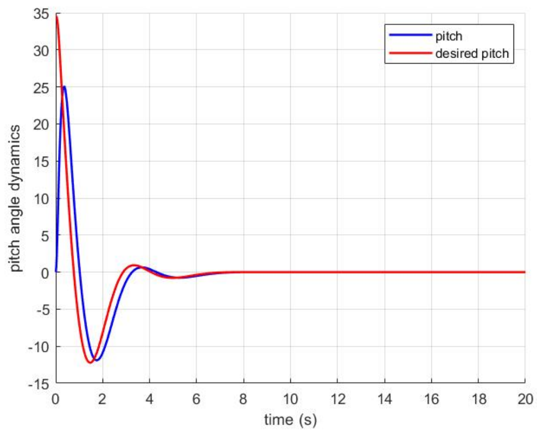


(b) velocity error x, y, z [m/s]

6.3.2 Individual Altitude Dynamics Response and Virtual Control Inputs

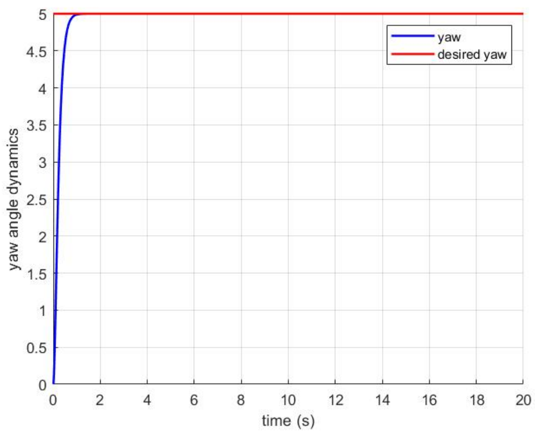
The desired positions of the quadcopter altitude dynamics are given such that $[\phi_d \quad \theta_d \quad \psi_d]$ $= [0^\circ \quad 0^\circ \quad 5^\circ]$. The initial positions are outlined as $[\phi \quad \theta \quad \psi] = [0^\circ \quad 0^\circ \quad 0^\circ]$. The figure in 33a depicts the roll dynamics response of the quadcopter. It can be observed that the desired roll angle is tracked at $t = 6s$ and the signal reaches steady state $t = 7s$. The figure in 33b depicts the roll dynamics response of the quadcopter. It can be observed that the desired roll angle is tracked at $t = 6s$ and the signal reaches steady state $t = 8s$.



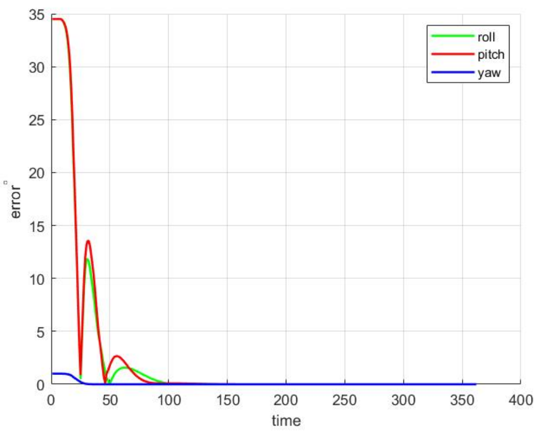


(a) total disturbance response of roll dynamics (b) total disturbance response of pitch dynamics

The figure in 34a depicts the yaw dynamics response of the quadcopter. It can be observed that the desired yaw angle is tracked and it reaches steady state at $t = 1s$. The figure is 34b is used to depict the error between the desired altitude dynamics $[\phi_d \quad \theta_d \quad \psi_d]$ and current altitude states $[\phi \quad \theta \quad \psi]$. A maximum error can be observed in the pitch dynamics of $35°$ however, this is considered trivial due to the different starting positions of the actual and desired signal. Maximum error in the roll and yaw channels are $11°$ and $1°$ respectively. The error dynamics of all the channels reach steady state at $t = 100s$.



(a) total disturbance response of yaw dynamics



(b) error response in altitude subsystem

Outlined in 35 are the dynamic responses of the virtual control inputs U_1, U_2, U_3 and U_4. The control inputs U_2, U_3 and U_4 reach steady state and settle at 0 while U_1 settles at -20.

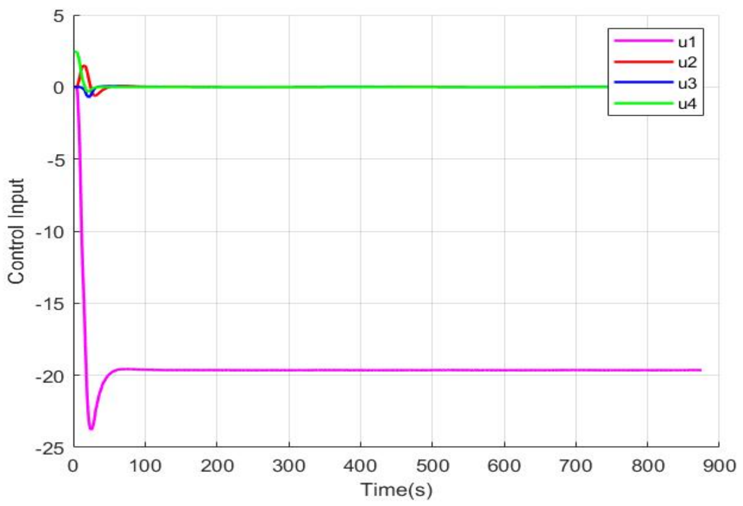


Figure 35: virtual control input response

6.4 Linear Active Disturbance Rejection Controller

The performance of the LADRC structure in 5 for quadcopter trajectory tracking and altitude system dynamics response is evaluated. The robustness of the controller against parametric wind gusts is investigated. The observer gain parameters for each channel are chosen as the same for the sake of simplicity nevertheless, gain optimization experiments carried out showed consistency of desired results, as such the need for observer manipulation was not deemed necessary by the author. These gains are outlined such that $[L_1 \quad L_2 \quad L_3] = [29.578 \quad 290.7 \quad 3000]$.

