A Thesis
entitled

Development of a Neural PD Controller for Quad-rotors for Rejection of Wind Disturbances

by

Jisheng Li

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the
Master of Science Degree in Mechanical Engineering

_______________________________________
Dr. Manish Kumar, Committee Chair

_______________________________________
Dr. Mohammad Elahinia, Committee Member

_______________________________________
Dr. Abdollah A. Afjeh, Committee Member

_______________________________________
Dr. Patricia R. Komuniecki, Dean

College of Graduate Studies

The University of Toledo

May 2016
Copyright © 2016, Jisheng Li

This document is copyrighted material. Under copyright law, no parts of this document may be reproduced without the expressed permission of the author.
An Abstract of

Development of a Neural PD Controller for Quad-rotors for Rejection of Wind Disturbances

by

Jisheng Li

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Mechanical Engineering

The University of Toledo

May 2016

UAVs have become increasingly popular around the world both in civilian and military application within the past few years. UAV applications are essentially fueled by advances in a combination of technologies, such as communication, embedded systems, processing, sensing and algorithms. This thesis focuses on the control aspect of UAVs, particularly lots of work has been carried out in this area recently. This thesis mainly focuses on the performance of control algorithms under wind disturbances. Recent works that include an adaptive controller and a PD controller are discussed. Then the thesis proposes a neural PD controller, in which the neural network is added to the outer PD control loop. A comparison between neural PD controller and PD controller under various scenarios of wind disturbances is carried out. The results show neural PD controller performs better than the PD controller in position tracking under wind disturbances.

Keywords: Adaptive controller, PD controller, neural PD controller, wind disturbance.
Acknowledgments

Firstly, I would like to thank my family who supports me to study in US. They usually tell me not to worry about other stuff and just put your time on your work. And also they give lots of useful information about the future career. This gives a lot of encouragement since studying abroad is not easy and makes me confused about the future sometimes.

Meanwhile, I would be grateful for my advisor's help. When I begin my study at college of engineering at UT, Dr Kumar gives me lots of help in study. He is so patient for helping me solving my problems in his course. When stepping into the research, I usually get the useful advices related to the paper I am working on, which makes me have a better understanding of the paper. In my mind, he is a nice and knowledgeable professor who deals with both life and work very well.

Also, I would thank my committee members. They give me a lot of useful comment, which helps me recognize my work from many different perspectives.

Finally, I am very glad to know the friends in my lab. They keep company with me and give me advice for the paper and life in US. I also learn what team work is and many interesting cultures. I believe these would be very helpful for my future. I am very happy to have such a good group in our CDS lab.
# Table of contents

Abstract...........................................................................................................................................iii

Acknowledgments..................................................................................................................................iv

Table of contents.................................................................................................................................v

1 Introduction.........................................................................................................................................1

1.1 General introduction of UAVs.......................................................................................................1

1.2 Problem motivation.......................................................................................................................5

1.3 Outline of the document...............................................................................................................6

2 Literature review...............................................................................................................................8

3 Quad-rotor dynamics analysis.........................................................................................................14

4 Adaptive controller for the quad-rotor control...............................................................................23

4.1 The adaptive position controller.................................................................................................28

4.2 The adaptive attitude controller.................................................................................................30

4.3 Simulation results for adaptive controller..................................................................................35

5 PD controller for the quad-rotor control.........................................................................................41
5.1 The PD controller..............................................................41

5.2 Simulation results for PD controller........................................44

6 RBF Neural Network PD controller design..................................48

6.1 The structure of the neural PD position controller......................51

6.2 The neural PD position controller..........................................53

6.3 Simulation results for neural PD controller...............................58

7 Comparison between PD controller and

neural PD controller under wind disturbance.................................63

8 Conclusions and future work....................................................73

8.1 Conclusions........................................................................73

8.2 Future work.......................................................................74

References..................................................................................76

A Appendix A Codes for neural PD controller and PD controller .............83

B Appendix B Codes for adaptive controller...................................94
Chapter 1

Introduction

1.1 General introduction of UAVs

UAV, Unmanned Aerial Vehicle, is a kind of aircraft without a human pilot. Its flight is controlled either by onboard computers or by the remote ground station. And these years, there is much research related to UAV, since UAV has become more and more useful for military, and even for average people, as the technology has become more and more advanced. Because technology is accumulated step by step, then learning about the history of UAV is necessary, and this history will be presented.

In 1917, the first UAV that was born in the world was invented by British. It was a small, one-wing aircraft which was controlled wirelessly. However, the experiment for flying the aircraft failed since the technology in some aspects was not good enough. Then, around 1930, the aircraft finally was successful flying by a series of efforts. In the following decade, the automatic aircraft was popularly used as flying targets. After 1950, UAV, the automatic aircraft, began to be applied in the military: "In the early 1950s, Ryan Aeronautical Company produced 32 jet-propelled, subsonic UAVs which are known as Ryan "Firebees"[1]. The Firebee design goes through UAV history. This design was used initially as a target drone. The political fallout of Francis Gary Powers being
shot down over the USSR in his U-2 in 1960 caused people in the DoD to start thinking about unmanned reconnaissance of the USSR. Ryan Aeronautical turned some of its standard Firebee training targets into reconnaissance UAVs (recon-UAVs) and designated them the 147A "Firefly". Afterwards, the Firefly, renamed the "Lightning Bug," was modified to change it from a target drone to a recon-UAV. In the early 1960s, Ryan Aeronautical Company generated more than 20 versions of its famous Lightning Bug unmanned subsonic target drones. Among the drones, the AQM-34N had wet wings, meaning that it carried fuel in its wings, give it a range of approximately 2500 miles. And in the Vietnam War, the US totally used 3435 operational reconnaissance UAVs to operate tasks from 1964 to 1975. Around 30 percent of tasks were completed by different types of Lightning Bug."[1] Actually, the UAV has been used in military for some time and becomes more and more popular around the world. Presently, many countries try to build up their own UAV to keep their security superior.

Additionally, the civilian application of UAV has increased as well. There is a TV show called "where are we going, dad" in China. In 2013, UAV was used to get some different perspectives on videos in this show. In 2015, Amazon received permission for testing delivering goods with UAV in the US and applied the patent for its UAV delivery system. During the same year, Michigan State Police become the first law enforcement agency to be permitted by the FAA, using UAV in every corner of the state in the US. In the future, the civil application of UAV will be involved in many fields, such as monitoring disaster in agriculture, searching and rescuing, taking aerial photos, and delivering goods, based on its powerful function. Thus, in the civil application of UAV, there are potentially large commercial benefits.
Presently, many people and companies realize the large development potential of civil UAV and begin to work on the technology of UAV in order to develop good UAV products and offer better services for people's daily lives. To gain a better understanding of development situations of civil UAV, some good companies related to the UAV will be introduced as follows:

DJ-Innovations Company was established in 2006, which was located in Shenzhen, China. Its main products include Inspire 1 and Phantom 3, which are used for capturing aerial video or pictures.

Parrot Company was established in 1994, which is located in Paris, France. The company has UAV products, such as Parrot Bebop Drone and AR Drone. Parrot's drones are very popular with hobbyists.

3DR Company is located in Berkeley, US. The company sells UAV products, such as 3DR IRIS and Solo which was published in 2015. Solo has much powerful function when taking videos and photos, like cable cam allowing you to lock solo in a specific line between two points in the air.

All the products these companies sell are used for aerial photos whose field will be grown into a very large market in the near future since aerial photos cannot only take different perspective photos, which used to be expensive before, but also satisfy most people's interest. Although the UAV that all the above companies sell are classified into quad rotors, actually, there are different kinds of UAV that are applied around the world. Then, the classification of UAV will be brought. In fact, there are many ways to classify the UAV. Two ways of classification are as follows:
Classification based on application domain:

For military application, UAV is required to have better sensitivity, higher flight altitude, higher speed and more intelligence, which represent superior advancements in technique.

For civil application, UAV is required to have a lower speed, smaller cruising ability, and lower altitude compared to military application. However, the industries for civil UAV need to offer low price components and services to increase their competition among companies. The civil application includes police application and fire fighting application, etc.

For consuming application, UAV is mostly used for aerial photos and entertainment.

Classification Based on Structure:

UAVs are mainly divided into three types: fixed wing, helicopters, and multi-rotors.