The trajectory being used for the purposes of simulation is a spiral trajectory defined as $x_d = 10cos(0.1t)$m, $y_d = 10sin(0.1t)m$, $z_d = 0.1tm$ and $\psi_d = \frac{\pi}{6}rads$ for UAV quadcopter speed of $4m/s$. The disturbance variables are defined such that $[V_w \quad V_p \quad \rho \quad A \quad L \quad m \quad k_F \quad w_i]$ $= [1\frac{m}{s} \quad 0.5\frac{m}{s} \quad 1.293\frac{g}{L}\frac{N}{m^2} \quad 0.0016m^2 \quad 0.046m \quad 0.003kg \quad 6.11 \text{ x } 10^{-}8\frac{N}{rpm^2} \quad 2023rpm]$. The response of the altitude loop dynamics ϕ, θ, and ψ as well as virtual control inputs U_1, U_2, U_3 and U_4 responses is evaluated. MATLAB code and Simulink blocks used for the design of the LADRC structure is enclosed in C and 45. The initial states of the quadcopter are given such that $[x \quad y \quad z]$ and $[\dot{x} \quad \dot{y} \quad \dot{z}] = [0 \quad 0 \quad 0]$ and $[0 \quad 0 \quad 0]$ respectively.

6.4.1 Spiral Trajectory Performance with Disturbances

The figure in 36 below illustrates the quadcopter tracking a 3-D spiral trajectory where it can be seen that the quadcopter tracks the desired trajectory. It is noted that for the figure in 29 the quadcopter dynamics contains no system uncertainties. Outlined in 30a is the position tracking error between the desired path position states $[x_d \quad y_d \quad z_d]$ and the actual states of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.6m$ which occurs at $t = 65s$. The maximum error for the y component of trajectory is $\approx 0.5m$ at $t = 35s$ and $t = 65s$.The maximum error for the z component of trajectory is negligible.

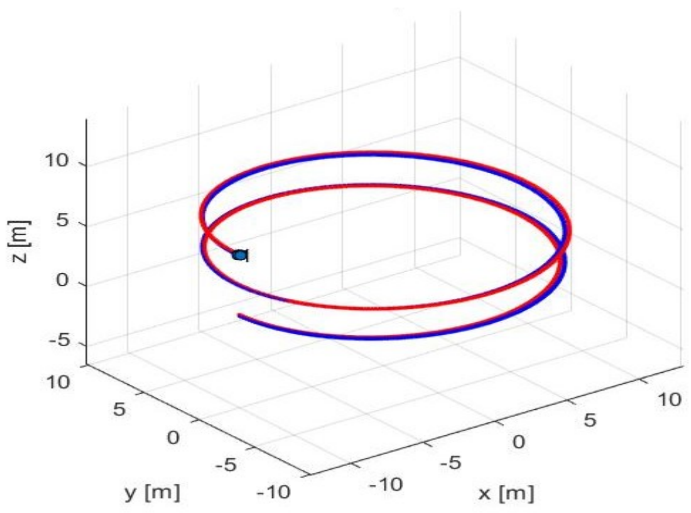
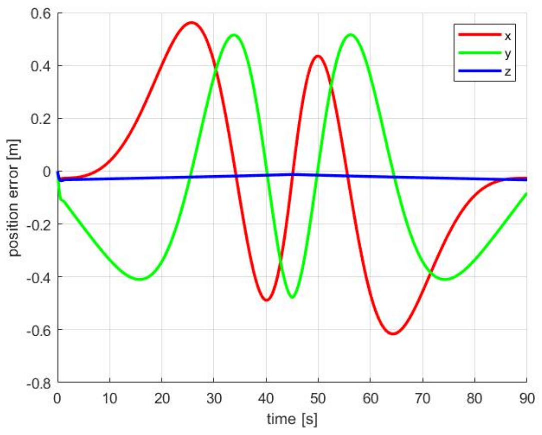
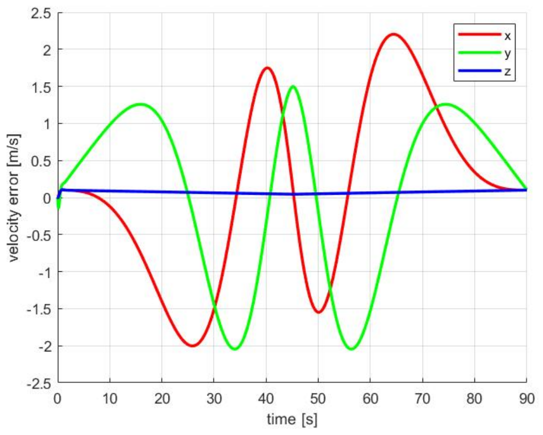


Figure 36: trajectory response of ladrc strategy

Illustrated in 37a is the velocity tracking error between the desired path velocity states $[\dot{x}_d \quad \dot{y}_d \quad \dot{z}_d]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 2.2\frac{m}{s}$ which occurs at $t = 65s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 2\frac{m}{s}$ at $t = 45s$ while the maximum error for the $\dot{z}$ component of trajectory is negligible.



(a) position error x, y, z [m]



(b) velocity error x, y, z [m/s]

The 38 below illustrates the quadcopter which is made to track a 3-D spiral trajectory under the presence of system disturbances. It can be seen that a mismatch is apparent between the desired trajectory and the quadcopter trajectory. Outlined in 39a is the position tracking error between the desired path position state $[x_d \quad y_d \quad z_d]$ and the actual state of the quadcopter $[x \quad y \quad z]$. The maximum error for the x component of trajectory is $\approx 0.8m$ which occurs at $t = 25s$. The maximum error for the y component of trajectory is $\approx 0.7m$ at $t = 55s$.The maximum error for the z component of trajectory is $\approx 0.3m$ which occurs at various intervals.

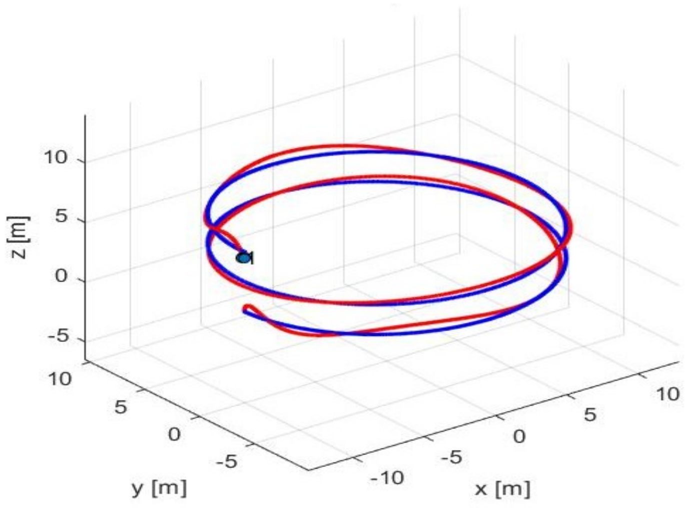
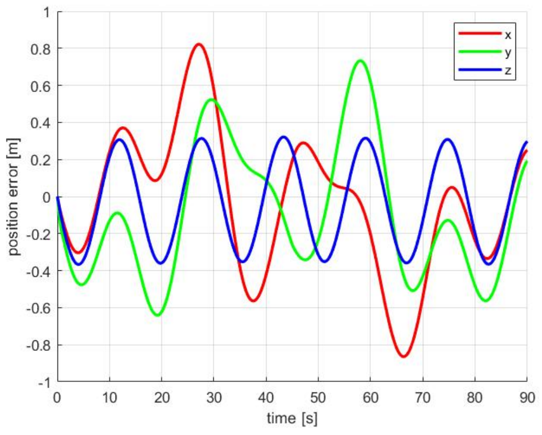
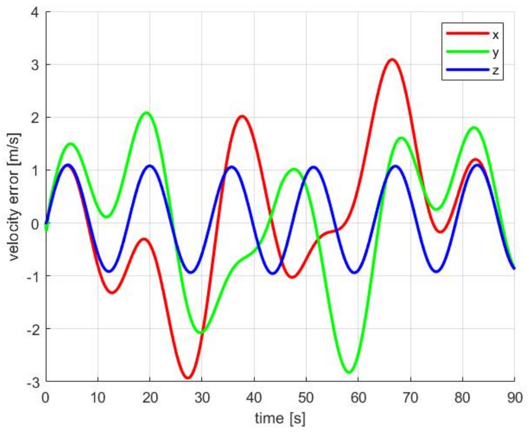