These three UAV types have their own advantages and disadvantages. Fixed wing UAVs are able to use the lift that is generated by their wings, which leads to better endurance and payload. Moreover, fixed wing UAVs move faster compared to the other two kinds of UAV. Although, they have many advantages, fixed wing UAVs are not agile enough. Compared to fixed wing UAVs, helicopters are more sensitive, and they have better payload ability. Meanwhile, due to their driven systems, helicopters have better endurance than multi-rotors, and they do well in wind. However, helicopters have their own disadvantages. Their mechanical structures are more complex to build than those of multi-rotors and it is difficult to remove their vibrations when they are flying. As
for multi-rotors, their structures are simple and easy to build, and they are very agile. They are especially suitable for flying in small spaces. As technology advances, there is much space for multi-rotor development and the further improvement of flying performance. Nevertheless, there are still some limitations for multi-rotors at present. For example, there is not enough power to operate the multi-rotors for a long time because there are lots of devices on it that require electricity, such as sensors and onboard computers. They also require the lifting energy for their own weight. Based on these advantages and disadvantages, different kinds of UAVs are applied in different fields.

In order to fly a UAV, many components need to be installed. In general, a UAV system includes: an autopilot, an onboard computer, a receiver, a radio controller, a GPS module, motors, power systems, telemetry equipment, and the ground station. Algorithms guide UAVs to complete their missions. That is, when the components and algorithms that are used are more advanced, the UAV is more capable of completing complex tasks. For instance, a UAV is capable of avoiding the obstacles automatically, following a specific path, tracking a moving object, reaching a specific position that has been set in advance, and so on. A UAV may even be like a smart buddy helping you cope with your problems.

1.2 Problem motivation

Recently, Germany introduced the Industrial 4.0, which makes industries even more automatic and intelligent. Automatic and intelligent machines or robotics will become the main workforce in the future. For now, in automotive companies, arm robots have been widely used to weld and assemble cars. Robot floor cleaners help to clean floors
automatically and intelligently, and have become more and more popular among families. In some public areas, robot servants have begun to serve average people. In addition, UAVs, a type of robot, are able to be applied in searching, aerial photography, and military uses. Robots are getting an increasing amount of applications for serving people in some fields. Sometimes, robots can replace people in risky or dangerous situations. That is to say, robotics plays an increasingly important role in people’s lives and will be a large market in the future. Thus, all kinds of technology used for robots prove to be more and more valuable. Among the advanced and complicated technology for robots, control methods is a critical and valuable branch that stands for autonomy and intelligence. As a result, this thesis focuses on controlling the motion of the UAV to follow a desired path. Specifically, the thesis focuses on designing controllers that can enable the UAV to operate in wind disturbances since any outdoor application of UAVs would require it to be robust against wind disturbances.

1.3 Outline of the document

This thesis is divided into the following five main parts:

   1) Dynamics Model for the Quad-Rotor
   2) Adaptive Controller for the Quad-Rotor
   3) PD Controller for the Quad-Rotor
   4) Neural PD Controller for the Quad-Rotor
   5) Comparison between the neural PD controller and PD controller under wind disturbances
The thesis presents the UAV dynamics and an adaptive and PD controller available in literature. Then it presents the proposed Neural-PD controller developed as a part of the thesis research. Finally, the numerical results obtained from implementation of the proposed technique is presented and compared with other traditional controllers used in literature. In the thesis, the programming in Matlab is used, and all the results are the simulation results from Matlab.
Chapter 2

Literature review

In recent years, UAVs have become increasingly popular. This thesis focuses on quad-rotors. A quad-rotor has recently become widely popular because of its application, which brings a new feeling to people’s daily life. At the same time, different kinds of functionalities have been invented which make quad-rotors more and more powerful. For example, quad-rotors possess better ears and eyes and have become more intelligent. Ears and eyes belong to the sensor capability. Intelligence is equal to the plan, decision, and control algorithm. Thus, an intelligent control system is a process that can be described as follows: Firstly, the task is set up; secondly, all the information that is needed in the process of operating the task is picked up by the sensors; thirdly, all the information picked up by sensors is conveyed to the onboard computer through the communication technique and then used in the task algorithm; fourthly, data obtained from the task algorithm is conveyed to control algorithm; and finally, another set of data gained from control algorithm is sent to actuator and the system will run as desired. In general, inside control system, there exists a suspended mechanism to response to emergent situations. As a consequence, one of these steps above can be improved to enhance the performance of the quad-rotor.
The length of micro UAVs (such as quad-rotors) are from 0.1 m to 0.5 m and the mass of those are from 0.1 kg to 0.5 kg [2]. Some teams have worked on micro UAVs in the 10 cm range [3], [4]. One of the smallest UAVs is the Picoflyer whose propeller diameter is 60 mm and mass is 3.3 g [5]. The 50 cm platform UAVs are more popular and a few groups have focused on systems of this size [6], [7]. When doing research, test beds are very important, since they are influenced by the environment situations and some other UAVs that work together with those test beds. Some teams have worked with outdoor test beds [8], [9] as well as indoor environments [10]. McKerrow [11] studied the dynamic model of the VTOL quad-rotor. Naidoo et al. [12] studied modeling of the quad-rotor. Hamel et al. [13] modified the gyroscopic torque expression of the dynamic model in [11]. The dynamic model is inevitable in control problems. The essence of control is actually simulating the dynamic model of the system in the onboard computer and then figure out the controls to get the desired output. Therefore, dynamic models of the system are necessary.

Working on the control problem under wind disturbance for the quad-rotor, we need to look into two areas. One is the control method that can be used on the quad-rotor, and the other is how is the performance of the quad-rotor under wind disturbance, that is, the endurance for wind disturbance.

Firstly, consider the control method. If we focus on control algorithms, a lot of valuable research materials are found. PID controllers are very popular for controlling quad-rotors. For example, Michael et al. [14] studied PD controller for the quad-rotor. In this paper, the author used the linearization of the nonlinear mathematic model. Erginer et al. [15] also used the PD controller to control the quad-rotor, but with a different way of
describing the controls. Salih et al. [16] used the simulation method to express the
performance of PD controller. LQR, running a system with minimum cost, is another
useful method of control that belongs to the optimal control method. LQR [17] is used in
the attitude control which helps improve the stabilization and robustness under bounded
disturbances. In [18], the authors have introduced the combination of LQR and formation
flight, which combines the decision layer and control layer together. Besides the normal
control methods mentioned above, sliding mode controllers [19], [20], [21], adaptive
controllers [19], nonlinear controllers [19], neural controllers [22], back stepping
controllers [23], [24], robust controllers, and fuzzy controllers are also used for
controlling quad-rotors. In [19], the authors have focused on linearization controller with
a higher sliding mode observer which is operated in parallel with the controller. The
sliding mode observer is used for estimating the parameters of disturbances and an
observer of some measurements which help reduce sensors' cost. In [20], the author pays
more attention to the robustness of the controller under disturbances. The controller is
driven by the sliding mode disturbance observer without using high control gain and
extensive computational power. In [22], the neural network is applied for estimating the
angular and translational velocity of the UAV as well as learning the uncertainties in the
environment. Thus, two neural networks are included that have different functionalities.
In this paper, an idea that can be found is that the intellectual controllers are actually
embedded with some intellectual methods to reach some goals which traditional
controllers cannot achieve. However, for reaching these goals, intellectual methods are
not exclusive. There are also some other mathematical methods that are effective. In [24],
the quad-rotor system is divided into three subsystems. The first one is the horizontal
positions (x, y) with pitch and roll angles. The second one is the vertical z with yaw angle and the final one is the total thrust. Next, the back stepping control is to stabilize the whole system based on Lyapunov stability theory. More generally, combinations of these types of controllers have been invented in order to adapt quad-rotors to different situations, since the combination can improve some specification compared to the single control method [25]-[39]. In [25], the author uses the SISO method to achieve desired attitude and use the classical control method to tune the parameters in PID controller. In [27], the author develops a new state-space representation. A sliding mode controller is developed to insure Lyapunov stability, dealing with nonlinear terms in the system after applying the back stepping approach. In [28], the author introduced a controller with a single layer neural network to estimate the unknown parameters in the dynamic model, and a new adaptive observer to measure the non measurable states. The advantage of this new adaptive observer is reducing the measurement noise compared to traditional high gain observer. In [29], the author focuses on two parts, path planning and fuzzy PID controller. For the fuzzy PID controller, the parameters in PID controller are tuned by fuzzy algorithm. Meanwhile, the parameters in the fuzzy algorithm are tuned through EKF algorithm. The approach for tuning fuzzy algorithm has the advantage of reducing the time cost. In [35], the author mainly focuses on a gain scheduling based PID controller that can be effective even if faults happen during the flight, that is, this type of controller can control the quad-rotor movement smoothly under normal situations and fault situations. There is a decision variable associated with the fault detection and isolated scheme. If the decision variable goes to some certain value, the pre-tuned PID controller will be tuned to be applied to reject the fault. In [37], the authors talk about
approximate-adaptive control. This approach applies updated adaptive parameters and CMAC weights to gain adaption of unknown payloads and robustness of disturbance. As to make the weights turn large enough to compensate the payloads, the method proposed depends on a group of alternative weights to lead the training. However, for e-modification approach, it becomes robust only by limiting the weight growth.