Figure 38: trajectory response of ladrc with disturbance

Illustrated in 39b is the velocity tracking error between the desired path velocity states $[\dot{x}_d \quad \dot{y}_d \quad \dot{z}_d]$ and the actual states of the quadcopter $[\dot{x} \quad \dot{y} \quad \dot{z}]$. The maximum error for the $\dot{x}$ component of trajectory is $\approx 3\frac{m}{s}$ which occurs at $t = 65s$. The maximum error for the $\dot{y}$ component of trajectory is $\approx 2.8\frac{m}{s}$ at $t = 60s$ while the maximum error for the $\dot{z}$ component of trajectory is $\approx 1\frac{m}{s}$ which occurs at various intervals.



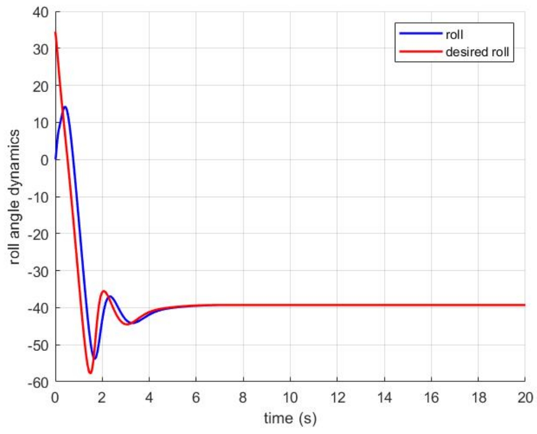
(a) position error x, y, z [m]

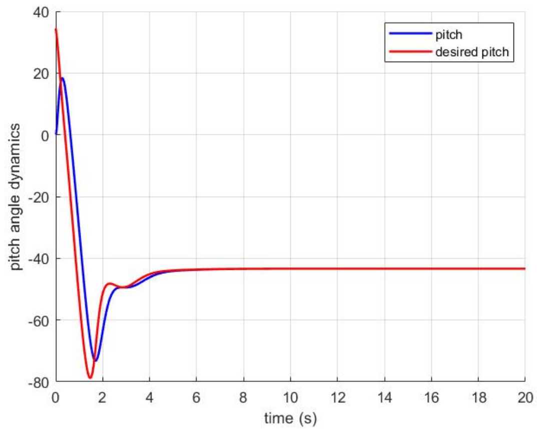


(b) velocity error x, y, z [m/s]

6.4.2 Individual Altitude Dynamics Response and Virtual Control Inputs

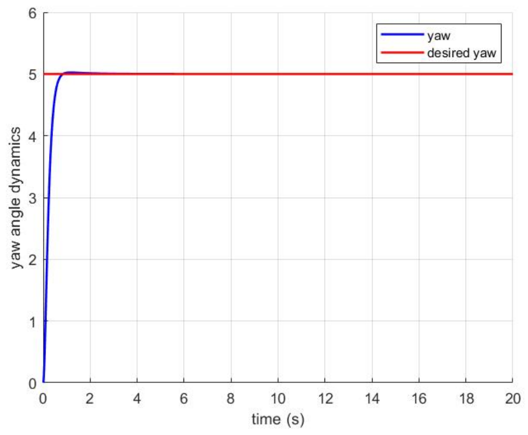
The desired positions of the quadcopter altitude dynamics are given such that $[\phi_d \quad \theta_d \quad \psi_d]$ $= [0° \quad 0° \quad 5°]$. The initial positions are outlined as $[\phi \quad \theta \quad \psi] = [0° \quad 0° \quad 0°]$. The figure in 40a depicts the roll dynamics response of the quadcopter. It can be observed that the desired roll angle is tracked at $t = 6s$ and the signal reaches steady state $t = 7s$. The figure in 40b depicts the roll dynamics response of the quadcopter. It can be observed that the desired roll angle is tracked at $t = 6s$ and the signal reaches steady state $t = 8s$.



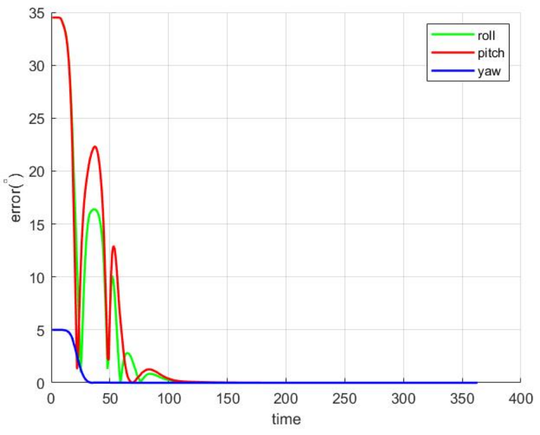


(a) total disturbance response of roll dynamics (b) total disturbance response of pitch dynamics

The figure in 41a depicts the yaw dynamics response of the quadcopter. It can be observed that the desired yaw angle is tracked and it reaches steady state at $t = 1s$. The figure is 41b is used to depict the error between the desired altitude dynamics $[\phi_d \quad \theta_d \quad \psi_d]$ and current altitude states $[\phi \quad \theta \quad \psi]$. A maximum error can be observed in the pitch dynamics of 35° however, this is considered trivial due to the different starting positions of the actual and desired signal. Maximum error in the roll channel is 16° and yaw channels is 5° respectively. The error dynamics of all the channels reach steady state at $t = 100s$.