Now we focus on some controllers under wind disturbance, that is, the performance of the system under wind disturbance. Xu et al. [40] built up a model for wind disturbance. Escareño et al. [41] used robust adaptive approach to resist the 2D wind disturbance. Guerrero et al. [42] planned the path of the quad-rotor under some specific wind field. Waslander et al. [43] wrote the position controller for the estimated wind disturbance. In [43], the author focuses on improving the performance in position track under wind disturbance. By modeling the affect on the quad-rotor's dynamics, the wind velocity can be estimated by an estimation algorithm. The purpose of estimating wind velocity is to improve the accuracy of desired position tracking, which is achieved by removing the effect of wind disturbance on the position controller and adding a compensator to reduce the effect of the expected wind disturbance. Hence, all these papers mainly talk about the performance of the UAVs under wind disturbance. Different control methods are explored to see if the specifications required under wind disturbance can be fulfilled or not.

For these controllers, sometimes, we update the parameters in the control law, which makes the control law nonlinear. We may also choose to estimate some parameters in the control system. In addition, some controllers are able to learn from uncertain environments. All of the different control methods were invented to track the desired
value more accurately. Sometimes, for a control system, only one controller may be applied. Furthermore, in general control problems, the situation is more complex which results in multi-controllers working together.

The system controlled by special controllers performs well even under uncertainties including unpredictable uncertainties and predictable uncertainties. However, some methods are not that effective under uncertain environment. The wind disturbance is usually regarded as a type of unpredictable uncertainty, a type of noise as well. Therefore, exploring control methods under wind disturbance needs to consider both the controller’s performance itself and the wind disturbance.

After going through the literature, we can figure out that a lot of control methods have been invented till now. For some new methods, sometimes, it is combining two or more previous control methods together to gain some better specification that is required. Also, if one control method is applied in one field, it can be tested in other fields.
Chapter 3

Quad-rotor dynamics analysis

For UAVs, or any kind of robot, figuring out a dynamic model is necessary because a dynamic modeling is usually the first step for controlling. If a better dynamic model is obtained, then the performance of controls can be better in the same situation. Therefore, dynamics of the robots including UAV is crucial. When a UAV is in the sky, it is driven by torque and thrust. The torque and thrust are generated by the four rotors. Changes in the speed of the four rotors will produce differences in thrust and torque so that the quad-rotor can move as desired. When talking about building up the dynamic model of quad-rotors, there are two popular approaches. One is the Euler-Lagrange approach. The other is called the Newton-Euler approach. For each approach, there are two types of quad-rotor movement called classical motor arrangement and “X-Flyer” motor arrangement, which are shown in Figure (3.1) and Figure (3.2), respectively.
For these two types above, actually they have different front sides when moving, which means the difference of the dynamic models of the two types of the quad-rotors exists. And the torque and thrust of two types of the quad-rotors will be analyzed in Figures (3.3), (3.4) and (3.5), (3.6) respectively.
Figure (3.3) Simplified dynamic model in the translational "X-Flyer" quad-rotor

Figure (3.4) Simplified dynamic model in the rotational "X-Flyer" quad-rotor
Figure (3.5) Simplified dynamic model in the translational classical quad-rotor

Figure (3.6) Simplified dynamic model in the rotational classical quad-rotor
In Figure (3.3), it means the quad-rotor moves towards the forward. The four circles stand for the four rotors. The arrow in the top represents the direction of the quad-rotor. And the wider the arrow is, the faster the speed of the corresponding rotor is. Furthermore, the thrust offered by that rotor is larger. In this figure, the rotational torque is zero, but the thrust in both sides is different which offers the force to push the quad-rotor to result in the translational motion. In Figure (3.4), the quad-rotor is spinning around z axis. The meaning of every pattern is same with those in above figure. This time, the rotational torque is not zero which makes the quad-rotor rotate. Meanwhile, the thrust in both sides are equal. Thus, the quad-rotor rotates around center of the quad-rotor without any translational force ideally. In Figure (3.5), the meaning of the pattern is same with that in Figure (3.3). For the classical quad-rotor, it only uses one rotor to generate the translational force, while the "X-Flyer" quad-rotor uses two rotors to generate the translational force. For the simplified classical quad-rotor dynamic model in Figure (3.6), it is actually same with that in Figure (3.4), but for the different front direction. In this thesis, the classical quad-rotor dynamic model is applied.

In a sense, learning about the dynamic model of robots is essential for control. When controlling a given system, the variables of that need to be controlled have to be figured out. For the controlling of the quad-rotor, indeed there are two aspects of variables that need to be regulated. One is the position, and the other is the attitude, including pitch, roll, and yaw. Figure (3.7) shows what pitch, row and yaw really are.
Figure (3.7) Simplified diagram for attitude

In Figure (3.7), there are two coordinate systems, one is the world coordinate system $O_w$NED and the other is the body coordinate system $Oxyz$, which is fixed in the center of mass of the quad-rotor. The attitude, pitch, roll and yaw, are actually the angles generated by orthogonally rotating the body coordinate along with the three axes in the world coordinate system, which are also called Euler angles $\Theta = (\phi, \theta, \psi)^T$. Equations (3.1)-(3.7) are applied in the paper [44] written by Zuo. Next, when talking about the kinematic equations, it is convenient to be in the absolute (inertial or world) coordinate system. Thus, the rotation matrix from body to word coordinate system is mentioned as follows,

$$
R = \begin{bmatrix}
  c\theta \cdot c\psi & s\theta \cdot c\psi \cdot s\phi - s\psi \cdot c\phi & s\theta \cdot s\psi \cdot c\phi + s\psi \cdot s\phi \\
  c\theta \cdot s\psi & s\theta \cdot s\psi \cdot s\phi + c\psi \cdot c\phi & s\theta \cdot s\psi \cdot c\phi - c\psi \cdot s\phi \\
  -s\theta & c\theta \cdot s\phi & c\theta \cdot c\phi
\end{bmatrix}
$$

(3.1)

Where $s* \equiv \sin *$, $c* \equiv \cos *$. The quad-rotor kinematic equations are shown as

$$
\dot{p} = v
$$

(3.2)
\[
\ddot{\Theta} = W \Omega
\]  
(3.3)

where \( p \) represents the absolute position of the origin of the body coordinate system, and \( v \) is the absolute velocity. \( \Omega \) is the angular velocity in the body coordinate system, and \( W \) is shown in equation (3.4)

\[
W = \begin{bmatrix}
1 & s\phi t\theta & c\phi t\theta \\
0 & c\phi & -s\phi \\
0 & s\phi/c\theta & c\phi/c\theta
\end{bmatrix}
\]  
(3.4)

Where \( t* \leftrightarrow \tan^* \), \( s* \leftrightarrow \sin^* \), \( c* \leftrightarrow \cos^* \), and \( \theta \neq (2k - 1)\pi/2 \), \( k \in \mathbb{Z} \).

Before using dynamic equations, there exist the assumptions that the quad-rotor is a rigid body and symmetric about axes 0x and 0y, then dynamic equations are mentioned in (3.5) and (3.6).

\[
m \ddot{v} = -mg \overset{\cdot}{E}_z + \overset{\cdot}{F}_{aero} + TR \overset{\cdot}{E}_z
\]  
(3.5)

\[
I_f \ddot{\Omega} = -\overset{\cdot}{\Omega} \times I_f \overset{\cdot}{\Omega} - \overset{\cdot}{G_a} + \overset{\cdot}{T}_{aero} + \tau
\]  
(3.6)

Where \( m \) is the mass of the quad-rotor, \( g \) is the gravity acceleration, \( E_z = (0,0,1)^T \) is the unit vector in the body coordination, \( \Omega \) is the angular velocity in body coordinate, \( I_f \) is the diagonal matrix filled with the moment of inertial along with 0x, 0y and 0z.

In these two dynamic equations, it adds three more terms, \( G_a \), \( F_{aero} \) and \( T_{aero} \), which makes the dynamic model more close to the real situation.
\[
\bar{G}_a = \sum_{i=1}^{4} I_i (\Omega \times \bar{E}_i) (-1)^{i+1} \omega_i
\]

(3.7)

where \(\bar{G}_a\) denotes the gyroscopic torque.

\(\bar{F}_{aero}\) and \(\bar{T}_{aero}\) are the aerodynamic drag [29],

\[
\bar{F}_{aero} = K_a (v - \bar{v}_{air})
\]

(3.8)

\[
\bar{T}_{aero} = K_r (\Omega - \bar{\Omega}_{air})
\]

(3.9)

Where \(K_a\) and \(K_r\) are two unknown aerodynamic matrices, \(\bar{v}_{air}\) and \(\bar{\Omega}_{air}\) are translational velocity and angular velocity of the quad-rotor in world coordinate and body coordinate, respectively.