(a) total disturbance response of yaw dynamics



(b) error response in altitude subsystem

Outlined in 42 are the dynamic responses of the virtual control inputs U_1, U_2, U_3 and U_4. The control inputs U_2, U_3 and U_4 reach steady state and settle at 0 while U_1 settles at -18 while exhibiting oscillatory behaviour.

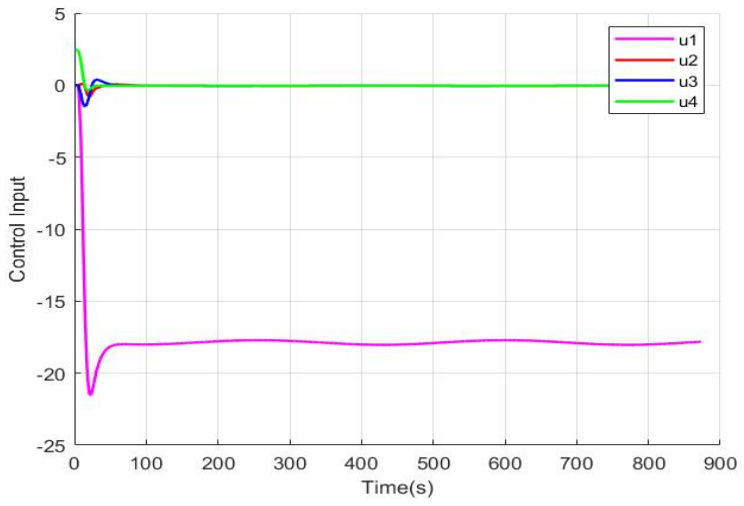


Figure 42: virtual control input response

6.5 Controller Performance Analysis

Upon the presentation of the results in the earlier parts of this document, critical discussion point arise. The PD control strategy was shown to have desirable UAV trajectory tracking performance albeit with some drawbacks and shortcomings. Firstly, it was shown that PD control structure was able to facilitate trajectory tracking for modest speeds of up to $4\frac{m}{s}$ for non-agile trajectories such as a the circular trajectory shown in this document. Reaffirming the aforementioned point, it was proven that PD controller fails to steer the quadcopter towards a desired trajectory for agile manoeuvres. When disturbances are incorporated into the system dynamics, the PD controller did not exhibit the capability of accurately tracking a desired reference trajectory.

The tests carried out for validation purposes of the classical ADRC and LADRC structures respectively were concise for the sake of brevity, as such a spiral trajectory with incorporated uncertainty dynamics was leveraged. Without loss of generality the classical ADRC structure exhibited better performance results compared to the LADRC structure although the difference in the quality of results is not worth noting due to minimal difference of the error dynamics. Works carried out in Zhang et al. [2019] and Xu et al. [2020] also carried out similar experimental validation using Back-stepping SMC based ADRC in Xu et al. [2020] and SMC based ADRC in Zhang et al. [2019]. These results yielded on average $\approx 10\%$ better performance to that proposed in this thesis. It is however noted that the uncertainty dynamics being used in this thesis would have a greater adverse affect on error dynamics than that used in the aforementioned works.

A case in point is Zhang et al. [2019] where time-varying disturbances were only added to the altitude loop and in Xu et al. [2020] stochastic white noise was added to solely on the altitude channel. The impact of this cannot be explicitly stated at this point and left as an exercise to the reader, however, it is trivial to assume that error dynamics being used in this work which affects both position and altitude loop would have deteriorated the quality of results comparatively that is. The error dynamics leveraged were similar to Ding and Wang [2018] for LADRC however, the results in the work was only provided for altitude dynamics, nevertheless, the results in this work were comparable and even better results were observed for the classical ADRC. Finally, it can be said that both the proposed classical ADRC and LADRC structures exhibit better disturbance rejection capabilities to the PD control structure presented within the scope of this document.

7 Conclusion and Further Works

The economic and environmental incentive of UAV quadcopters was presented by way of a literature survey. Methods of UAV trajectory tracking control were also outlined from literature. The theoretic framework of the ADRC framework is subsequently presented. Furthermore, 6DOF quadcopter system dynamics was derived and outlined in the consequent section as well as the a model for system uncertainty and disturbances dynamics. The next section investigated controller design for the purposes of quadcopter trajectory tracking which entailed PD control, classical ADRC and LADRC structures respectively. Subsequently, simulation results were provided highlighting the ability and shortcomings of all three controllers to facilitate UAV trajectory tracking objectives as well as system dynamics responses.

The results ascertain the limitations of the PD control to provide UAV trajectory tracking objectives when uncertainties are introduced to the system dynamics as well the inability of the quadcopter to track a pre-defined trajectory for agile manoeuvres and high velocities. The classical ADRC framework exhibited the ability of enabling the UAV to accurately track desired reference trajectory was highlighted although, less desirable results were shown when model uncertainties were included. The altitude dynamics responses were shown to converge to steady state with zero error. The LADRC structure yielded similar results to the classical ADRC although the results were not as accurate, though this is considered trivial due to the minimal nature of the error. Further works will investigate the use of a Back-stepping SMC based ADRC controller with the model for system uncertainties explored in this work. It is anticipated that this would enable the error dynamics to converge to arbitrarily small compact sets. It is noted that this was not done only because of the presence time constraints on this project.

References

K. Abdulmajeed. Autonomous control of a quadrotor-manipulator; application of extended state disturbance observer. *arXiv preprint arXiv:1910.09052*, 2019.

A. P. Aguiar and J. P. Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE transactions on automatic control*, 52(8):1362–1379, 2007.

A. P. Aguiar, J. P. Hespanha, and P. V. Kokotovic. Path-following for nonminimum phase systems removes performance limitations. *IEEE Transactions on Automatic Control*, 50 (2):234–239, 2005.

A. P. Aguiar, J. P. Hespanha, and P. V. Kokotović. Performance limitations in reference tracking and path following for nonlinear systems. *Automatica*, 44(3):598–610, 2008.

G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, E. De Lellis, and A. Pironti. Path generation and tracking in 3-d for uavs. *IEEE Transactions on Control Systems Technology*, 17(4):980–988, 2009.

A. Bhatia, M. Graziano, S. Karaman, R. Naldi, and E. Frazzoli. Dubins trajectory tracking using commercial off-the-shelf autopilots. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 6300, 2008.

C. Cao, N. Hovakimyan, V. V. Patel, I. Kaminer, and V. Dobrokhodov. Stabilization of cascaded systems via l1 adaptive controller with application to a uav path following problem and flight test results. In *2007 American Control Conference*, pages 1787–1792. IEEE, 2007.