In the dynamic equations, \(\bar{F}_{aero}\) and \(\bar{T}_{aero}\) are included, which means the disturbances, \(K_a \bar{v}_{air}\) and \(K_r \bar{\Omega}_{air}\), from wind, are added. In this thesis, Gaussian noise is used to replace the disturbances, \(K_a \bar{v}_{air}\) and \(K_r \bar{\Omega}_{air}\).

\[
\begin{bmatrix}
T \\
\tau_1 \\
\tau_2 \\
\tau_3
\end{bmatrix} =
\begin{bmatrix}
f_1 + f_2 + f_3 + f_4 \\
f_1 l - f_3 l \\
f_2 l - f_4 l \\
M_1 + M_3 - M_2 - M_4
\end{bmatrix}
\]

(3.10)

Where \(\tau = [\tau_1, \tau_2, \tau_3]^T\), \(f_i = b \omega_i^2\), \(M_i = k \omega_i^2\), \(i \in [1, 2, 3, 4]\), \(l\) is the distance between the center of the motor and the center of mass of the quad-rotor, \(b\) and \(k\) are positive proportionality constants relative to the density of air, the shape of the blades, the number...
of blades, the chord length of the blades, the pitch angle of the blade airfoil and the drag coefficient [44].

So far, the dynamic model of the classical quad-rotor has been analyzed completely.
Chapter 4

Adaptive controller for the quad-rotor control

In the history of controls, there are three phases, including classic control, modern control, and intelligent control. Classic control mainly focuses on the analysis of stability and the specifications of the system. The methods include Bode diagrams, the root locus, Nyquist criterion, and so forth. These methods are usually used in the frequency domain. After getting a good performance, the transfer function is changed into the time domain. Modern control is applied based on building up the mathematical model of the system and then the specifications of the system are analyzed. Moreover, the observability and controllability are proposed in the modern control. The accuracy of controlling mostly depends on the accuracy of the mathematical model. Intelligent control mainly uses math to simulate the character of living beings trying to obtain the similar capacity, such as swarm, Bayesian network, the neural network and so on. The adaptive controller that the thesis will discuss belongs to modern controls. In what follows, a short description of adaptive control is provided.

Adaptive control means adjusting the behavior of the plant to its changing environment in order to realize ideal performance. Some would say the feedback controller can get the same effect because it is trying to reduce the uncertainties in the system and reach the goal of controlling. The difference lies in
identifying whether a structure is applied for adjusting the parameters in the controller. If this structure exists, then the controller is nonlinear because of the parameter adjustment mechanism, and this type of system is called an adaptive controller. If it does not, then this type of system cannot be regarded as an adaptive controller.

Under this guidance, there exist many adaptive controllers, such as gain scheduling, real-time parameter estimation, deterministic self-tuning regulators, and model reference adaptive control. In this thesis, the model uses adaptive control in order to control the quad-rotor. In this method, there are three key points; one is the Lyapunov stability theory, the second is parameter estimation, and the final point is the reference model.

In order to understand model reference control, knowledge of the Lyapunov stability theory is necessary. Actually, there are four states used to distinguish four types of tendencies, and the most useful one is asymptotically stable. It is used to determine whether a control system is stable or not. When this judgment is applied, it is based on a function called the Lyapunov function. Next, the concept of Lyapunov theory in linear system is used in as an example in order to obtain a better understanding of the theory.

The linear autonomous dynamical system is shown in equation (4.1).

\[ \dot{x} = Ax, \quad x \in \mathbb{R}^n \] (4.1)

\[ V = x^T P x \] (4.2)

Take derivative of V with respect to time, we get,
\[
\dot{V} = x^T P x + x^T P \dot{x} \\
= x^T A^T P x + x^T P A x \\
= -x^T Q x 
\] (4.3)

Where equation (4.2) is the Lyapunov function of the linear autonomous dynamical system, \( P \in \mathbb{R}^{m \times m} \) is symmetric, \( A^T P + PA = -Q \)

The essential idea is to generalize the concept of energy \( V \) for a conservative system in mechanics, where a well-known result states that an equilibrium point is stable if the energy is minimum. Thus, \( V \) is a positive function which has \( \dot{V} \) negative in the neighborhood of a stable equilibrium point.

Actually, there are three Lyapunov's theorems, and in the model reference adaptive control, the first theorem is the only one that is useful. Thus, only the first theorem is listed in the thesis.

Through linear system, the Lyapunov theory is learned in general. Then, how would the Lyapunov's theory be used in the controller? How can it be applied in the nonlinear system?

The basic idea is that Lyapunov function of a system including a linear system or nonlinear system needs to be figured out firstly, and then, if the Lyapunov function satisfies the asymptotical stability, the controller that is obtained from Lyapunov function is qualified. Meanwhile, the updated equations of the parameters can also be obtained from the Lyapunov function. Within an iteration, the parameters are updated and the controller makes the plant reach the desired requirement.
In the control of the quad-rotor, the controller includes two parts, the position controller and the attitude controller. The position controller is applied to track a desired trajectory and to get the corresponding thrust. Through the thrust, the command attitude can be obtained. Next, the command attitude is turned into the desired attitude and then, the attitude controller is used for pursuing the desired attitude. The stability of the trajectory tracking and the attitude tracking including parameter uncertainties and disturbances is guaranteed by the Lyapunov theory. After desired attitude, desired angular velocity and the thrust are obtained, these variables are sent into the motor controller and further, the speed of the four rotors is solved. Equations (4.4)-(4.39) are applied in the paper [44] written by Zuo. The adaptive control block of the quad-rotor is displayed in Figure (4.1).
Use four estimated parameters \( \{ \hat{K}_r, \hat{K}_s, \hat{l}_1, \hat{l}_2 \} \) to represent the four unknown parameters \( \{ K_r, K_s, l_1, l_2 \} \) approximately, and define the errors in equations (4.4), (4.5), (4.6) and (4.7).

\[
\tilde{K}_r = \hat{K}_r - K_r \tag{4.4}
\]

\[
\tilde{K}_s = \hat{K}_s - K_s \tag{4.5}
\]

\[
\tilde{l}_i = l_i - \hat{l}_i \tag{4.6}
\]
\[
\ddot{I}_2 = \dot{I}_2 - \ddot{I}_2
\] (4.7)

Where \( \ddot{I}_1 = (I_{11}, I_{12}, I_{13}) \) and \( \ddot{I}_2 = (I_{21}, I_{22}, I_{23}) \) denotes the upper bounds of the \( K_i \) and \( K_r \).

4.1 The adaptive position controller

Define the position error and the velocity error in equations (4.8) and (4.9),

\[
\dot{Z}_1 = \ddot{p} - \ddot{p}_c
\] (4.8)

\[
\ddot{Z}_2 = \ddot{v} - \ddot{v}_c
\] (4.9)

Where \( \ddot{p}_c \) and \( \ddot{v}_c \) are the desired position and velocity of the quad-rotor. Propose a candidate Lyapunov function in equation (4.10),

\[
V_1 = \frac{1}{2} \dot{Z}_1^T K_1 \dot{Z}_1
\] (4.10)

Where \( K_1 \) is positive definite, which means that \( K_1 \) makes \( \dot{V}_1 \) be a positive function.

\[
\dot{V}_1 = \dot{Z}_1^T K_1 \dot{Z}_1 = \dot{Z}_1^T K_1 (\ddot{v} - \ddot{p}_c)
\] (4.11)

In order to guarantee \( \dot{V}_1 \) negative to keep stability, set up \(- Z_1 = \ddot{v}_c - \ddot{p}_c \), then construct the compound Lyapunov function in equation (4.12),

\[
V_2 = \frac{1}{2} \dot{Z}_1^T K_1 \dot{Z}_1 + \frac{1}{2} \dot{Z}_2^T K_2 \dot{Z}_2 + \frac{1}{2} tr (\dot{Z}_1^T K_1^{-1} \dot{Z}_2) + \frac{1}{2} \ddot{I}_1^T \Gamma_1^{-1} \ddot{I}_1
\] (4.12)