W.-H. Chen. Disturbance observer based control for nonlinear systems. *IEEE/ASME transactions on mechatronics*, 9(4):706–710, 2004.

G. Chowdhary, E. Frazzoli, J. How, and H. Liu. Nonlinear flight control techniques for unmanned aerial vehicles. *Handbook of Unmanned Aerial Vehicles, Springer, Houten*, 2014.

B. R. Copeland. The design of pid controllers using ziegler nichols tuning. *Internet: http://educypedia. karadimov. info/library/Ziegler_Nichols. pdf*, 2008.

L. Ding and Z. Wang. A robust control for an aerial robot quadrotor under wind gusts. *Journal of Robotics*, 2018, 2018.

W. Dong, G.-Y. Gu, X. Zhu, and H. Ding. Modeling and control of a quadrotor uav with aerodynamic concepts. In *Proceedings of World Academy of Science, Engineering and Technology*, number 77, page 437. World Academy of Science, Engineering and Technology (WASET), 2013.

P. Encarnaçao and A. Pascoal. Combined trajectory tracking and path following: an application to the coordinated control of autonomous marine craft. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*, volume 1, pages 964–969. IEEE, 2001.

E. W. Frew, D. A. Lawrence, and S. Morris. Coordinated standoff tracking of moving targets using lyapunov guidance vector fields. *Journal of guidance, control, and dynamics*, 31(2):290–306, 2008.

X. Gong, Y. Tian, Y. Bai, and C. Zhao. Trajectory tacking control of a quad-rotor based on active disturbance rejection control. In *2012 IEEE International Conference on Automation and Logistics*, pages 254–259. IEEE, 2012.

J. Han. From pid to active disturbance rejection control. *IEEE transactions on Industrial Electronics*, 56(3):900–906, 2009.

A. J. Healey and D. Lienard. Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles. *IEEE journal of Oceanic Engineering*, 18(3):327–339, 1993.

R. Hopkins and Y. Xu. Position tracking control for a simulated miniature helicopter. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 6485, 2008.

S. Jackson, J. Tisdale, M. Kamgarpour, B. Basso, and J. K. Hedrick. Tracking controllers for small uavs with wind disturbances: Theory and flight results. In *2008 47th IEEE Conference on Decision and Control*, pages 564–569. IEEE, 2008.

I. Kaminer, A. Pascoal, E. Hallberg, and C. Silvestre. Trajectory tracking for autonomous vehicles: An integrated approach to guidance and control. *Journal of Guidance, Control, and Dynamics*, 21(1):29–38, 1998.

I. Kaminer, O. Yakimenko, A. Pascoal, and R. Ghabcheloo. Path generation, path following and coordinated control for timecritical missions of multiple uavs. In *2006 American Control Conference*, pages 4906–4913. IEEE, 2006.

D. Kotarski, Z. Benić, and M. Krznar. Control design for unmanned aerial vehicles with four rotors. *Interdisciplinary Description of Complex Systems: INDECS*, 14(2):236–245, 2016.

R. Li, T. Li, R. Bu, Q. Zheng, and C. Chen. Active disturbance rejection with sliding mode control based course and path following for underactuated ships. *Mathematical Problems in Engineering*, 2013, 2013.

C. Liu, W.-H. Chen, and J. Andrews. Tracking control of small-scale helicopters using explicit nonlinear mpc augmented with disturbance observers. *Control Engineering Practice*, 20(3):258–268, 2012.

C. Liu, O. McAree, and W.-H. Chen. Path-following control for small fixed-wing unmanned aerial vehicles under wind disturbances. *International Journal of Robust and Nonlinear Control*, 23(15):1682–1698, 2013.

R. Madoński, M. Kordasz, and P. Sauer. Application of a disturbance-rejection controller for robotic-enhanced limb rehabilitation trainings. *ISA transactions*, 53(4):899–908, 2014.

D. W. Mellinger. Trajectory generation and control for quadrotors. 2012.

D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529, 2007.

W. Ren and R. W. Beard. Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints. *IEEE Transactions on Control Systems Technology*, 12(5): 706–716, 2004.

R. Rysdyk. Unmanned aerial vehicle path following for target observation in wind. *Journal of guidance, control, and dynamics*, 29(5):1092–1100, 2006.

X. Wang, W. Kong, D. Zhang, and L. Shen. Active disturbance rejection controller for small fixed-wing uavs with model uncertainty. In *2015 IEEE International Conference on Information and Automation*, pages 2299–2304. IEEE, 2015.

Z. Wang, W. Yang, and G. Ding. Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots based on neural dynamic model. In *2010 Second WRI Global Congress on Intelligent Systems*, volume 2, pages 270–273. IEEE, 2010.

S. Xingling and W. Honglun. Back-stepping active disturbance rejection control design for integrated missile guidance and control system via reduced-order eso. *ISA transactions*, 57:10–22, 2015.

L. Xu, D. Guo, and H. Ma. Cascade active disturbance rejection control for quadrotor uav. In *2019 Chinese Control Conference (CCC)*, pages 8044–8048. IEEE, 2019.

L. Xu, H. Ma, D. Guo, et al. Backstepping sliding-mode and cascade active disturbance rejection control for a quadrotor uav. *IEEE/ASME Transactions on Mechatronics*, 2020.

J. Yang, C. Liu, M. Coombes, Y. Yan, and W.-H. Chen. Optimal path following for small fixed-wing uavs under wind disturbances. *IEEE Transactions on Control Systems Technology*, 2020.