Where \( \Gamma_1^{-1} \) and \( \Gamma_2^{-1} \) are called adaptation gains which is symmetric. Take derivative of \( \dot{V}_2 \) with respect to time, and get it in equation (4.13),
\[
\dot{V}_2 = \tilde{Z}_1^{\top} K_i \tilde{Z}_1 + \tilde{Z}_2^{\top} \hat{Z}_2 + \text{tr} (\tilde{K}_i^{\top} \Gamma_i^{-1} \hat{K}_i) + \hat{l}_i^{\top} \Gamma_2^{-1} \hat{l}_i \\
\leq - \tilde{Z}_1^{\top} K_i \tilde{Z}_1 + \tilde{Z}_2^{\top} \left( K_i \tilde{Z}_1 - gE_\varepsilon \right) + \frac{1}{m} K_i \tilde{v} + u_\varepsilon - \hat{v}_c \\
+ \frac{1}{m} \text{Sign} (\tilde{Z}_2 \hat{l}_1) + \text{tr} (\tilde{K}_i^{\top} \Gamma_i^{-1} \hat{K}_i) + \hat{l}_i^{\top} \Gamma_2^{-1} \hat{l}_i 
\] (4.13)

From equation (4.13), we can pull the position control law in equation (4.14),

\[
\tilde{u}_x = \tilde{v}_c - K_1 \tilde{Z}_1 - K_2 \tilde{Z}_2 - \frac{1}{m} K_i \tilde{v} = - \frac{1}{m} \text{Sign} (\tilde{Z}_2 \hat{l}_1) + g E_\varepsilon 
\] (4.14)

Furthermore,

\[
\dot{\hat{K}}_i = \hat{K}_i = \frac{1}{m} \Gamma_i \tilde{v}^T 
\] (4.16)

\[
\dot{\hat{l}}_1 = \hat{l}_1 = \frac{1}{m} \Gamma_2 \text{Sign} (\tilde{Z}_2 \hat{l}_1) 
\] (4.17)

The reason for constructing position control law and adaption laws as above is to make the derivative of the Lyapunov function be qualified for the stability. \( \tilde{Z}_1, \tilde{Z}_2, \tilde{K}_i \)

and \( \hat{l}_i \) are bounded. Moreover, \( \tilde{Z}_1, \tilde{Z}_2 \to 0 \) as \( t \to \infty \) and \( \hat{K}_i, \hat{l}_i \) do not need to converge
to their real value. Use the balance in force and get the equation (4.18) as follows,
\[ R^T \ddot{u}_f = R^T (\ddot{u} + g \ddot{E}_z) = \frac{1}{m} T \dot{E}_z \] (4.18)

Where \( \ddot{u} = (u_1, u_2, u_3)^T \) and denotes the acceleration in x, y and z in the world frame. By replacing the matrix and vector into equation (4.18), the following equations (4.19), (4.20), (4.21) are obtained.

\[ \theta_c = \arctan(\frac{u_1 c \psi_c + u_2 s \psi_c}{u_3 + g}) \] (4.19)

\[ \phi_c = \arcsin(\frac{u_1 s \psi_c - u_3 c \psi_c}{\sqrt{u_1^2 + u_2^2 + (u_3 + g)^2}}) \] (4.20)

\[ T = m[u_1 (s \theta c \psi c \phi + s \psi s \theta) + u_2 (s \theta s \psi c \phi - c \psi s \phi) + (u_3 + g) c \theta c \phi] \] (4.21)

where \( [\phi_c, \theta_c, \psi_c] \) is the commanded attitude, the desired attitude, \( [\phi, \theta, \psi] \) is the real attitude.

### 4.2 The adaptive attitude controller

Define the attitude error and the angular velocity error in equations (4.22) and (4.23),

\[ \ddot{Z}_3 = \ddot{\Theta} - \ddot{\Theta}_c \] (4.22)

\[ \ddot{Z}_4 = \ddot{\Omega} - \ddot{\Omega}_c \] (4.23)

Where \( \ddot{\Theta}_c = [\phi_c, \theta_c, \psi_c]^T \), \( \ddot{\Omega}_c \) represents the commanded angular velocity, instead of desired angular velocity. Use the Lyapunov candidate function in equation (4.24),
\[
V_3 = \frac{1}{2} \tilde{Z}_3^T \tilde{Z}_3
\]  

(4.24)

Take derivative of \( \tilde{V}_3 \) with respect to time in equation (4.25),

\[
\dot{\tilde{V}}_3 = \tilde{Z}_3^T \tilde{Z}_3 = \tilde{Z}_3^T (W \Omega - \hat{\Theta}_c)
\]  

(4.25)

To insure \( \dot{\tilde{V}}_3 < 0 \), set up equation (4.26),

\[
\dot{\Omega}_d = \Omega^{-1} (\hat{\Theta}_c - \kappa \tilde{Z}_3)
\]  

(4.26)

Where \( \kappa \) is positive definite.

Since \( \hat{\Theta}_c \) is complicated to solve, thus, use a tracking differentiator to obtain an estimator \( \hat{\Theta}_c \) to replace \( \hat{\Theta}_c \). The TD [45] is listed in equation (4.27),

\[
\hat{X}_1 = \tilde{X}_2
\]

\[
\hat{X}_2 = -2 \Lambda \tilde{X}_2 - \Lambda^2 (\tilde{X}_1 - \hat{\Theta}_c)
\]  

(4.27)
Figure (4.2) Commanded attitude tracking based on tracking differentiator

After solving these two equations, we can get \( \ddot{x} \), firstly and then take derivative of \( \ddot{x} \), getting \( \dddot{x} = \dot{\Theta} \). In addition, \( \Lambda = \text{diag} \{a_1, a_2, a_3\} > 0 \) and should be large enough to make \( \dot{\Theta} \) approach \( \dot{\Theta}_c \) quickly. The effect of the tracking differentiator is shown in Figure (4.2). Thus, use \( \dot{\Theta}_c \) to replace \( \dot{\Theta} \) in equation (4.27), we get,

\[
\ddot{\Omega}_d = W^{-1}(\dot{\Theta}_c - K_c \ddot{Z}_c)
\]  (4.28)

If using \( \ddot{\Omega}_d \) directly, then it needs to take derivative of \( \ddot{\Omega}_d \) with respect to time. It is complicated as well. Thus, choose the first-order command filter to track \( \ddot{\Omega}_d \) and the tracking value is represented by \( \ddot{\Omega}_c \), that is, the actual desired angular velocity is indeed
replaced by $\hat{\Omega}_c$. Then, use the attitude controller track $\hat{\Omega}_c$ directly. Equation (4.29) is the first-order command filter for $\hat{\Omega}_c$.

$$\hat{\Omega}_c = -\bar{T} (\hat{\Omega}_c - \hat{\Omega}_d)$$  \hspace{1cm} (4.29)$$

Where $\bar{T} = \text{diag} (t_1, t_2, t_3) > 0$ is necessary large enough to keep fast tracking. And the mechanism is when $\hat{\Omega}_c - \hat{\Omega}_d > 0$, then $\hat{\Omega}_c < 0$, which means $\hat{\Omega}_c$ decreases to $\hat{\Omega}_d$. If $\hat{\Omega}_c - \hat{\Omega}_d < 0$, then $\hat{\Omega}_c > 0$, which means $\hat{\Omega}_c$ increase to $\hat{\Omega}_d$. Therefore, $\hat{\Omega}_c$ always approaches $\hat{\Omega}_d$. Figure (4.3) shows the tendency of the first-order command filter.

![Figure (4.3) Command angular velocity tracking based on the first-order command filter](image-url)
Since $\tilde{\Omega}$ is used as $\tilde{\Omega}_c$, actually there exists the extra error $\hat{\lambda}$ in $\tilde{Z}_3$, where

$$\hat{\lambda} = (\lambda_1, \lambda_2, \lambda_3)^T.$$ 

From equation (4.30), we can get the compensation error $\hat{\lambda}$.

$$\hat{\lambda} = -K_c \tilde{\lambda} + W (\tilde{\Omega} - \tilde{\Omega}_c) = W \tilde{\Omega}_c + K_c (\tilde{Z}_3 - \lambda) - \hat{\lambda} \quad (4.30)$$

And the new error in attitude shows in equation (4.31),

$$\tilde{Z}_3 = \hat{\Theta} - \hat{\Theta}_c - \hat{\lambda} \quad (4.31)$$

Construct Lyapunov-like function in equation (4.32),

$$V_4 = \frac{1}{2} \tilde{Z}_3^T \tilde{Z}_3 + \frac{1}{2} \tilde{Z}_4^T \tilde{Z}_4 + \frac{1}{2} \text{tr} (K_c \Gamma_3^{-1} \tilde{\omega}) + \frac{1}{2} I_c^T \Gamma_4^{-1} I_c \quad (4.32)$$