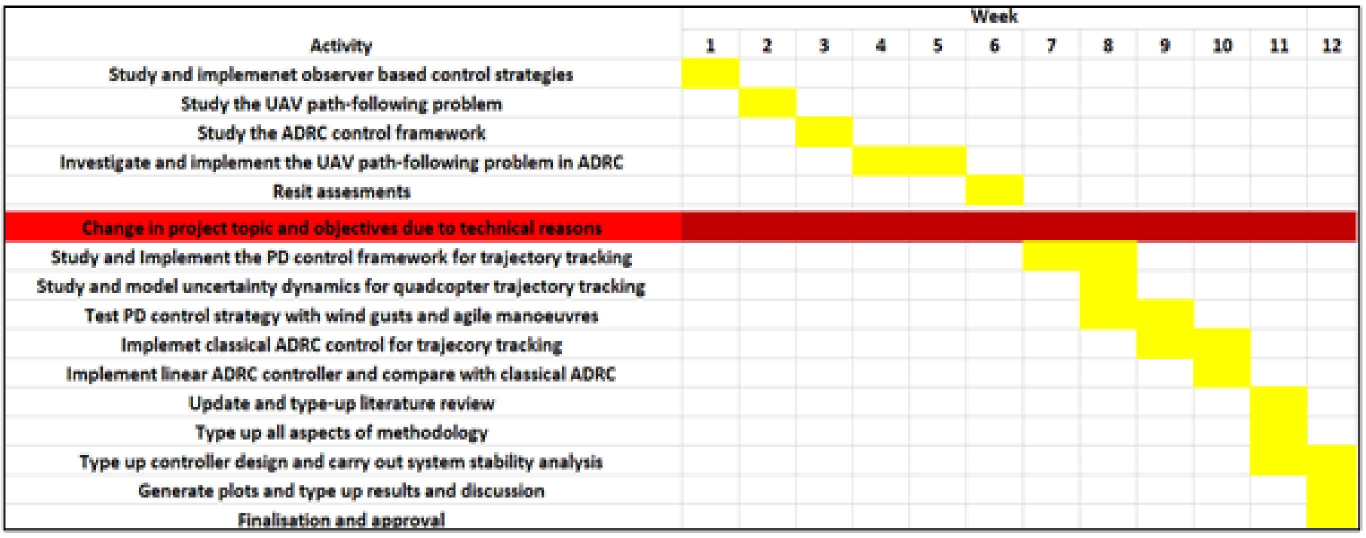
X. Zhang, H. Wang, X. Shao, and C. Liu. Precise trajectory tracking for uav based on active disturbance rejection control. In *2015 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 464–469. IEEE, 2015.

Y. Zhang, Z. Chen, M. Sun, and X. Zhang. Trajectory tracking control of a quadrotor uav based on sliding mode active disturbance rejection control. *Nonlinear Analysis: Modelling and Control*, 24(4):545–560, 2019.

Z. Zuo. Adaptive trajectory tracking control of a quadrotor unmanned aircraft. In *Proceedings of the 30th Chinese Control Conference*, pages 2435–2439. IEEE, 2011.

Appendices

A Project Timeline



Activity	Week 1	2	3	4	5	6	7	8	9	10	11	12
Study and implemenet observer based control strategies	🟨											
Study the UAV path-following problem		🟨										
Study the ADRC control framework			🟨									
Investigate and implement the UAV path-following problem in ADRC				🟨	🟨							
Resit assesments						🟨						
Change in project topic and objectives due to technical reasons	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Study and Implement the PD control framework for trajectory tracking							🟨	🟨				
Study and model uncertainty dynamics for quadcopter trajectory tracking								🟨				
Test PD control strategy with wind gusts and agile manoeuvres								🟨	🟨			
Implemet classical ADRC control for trajecory tracking									🟨	🟨		
Implement linear ADRC controller and compare with classical ADRC										🟨		
Update and type-up literature review											🟨	
Type up all aspects of methodology											🟨	
Type up controller design and carry out system stability analysis											🟨	🟨
Generate plots and type up results and discussion												🟨
Finalisation and approval												🟨

Figure 43: project timeline

B Classical ARDC Code and Simulink Block

```
function [sys,x0,str,ts] = quadcopteradrccontroller(t,x,u,flag)

switch flag

  case 0
    [sys,x0,str,ts] = mdlInitializeSizes;

  case 1
    sys=mdlDerivatives(t,x,u);

  case 3
    sys=mdlOutputs(t,x,u);
  case { 2, 4, 9 }
    sys = [];

  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 12;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 10;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
str = [];
ts  = [0 0];
function sys=mdlDerivatives(t,x,u)
g    = 9.81;
l    = 0.046;
Ixx  = 1.43e-5;
Iyy  = 1.43e-5;
Izz  = 2.89e-5;
m    = 0.03;
% fx = 0.1 + 1*sin(2*pi);
% fy = 0.1 + 1*sin(2*pi);
% fz = 0.1 + 1*sin(2*pi);
% f_phi = 3.2;
% f_theta = 3.2;
% f_psi = 0.055;
```

```
x1dot = x(2);
x2dot = fx - u(1)/m * cos(x(18)) * sin(x(15)) * cos(x(12))
+ sin(x(18)) * sin(x(12));
x3dot = x(5);
x4dot = fy - u(1)/m * sin(x(18)) * sin(x(15)) * cos(x(12))
- cos(x(18)) * sin(x(12));
x5dot = x(8);
x6dot = fz - u(1)/m * cos(x(12)) * cos(x(15)) - g;
x7dot = x(11);
x8dot = f_phi + u(2)/Ixx + (Iyy - Izz)/Ixx * x(15) * x(18) + l/Ixx;
x9dot = x(14);
x10dot = f_theta + u(3)/Iyy + (Izz - Ixx)/Iyy * x(12) * x(18) + l/Iyy;
x11dot = x(17);
x12dot = f_psi + u(4)/Izz + (Ixx - Iyy)/Izz * x(12) * x(15) + l/Izz;


sys        = [x1dot; x2dot; x3dot; x4dot; x5dot; x6dot;
    x7dot; x8dot; x9dot; x10dot; x11dot; x12dot];

function sys=mdlOutputs(t,x,u)
u1 = u(1);
u2 = u(2);
u3 = u(3);
u4 = u(4);

                % phi / theta / psi / z / x / y /
sys = [x(1); x(4); x(7); x(10); x(13); x(16); u1; u2; u3; u4];

function [sys,x0,str,ts] = eso(t,x,u,flag)



switch flag
```

```
  case 0
    [sys,x0,str,ts]=mdlInitializeSizes();


  case 1
    sys=mdlDerivatives(t,x,u);


  case 3
    sys=mdlOutputs(t,x,u);

  case { 2, 4, 9 }
    sys = [];

  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end


function [sys,x0,str,ts]=mdlInitializeSizes()

sizes = simsizes;
sizes.NumContStates  = 3;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = [0; 0; 0];
```

```
str = [];
ts  = [0 0];

function sys=mdlDerivatives(t,x,u)
b0 = 0.83;

delta  = 0.001;

hg     = 0.15;
beta01 = 1;
beta02 = 1/(2*(hg^0.5));
beta03 = 2/(25*(hg^1.2));
beta01 = beta01*100;
beta02 = beta02*100;
beta03 = beta03*100;
e      =  x(1) - u(1);

fe = fal(e, 0.5, delta);
fe1 = fal(e, 0.25, delta);

xdot1 = x(2) - beta01 * e;
xdot2 = x(3) - beta02 * fe + b0 * u(2);
xdot3 =  - beta03 * fe1;



sys = [xdot1; xdot2; xdot3];



function sys=mdlOutputs(t,x,u)

x(1);
x(2);
```

```
x(3);


sys = [x(1); x(2); x(3); x(1)];

function [sys,x0,str,ts] = eso(t,x,u,flag)


switch flag

  case 0
    [sys,x0,str,ts]=mdlInitializeSizes();



  case 1
    sys=mdlDerivatives(t,x,u);



  case 3
    sys=mdlOutputs(t,x,u);

  case { 2, 4, 9 }
    sys = [];

  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end



function [sys,x0,str,ts]=mdlInitializeSizes()

sizes = simsizes;
```

```
sizes.NumContStates  = 3;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = [0; 0; 0];
str = [];
ts  = [0 0];

function sys=mdlDerivatives(t,x,u)
b0 = 0.83;

delta  = 0.001;

hg     = 0.15;
beta01 = 1;
beta02 = 1/(2*(hg^0.5));
beta03 = 2/(25*(hg^1.2));
beta01 = beta01*100;
beta02 = beta02*100;
beta03 = beta03*100;
e      =  x(1) - u(1);

fe = fal(e, 0.5, delta);
fe1 = fal(e, 0.25, delta);

xdot1 = x(2) - beta01 * e;
xdot2 = x(3) - beta02 * fe + b0 * u(2);
xdot3 =  - beta03 * fe1;
```

```
sys = [xdot1; xdot2; xdot3];


function sys=mdlOutputs(t,x,u)

x(1);
x(2);
x(3);


sys = [x(1); x(2); x(3); x(1)];

function [sys,x0,str,ts] = nlsef(t,x,u,flag)


switch flag

  case 0
    [sys,x0,str,ts]=mdlInitializeSizes();

  case 1
    sys=mdlDerivatives(t,x,u);

  case 3
    sys=mdlOutputs(t,x,u);


  case { 2, 4, 9 }
    sys = [];
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

function [sys,x0,str,ts]=mdlInitializeSizes()

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];


function sys=mdlOutputs(t,x,u)
%c  = 2;
r  = 100;
h  = 0.01;



e1 = u(1) - u(2);
e2 = u(3) - u(4);
u0 = -fhan(e1,e2,r,h);

sys = u0;
```

```
function [ fal ] = fal(e,a,delta)
if abs(e) > delta
    fal = ((abs(e))^a) * sign(e);
else
    fal = e/(delta^(1-a));
end


end

function [ fhan ] = fhan( x1, x2, r, h )


d  = r*h;
d0 = h*d;
y  = x1 + h*x2;
a0 = sqrt((d^2) + 8*r*(abs(y)));


if abs(y) > d0
    a = x2 + ((a0 - d)/2)*sign(y);
else
    a = x2 + (y/h);
end


if abs(a) > d
    fhan = -r*sign(a);
else
    fhan = -r*a/d;
end


end
```