Take derivative with respect to time, we get equation (4.33)

$$\dot{V}_4 = -\tilde{Z}_3^T K_c \tilde{Z}_3 + \tilde{Z}_3^T \hat{\Theta}_c + \tilde{Z}_4^T W \tilde{Z}_3 - I_c^T (\tilde{\Omega} \times I_c \tilde{\Omega}) - I_c^{-1} \tilde{G}_a + I_c^{-1} K_c \tilde{\Omega}$$

$$+ I_c^T \dot{d}_c + I_c^{-1} \tilde{\tau} - \dot{\hat{\Theta}}_c] + \text{tr} (K_c \Gamma_3^{-1} \tilde{\omega}) + I_c^T \Gamma_4^{-1} I_c \quad (4.33)$$

$$V_4 \leq -\tilde{Z}_3^T K_c \tilde{Z}_3 + \tilde{Z}_3^T \hat{\Theta}_c + \tilde{Z}_4^T [W \tilde{Z}_3 - I_c^T (\tilde{\Omega} \times I_c \tilde{\Omega}) - I_c^{-1} \tilde{G}_a + I_c^{-1} K_c \tilde{\Omega}$$

$$+ \text{Sign} \ (Z_3) I_c^{-1} \tilde{\omega} + I_c^{-1} \tilde{\tau} - \dot{\hat{\Theta}}_c] + \text{tr} (K_c \Gamma_3^{-1} \tilde{\omega}) + I_c^T \Gamma_4^{-1} I_c \quad (4.34)$$

Form equation (4.34), we can pull the attitude controller in equation (4.35)

$$\tilde{\tau} = [\tilde{\Omega} \times I_c \tilde{\Omega}] + \tilde{G}_a - I_c W \tilde{Z}_3 - K_c \tilde{\omega}$$

$$- \text{Sign} \ (Z_3) I_c^T \tilde{\omega} + I_c \tilde{\omega} - I_c K_c \tilde{Z}_4 \quad (4.35)$$
Where $K_3$ and $K_4$ are positive definite. In addition, from equation (4.34), we can pull the attitude adaption law in equations (4.36) and (4.37),

$$
\dot{\hat{K}}_r = K_r = \Gamma_4 I_f^{-1} Z_4 \Omega^T
$$

(4.36)

$$
\dot{\hat{l}}_z = \hat{l}_z = \Gamma_4 I_f^{-1} \text{Sign} (Z_4) \bar{Z}_4
$$

(4.37)

$$
\dot{V}_4 \leq -Z_4^T K_4 Z_4 + \bar{Z}_4^T \dot{Z}_4 - \bar{Z}_4^T K_4 \bar{Z}_4
$$

(4.38)

$$
\leq - \sum_{i=1}^{3} k_{3i} \left| z_{3i} \right| \left| \dot{z}_{3i} \right| - k_{3i} \left| \theta_{ai} \right| - \bar{Z}_4^T K_4 \bar{Z}_4
$$

In equation (4.38), the second line is divided into two parts, and the sign of $\left| z_{3i} \right| - k_{3i} \left| \dot{z}_{3i} \right|$ needs to be discussed. If $\left| z_{3i} \right| > 0$, $Z_4$, $\bar{Z}_4$, $K_4$, $l_2$ are bounded, and $\dot{V}_4 \leq -Z_4^T K_4 Z_4$. By Barbalat lemma, $\lim_{t \to \infty} Z_4 = 0$. If $\left| z_{3i} \right| - k_{3i} \left| \dot{z}_{3i} \right| \leq 0$, $Z_4$ is bounded, and $Z_4$, $\bar{Z}_4$, $K_4$, $l_2$ cannot be guaranteed bounded or not.

Take equation (4.28) and equation (4.31) into equation (4.35), we get,

$$
\ddot{\bar{z}} = \left[ (\Omega \times I_f \Omega) + G_a - I_f T^T \dot{\hat{\theta}}_c + I_f T W^{-1} \dot{\Theta}_c - I_f T W^{-1} K_4 (\Theta - \hat{\Theta}_c) - I_f W^T (\hat{\Theta} - \hat{\Theta}_c - \lambda) - \dot{\hat{K}}_c \hat{\Omega} - \text{Sign} (Z_4) l_2 - I_f K_4 (\Omega - \hat{\Omega}_c) \right]
$$

(4.39)

Next, propose the Lyapunov-like function $V = V_2 + V_4$. Through this function, the whole control system can be proved stable.

### 4.3 Simulation results for adaptive controller
For running the algorithm, all the parameters are given as $m = 0.468 \text{kg}$,

$$g = 9.81 \text{m/s}^2, l = 0.225 \text{ m}, I_x = 4.856 \times 10^{-3} \text{ kg \cdot m}^2, I_y = 4.856 \times 10^{-3} \text{ kg \cdot m}^2,$$

$$I_z = 8.801 \times 10^{-3} \text{ kg \cdot m}^2, I_r = 3.357 \times 10^{-5} \text{ kg \cdot m}^2, b = 2.98 \times 10^{-6} \text{ N \cdot s}^2 / \text{rad}^2,$$

$$k = 1.14 \times 10^{-7} \text{ Nm \cdot s}^2 / \text{rad}^2.$$ And the tuning parameters are given as

$$K_1 = \text{diag} \{0.8, 0.8, 0.8\}, K_2 = \text{diag} \{1.2, 1.2, 1.2\}, K_3 = \text{diag} \{0.25, 0.25, 0.25\},$$

$$K_4 = \text{diag} \{5.5, 5.5\}, \Gamma_1 = \text{diag} \{0.15, 0.15, 0.15\}, \Gamma_2 = \text{diag} \{0.04, 0.04, 0.04\},$$

$$\Gamma_3 = \text{diag} \{0.001, 0.001, 0.001\}, \Gamma_4 = \text{diag} \{10^{-4}, 10^{-4}, 10^{-4}\}, \bar{T} = \text{diag} \{15, 15, 15\},$$

$$K_s = \text{diag} \{0.01, 0.01, 0.01\}, K_r = \text{diag} \{0.001, 0.001, 0.001\} \text{ and } \Lambda = 2\bar{T}.$$ The initial position is $\vec{p}_0 = [5, 0, 0]^T$ and the desired trajectory is $\vec{p}_d = (5 \cos(0.1t), 5 \sin(0.1t), 0.5)^T$. The initial values of attitude, angular velocity, velocity and estimated parameters are zero.

Figure (4.4) Desired-path tracking based on adaptive controller
Figure (4.5) Thrust generated from four rotors during the flight

Figure (4.6) Desired attitude generated from the desired acceleration
Figure (4.7) Desired-attitude tracking based on adaptive controller

Figure (4.8) Torque generated from four rotors during the flight
Figure (4.9) Estimated aerodynamic force coefficient based on adaptive law

Figure (4.10) Estimated wind disturbance upper bound for Kt
Figure (4.11) Estimated aerodynamic moment coefficient based on adaptive law

Figure (4.12) Estimated wind disturbance upper bound for Kr
Chapter 5

PD controller for the quad-rotor control

For PD controller, it also includes two loops, outer loop and inner loop. Outer loop is used as position controller and inner loop is the attitude controller. In this controller, we use the same dynamic model as that in adaptive controller.

5.1 The PD controller

The block for PD controller is showing in Figure (5.1),

Figure (5.1) The PD controller block

The specific steps are introduced below:
\[ \ddot{p}_r = \dot{p}_d - \dot{p}_r, \quad (5.1) \]

Where \( \ddot{p}_r = (\text{error}_x, \text{error}_y, \text{error}_z)^T \), \( \dot{p}_d = (x_d, y_d, z_d)^T \), \( \dot{p}_r = (x, y, z)^T \) is the desired position vector.

\[ \ddot{p}_r + K_d \dot{p}_r + K_p p_r = 0 \quad (5.2) \]

Where \( K_d \) and \( K_p \) are two positive definite matrices. And according to the Routh-Hurwitz criterion, the position error \( p_r \) converges to zero exponentially [44].

\[ \dddot{p}_r = \dddot{p}_d + K_d (\dddot{p}_d - \dddot{p}_r) + K_p (\dddot{p}_d - \dddot{p}_r) \quad (5.3) \]

Define the controls as \( \ddot{U} = \dddot{p}_r = (U_1, U_2, U_3)^T \). Since the acceleration of UAV is small, \( \dddot{p}_d \) is ignored, the new equation is obtained as,

\[ \ddot{U} = K_d (\dddot{p}_d - \dddot{p}_r) + K_p (\dddot{p}_d - \dddot{p}_r) \quad (5.4) \]

After getting \( \ddot{U} \), the equations are pulled out as follows,

\[ \theta_c = \arctan\left( \frac{U_1 c \psi_c + U_2 s \psi_c}{U_3 + g} \right) \quad (5.5) \]

\[ \phi_c = \arcsin\left( \frac{U_1 s \psi_c - U_2 c \psi_c}{\sqrt{U_1^2 + U_2^2 + (U_3 + g)^2}} \right) \quad (5.6) \]
Where $\psi$ is 0. Equations (5.1)-(5.3), (5.5), (5.6) are applied in the paper [46] written by Zuo. Equations (5.7)-(5.13) are applied in the paper [14] written by Kumar.

Then, we can get three terms, $\Delta \omega_\phi$, $\Delta \omega_\theta$, $\Delta \omega_\psi$, based on the inner loop PD controller, we can get,

$$\Delta \omega_\phi = k_{p,\phi} (\phi^\text{des} - \phi) + k_{d,\phi} (p^\text{des} - p) \quad (5.7)$$

$$\Delta \omega_\theta = k_{p,\theta} (\theta^\text{des} - \theta) + k_{d,\theta} (q^\text{des} - q) \quad (5.8)$$

$$\Delta \omega_\psi = k_{p,\psi} (\psi^\text{des} - \psi) + k_{d,\psi} (r^\text{des} - r) \quad (5.9)$$

Where $\dot{\phi} \approx p$, $\dot{\theta} \approx q$, $\dot{\psi} \approx r$, since the acceleration is small and the attitude is small.

When UAV is hovering in the air, the gravity of UAV is equal to the thrust from four rotors, and the equation is expressed as,

$$F_i = k \omega_i^2 = \frac{mg}{4} \quad (5.10)$$

Change equation (5.10) into equation (5.11),

$$\omega_i = \omega_h = \sqrt{\frac{mg}{4k}} \quad (5.11)$$

In addition, another term is listed as equation (5.12),

$$\Delta \omega_F = \frac{m}{8k \omega_h} U_j \quad (5.12)$$
From the equations (5.7), (5.8), (5.9), (5.11) and (5.12) above, the desired rotor speeds are obtained as,

\[
\begin{bmatrix}
\omega_{1 \text{des}} \\
\omega_{2 \text{des}} \\
\omega_{3 \text{des}} \\
\omega_{4 \text{des}}
\end{bmatrix} = \begin{bmatrix}
1 & 0 & -1 & 1
\end{bmatrix}
\begin{bmatrix}
\omega_{\phi} + \Delta \omega_{\phi} \\
\omega_{\theta} \\
\omega_{\gamma} \\
\omega_{\psi}
\end{bmatrix}
\]

Take the desired rotor speeds into equation (3.10), the thrust and torque of UAV can be obtained.

### 5.2 Simulation results for PD controller

All the parameters for describing the quad-rotor are same with those in the adaptive controller. And the parameters of \((k_{px}, k_{dx}), (k_{py}, k_{dy})\) and \((k_{pz}, k_{dz})\) are \((0.2, 0.8), (0.2, 0.8)\) and \((0.2, 0.8)\) for x, y and z, respectively. The parameters of \((k_{p, \text{pitch}}, k_{d, \text{pitch}}), (k_{p, \text{roll}}, k_{d, \text{roll}})\) and \((k_{p, \text{jaw}}, k_{d, \text{jaw}})\) are \((-100, 1), (90, 1)\) and \((10, 1)\) for pitch, roll and jaw, respectively.
Figure (5.2) Desired-path tracking based on PD controller

Figure (5.3) Thrust generated from four rotors during the flight
Figure (5.4) Desired attitude generated from the desired acceleration

Figure (5.5) Desired-attitude tracking based on PD controller
Figure (5.6) Torque generated from four rotors during the flight
Chapter 6

RBF Neural Network PD controller design

A Radial Basis Function (RBF) neural network PD controller (or simply a neural PD controller) is a method that combines a PD controller and an RBF neural network together. Next, PD position controllers and RBF neural networks will be introduced in general.

A PID controller is well known around the world. In this type of controller, there are three parts, including proportion, integration, and differentiation. Based on the specific requirements, a PID controller can be changed into two more types, such as PI controller and PD controller. Different parts have different functions in the control process. The principle of a PID controller is getting feedback about actual output and removing the error between the actual output and preference input by giving the desired input to the plant, which is a process of controlling. The reason why PID controllers are widely used is that they have some advantages. Firstly, this method has existed for a long time and become a very mature technology. Secondly, this method is easy to master compared to other controllers because a PID controller does not need to build up a mathematical model for the objective. Finally, the effect of a PID controller is usually qualified and has some robustness.
Before discussing RBF, the neural network should be introduced. A neural network is actually an intelligent method that simulates the nervous systems in our bodies. In 1943, Warren S. McCulloch and Walter H. Pitts used math to conduct research about how neural networks describe objective events, which started the research into the neural network. In 1957, Frank Rosenblatt proposed and created the Perceptron, which turned theory into practice, and afterward, neural network became a very popular field. So far, the neural network has had many models that can be used for different applications, such as MP models, BP models, and RBF models. The working process of neural network has two steps. The first step is that the calculation remains unchanged, and the weighting factors are modified through learning. The second step is that the weighting factors are fixed and the calculation units change to balance the system. Furthermore, the way of learning can be divided into three types: supervised learning, unsupervised learning, and reinforced learning. Based on these ways of learning, neural networks are capable of being adaptive and generalized, reflecting nonlinear objectives, and being highly parallel. It is worth pointing out that neural networks have excellent nonlinear capabilities, with which the fuzzy logic algorithm, genetic algorithm, and evolutionary mechanism are able to be better applied in pattern recognition, speech recognition, and intelligent control. In order to understand neural networks well, an artificial neuron model is necessary. Figure (6.1) shows the artificial neuron model.
Figure (6.1) Artificial Neuron Model

In this model, the working principle of the neural network is shown clearly. Firstly, some inputs, $x_1, x_2, \ldots, x_{n-1}, x_n$, are sent into the network, and then we get $U_n$ by weighting every input and summing all the terms. Secondly, $U_n$ is sent to function $f$, and the result of function $f$ turns out to be the $y_{out}$, where function $f$ is called the activation function. The activation function can be a jump function, a piece wise linear function, a sigmoid function, or some other functions.

In this thesis, an RBF PD controller is proposed in order to control the position of a UAV. This controller is constructed by two parts, one is the PD controller, and the other is RBF (radial basis function) neural network identification. In 1985, RBF was proposed by Powell. Three years later, RBF was applied in a neural network for the first time. Compared to a BP neural network, an RBF neural network performs better in approaching, classifying, and quickly learning. Thus, it is suitable for use in the identification of nonlinear system and real-time control. For this reason, an RBF PD controller was chosen to control the UAV, whose mathematical model is nonlinear and
for which real-time control is needed. For the RBF neural network, the hidden layer is formed by radial basis function. And the input vector is sent to the hidden layer directly, which replaces the process of weighting. Then the output of the hidden layer is weighted and summed. Finally, the output of the RBF neural network is obtained. Equation (6.1) is the radial basis function of the RBF neural network.

$$f_j(x) = \exp \left( -\frac{1}{2b_j^2} \left\| \tilde{x} - \tilde{c}_j \right\|^2 \right), \quad j = 1, 2, \ldots, m \quad (6.1)$$

Where,

$\tilde{x}$ The input vector;

$\tilde{c}_j = [c_{j1}, c_{j2}, \ldots, c_{jm}]^T$, $j = 1, 2, \ldots, m$ The centers of RBF;

$\tilde{b} = [b_1, b_2, \ldots, b_m]^T$ The basis vector;

### 6.1 The structure of the neural PD position controller

The control block for the RBF Neural Network PD position controller is shown in Figure (6.2).
In the figure (6.2), we can find the desired position, and the output, real position of the UAV. Both positions include x, y, z, three directions, respectively. The output of the PD controller, u, is the accelerations of the UAV in three directions. Besides, RBF Neural Network works as the identification and offers certain information for changing the parameters $k_p$ and $k_d$ in PD controller. Figure (6.3) shows the structure of the RBF Neural Network in detail.
In Figure (6.3), there are three layers, the input layer, the hidden layer and the output layer. For the input layer, there are nine input variables. And there exist six and three variables in hidden layer and output layer respectively. The three variables in output layer represent middle x (xm), middle y (ym) and middle z (zm) respectively, which are used for tracking real x, real y and real z (real p). In the hidden layer, the equation (6.1) is put in each circle. For the first three input variables in the input layer, they represent desired acceleration in x direction, the current iteration real x and the previous iteration real x, respectively. Then, for the following two sets, it is same to the first set, standing for y and z, respectively.