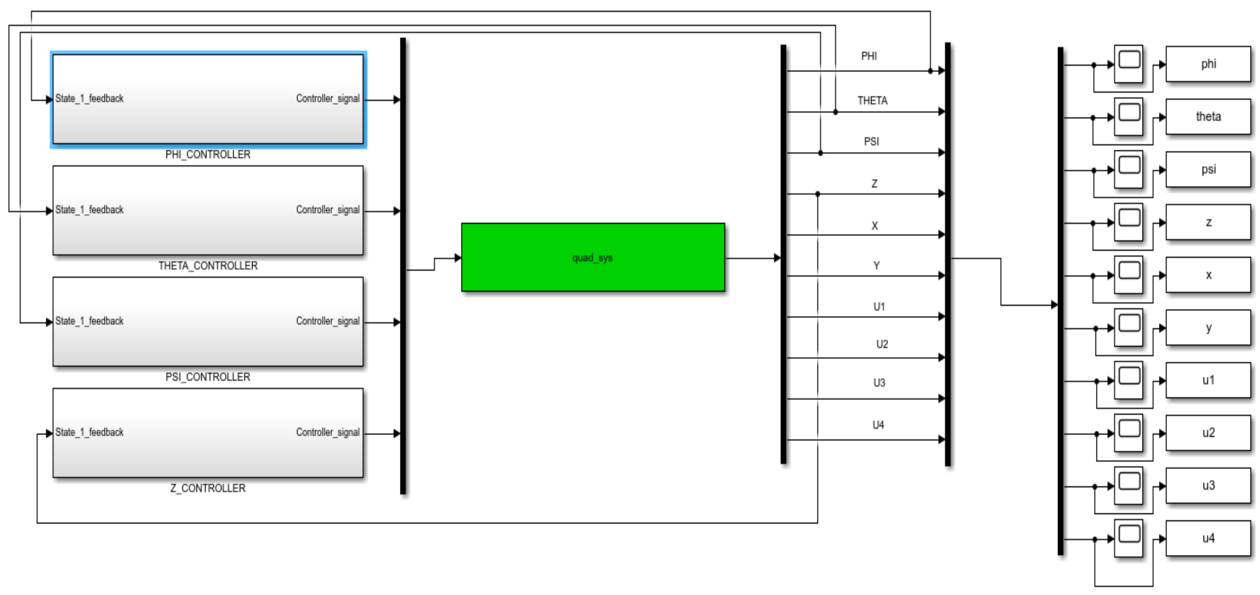


Figure 44: classical adrc simulink block

C Linear ARDC Code and Simulink Block

```
function [sys,x0,str,ts] = quadrotoradrc(t,x,u,flag)

switch flag

  case 0
    [sys,x0,str,ts] = mdlInitializeSizes;

  case 1
    sys=mdlDerivatives(t,x,u);

  case 3
    sys=mdlOutputs(t,x,u);

  case { 2, 4, 9 }
    sys = [];
    otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 24;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 16;
sizes.NumInputs      = 16;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);

x0  = [0; 0; 0; 0; 0; 0; 0; 0; 5; 0; 0;
    0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;];
% x0 = zeros(1,24);
str = [];
ts  = [0 0];

function sys = mdlDerivatives(t,x,u)
%% Defining system parameters
g   = 9.81;
l   = 0.45;
omega_r  = 0.1;
Jrr  = 6e-3;
Ixx = 0.018125;
Iyy = 0.018125;
Izz = 0.035;
m   = 2;
dz = 0;
a11 = (Iyy-Izz)/Ixx;
```

```
a22 = Jrr/Ixx;
a33 = (Izz-Ixx)/Iyy;
a44 = Jrr/Iyy;
a55 = (Ixx-Iyy)/Izz;
b11 = l/Ixx;
b22 = l/Iyy;
b33 = 1/Izz;
bb = -(cosd(x(1))*cosd(x(3)))/m;
%% Gain variables for linear observer

L1 = 29.5659;
L2 = 2907;
L3 = 3000;

%% Defining quadcopter system dynamics

xdot     = [x(2);
             x(4)*x(6)*a11 - x(4)*omega_r*a22 + b11*u(2) + dz1;
             x(4);
             x(2)*x(6)*a33 + x(2)*omega_r*a44 + b22*u(3) + dz2;
             x(6);
             x(2)*x(4)*a55 + b33*u(4) + dz3;
             x(8);
             - g - (u(1)/m)*(cosd(x(1))*cosd(x(3))) + dz;
             x(10)
- (u(1)/m)*(sind(x(1))*sind(x(5)) + cosd(x(1))*sind(x(3))*cosd(x(5))) + dz;
             x(12);
- (u(1)/m)*(sind(x(1))*cosd(x(5)) - cosd(x(1))*sind(x(3))*sind(x(5))) + dz;
             x(14) + L1*(x(1)-x(13));
             x(15) + L2*(x(1)-x(13)) + b11*u(2);
                  L3*(x(1)-x(13));
             x(17) + L1*(x(3)-x(16));
```

```
              x(18) + L2*(x(3)-x(16)) + b22*u(3);
                      L3*(x(3)-x(16));
              x(20) + L1*(x(5)-x(19));
              x(21) + L2*(x(5)-x(19)) + b33*u(4);
                      L3*(x(5)-x(19));
              x(23) + L1*(x(7)-x(22));
              x(24) + L2*(x(7)-x(22)) + bb*u(1);
                      L3*(x(7)-x(22))];
sys       = xdot;

function sys = mdlOutputs(t, x, u)
m   = 2;
l   = 0.45;
Ixx = 0.018125;
Iyy = 0.018125;
Izz = 0.035;
b11  = l/Ixx;
b22  = l/Iyy;
b33  = 1/Izz;
bb   = -(cosd(x(1))*cosd(x(3)))/m;

%% Defining x and y position and velocity state

x_pos = 5*cos(2*pi*0.1*t);  % desired x position
y_pos = 5*sin(2*pi*0.1*t);  % desired y position

x_vel = 0; % desired x velocity
y_vel = 0; % desired y velocity



%% Assigning PD controller gain variables
```

```
kpp = u(5);   kdp = u(6);
kpt = u(7);   kdt = u(8);
kpk = u(9);   kdk = u(10);
kpz = u(11); kdz = u(12);
kpx = u(13); kdx = u(14);
kpy = u(15); kdy = u(16);

%% Defining initial y-phi and x-theta values

 phid   =  kpy*(y_pos - x(11)) + kdy*(y_vel - x(12));
 thetad =  kpx*(x_pos - x(9))    + kdx*(x_vel - x(10));

%% Control Action for Theta, Phi, Psi and Altitude Channels
%
zd     = t; % desired z position
zddot = 0;
u01       = kpz*(zd - x(7)) + kdz*(zddot - x(8));
u1        = (u01 - x(24))/bb;

phid      = cosd(x(5))*phid - sind(x(5))*thetad; % desired psiangle
phiddot = 0;
u02       = kpp*(phid - x(1)) + kdp*(phiddot - x(2));
u2        = (u02 - x(15))/b11;

thetad     = sind(x(5))*phid + cosd(x(5))*thetad; % desired theta angle
thetaddot = 0;
u03        = kpt*(thetad- x(3)) + kdt*(thetaddot - x(4));
u3         = (u03 - x(18))/b22;



psid     = 1; % desired psi angle
psiddot = 0;
```

```
u04      = kpk*(psid - x(5)) + kdk*(psiddot - x(6));
u4       = (u04 - x(21))/b33;

%%
sys = [x(1); x(3); x(5); x(7); x(9); x(11); u1; u2; u3; u4; x_pos; y_pos; z
```

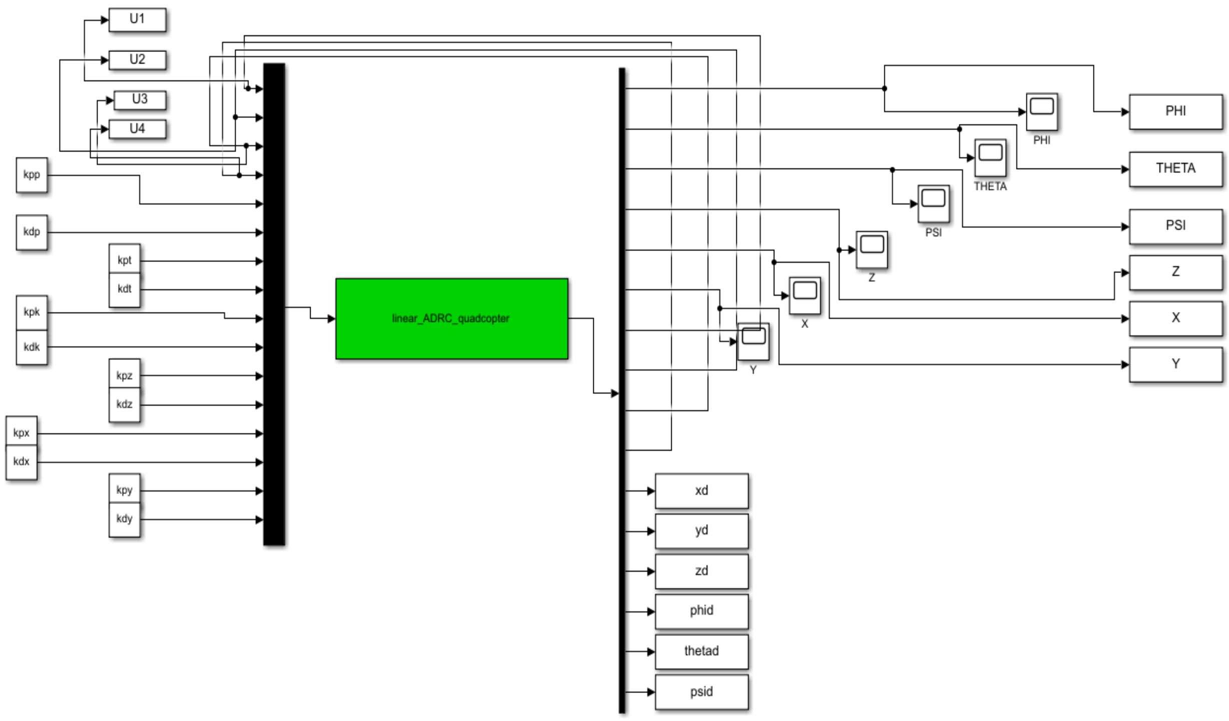


Figure 45: linear adrc simulink block