6.2 The neural PD position controller
In order to express clearly, the equations are introduced as follows:

\[
\ddot{p}_e = p_d - p_e \tag{6.2}
\]

Where \( p_e = (\text{error}_x, \text{error}_y, \text{error}_z)^T \), \( p_d = (x_d, y_d, z_d)^T \), and \( p_r = (x, y, z)^T \) is the desired position vector.

\[
\dddot{p}_e + K_d \ddot{p}_e + K_p p_e = 0 \tag{6.3}
\]

Where \( K_d \) and \( K_p \) are two positive definite matrices.

\[
\dddot{p}_r = \dddot{p}_d + K_d (\ddot{p}_d - \ddot{p}_r) + K_p (\dot{p}_d - \dot{p}_r) \tag{6.4}
\]

Define the controls as \( \ddot{U} = \dddot{p}_e = (U_1, U_2, U_3)^T \). Since the acceleration of UAV is small, \( \dddot{p}_d \) is ignored, the new equation is obtained as,

\[
\ddot{U} = K_d (\ddot{p}_d - \ddot{p}_r) + K_p (\dot{p}_d - \dot{p}_r) \tag{6.5}
\]

In order to express the process conveniently, the PD controller will be converted to digital PD model. Turn equation (6.5) into equation (6.6),

\[
\begin{align*}
U_1 &= k_{d1} (x_d - x) \\
U_2 &= k_{d2} (y_d - y) \\
U_3 &= k_3 (z_d - z)
\end{align*}
\tag{6.6}
\]

Next, converting the above equation into digital form, showing as equations (6.7), (6.8) and (6.9).
\[ U_1(k) = k_{d1} \times (error_x(k) - error_x(k - 1)) + k_{p1} \times error_x(k) \quad (6.7) \]

\[ U_2(k) = k_{d2} \times (error_y(k) - error_y(k - 1)) + k_{p2} \times error_y(k) \quad (6.8) \]

\[ U_3(k) = k_{d3} \times (error_z(k) - error_z(k - 1)) + k_{p3} \times error_z(k) \quad (6.9) \]

By this way, there is one advantage that feedback on only the position information compared to the adaptive control, which can reduce computational cost for the velocity of the UAV. Equations (6.10)-(6.19) are mainly from a book [47] written by Liu. Two sets of cost functions are defined,

\[ J_{1x} = 0.5 \times (x - x_m)^2 \]

\[ J_{1y} = 0.5 \times (y - y_m)^2 \]

\[ J_{1z} = 0.5 \times (z - z_m)^2 \quad (6.10) \]

\[ J_{2x} = 0.5 \times error_x(k)^2 \]

\[ J_{2y} = 0.5 \times error_y(k)^2 \]

\[ J_{2z} = 0.5 \times error_z(k)^2 \quad (6.11) \]

Where \( x_m, y_m \) and \( z_m \) represent the outputs of RBF neural network, \( k \) is the iteration number. The weighting factors between hidden layer and output layer in the neural network are defined as,

\[ W_s = [w_{s1}, w_{s2}, \ldots, w_{s6}] \]
\[
W_y = [w_{y1}, w_{y2}, \ldots, w_{y6}]
\]

\[
W_z = [w_{z1}, w_{z2}, \ldots, w_{z6}]
\]

(6.12)

The output of the neural network is,

\[
\begin{bmatrix}
  x_m \\
  y_m \\
  z_m
\end{bmatrix} = \begin{bmatrix}
  W_x \\
  W_y [f_1, f_2, \ldots, f_6]^T
\end{bmatrix}
\]

(6.13)

Where \( f_j = f_j(x), j = 1, 2, \ldots, 6 \)

Take the controls in x direction as an example, and use the gradient descent based on the cost function \( J_{1x} \), we can get equations (6.14), (6.15) and (6.16),

\[
w_{yj}(k) = w_{yj}(k-1) + s \times (x(k) - x_m(k)) \times f_j(x) + r(w_{yj}(k-1) - w_{yj}(k-2))
\]

(6.14)

\[
b_{yj}(k) = b_{yj}(k-1) + s \times (x(k) - x_m(k)) w_{yj}(k-1) f_j(x) \frac{\|x - c_{yi}\|}{b_{yi}(k-1)} + r(b_{yj}(k-1) - b_{yj}(k-2))
\]

(6.15)

\[
c_{yi}(k) = c_{yi}(k-1) + s \times (x(k) - x_m(k)) w_{yj}(k-1) f_j(x) \frac{x_{1j} - c_{yi}}{b_{yi}(k-1)} + r(c_{yi}(k-1) - c_{yi}(k-2))
\]

(6.16)

Where \( s \) is the speed of learning and \( r \) is the momentum factor. \( w_{yi} \) is the weighting factor for x direction. \( b_{yi} \) is the basis factor for x direction. \( c_{yi} \) is the center of the radial basis function obtained by the error in x direction, \( j \in [1, 2, \ldots, 6] \), \( i \in [1, 2, \ldots, 9] \). By the same way, \( w_{yi}, w_{zi}, b_{yi}, b_{zi}, c_{yi}, c_{zi} \) can be obtained. Through
this way, it makes $x_m$ tracks $x$, which leads to $p_m$ tracking $p_r$. Then, the cost function $J_{2x}$ is used to update the parameters $k_{p1}$ and $k_{d1}$ in the PD controller,

$$k_{d1}(k) = k_{d1}(k-1) - s \times \frac{\partial J_{2x}}{\partial k_{d1}}$$

$$= k_{d1}(k-1) - s \times \frac{\partial J_{2x}}{\partial x} \times \frac{\partial x}{\partial U_1} \times \frac{\partial U_1}{\partial k_{d1}}$$

$$= k_{d1}(k-1) + s \times \text{error}_{x} (k) \times \frac{\partial x}{\partial U_1} \times \text{error}_{x} (k - 1)$$

(6.17)

$$k_{p1}(k) = k_{p1}(k-1) - s \times \frac{\partial J_{2x}}{\partial k_{p1}}$$

$$= k_{p1}(k-1) - s \times \frac{\partial J_{2x}}{\partial x} \times \frac{\partial x}{\partial U_1} \times \frac{\partial U_1}{\partial k_{p1}}$$

$$= k_{p1}(k-1) + s \times \text{error}_{x} (k) \times \frac{\partial x}{\partial U_1} \times \text{error}_{x} (k)$$

(6.18)

$$\frac{\partial x}{\partial U_1} \approx \frac{\partial x_m}{\partial U_1} = \sum_{j=1}^{6} w_{j} \times f_{j}(x) \times (c_{j1} - U_{1})$$

(6.19)

Take equation (6.19) into equations (6.17) and (6.18), $k_{p1}$ and $k_{d1}$ can be updated smoothly. And this is the reason why $p_m$ needs to track $p_r$. In addition, equations (6.20) and (6.21) are used for $y$ and $z$ directions similar with equation (6.19).

$$\frac{\partial y}{\partial U_2} \approx \frac{\partial y_m}{\partial U_2} = \sum_{j=1}^{6} w_{j} \times f_{j}(x) \times (c_{j4} - U_{2})$$

(6.20)

$$\frac{\partial z}{\partial U_3} \approx \frac{\partial z_m}{\partial U_3} = \sum_{j=1}^{6} w_{j} \times f_{j}(x) \times (c_{j7} - U_{3})$$

(6.21)

For the inner loop PD controller, the attitude controller, it is same with that in Chapter 6.
6.3 Simulation results for neural PD controller

All the parameters for describing the quad-rotor are same with those in the adaptive controller. And the initial values of \((k_{px}, k_{dx})\), \((k_{py}, k_{dy})\) and \((k_{pz}, k_{dz})\) are same with those in PD controller. The parameters of \((k_{p,pitch}, k_{d,pitch})\), \((k_{p,roll}, k_{d,roll})\) and \((k_{p,yaw}, k_{d,yaw})\) are same with those in PD controller as well. The initial values of \((w_x, w_y, w_z)\), \((c_x, c_y, c_z)\) and \((b_x, b_y, b_z)\) are \((1.3, 0.001, 20)\), \((10, 10, 10)\) and \((30, 30, 30)\) respectively. The learning factor \(s\) is 1. The momentum factor is 0.2. The parameters for updating the \((k_{px}, k_{dx})\), \((k_{py}, k_{dy})\) and \((k_{pz}, k_{dz})\) are \((40, 10)\), \((90, 60)\) and \((90, 60)\) respectively.

![Desired-path tracking based on neural PD controller](image)

Figure (6.4) Desired-path tracking based on neural PD controller
Figure (6.5) Thrust generated from four rotors during the flight

Figure (6.6) Desired attitude generated from the desired acceleration
Figure (6.7) Real-attitude tracking based on neural PD controller

Figure (6.8) Torque generated from four rotors during the flight
Figure (6.9) Updated kp and kd in x direction based on neural network

Figure (6.10) Updated kp and kd in y direction based on neural network
Figure (6.11) Updated kp and kd in z direction based on neural network
Chapter 7

Comparison between traditional PD controller and neural PD controller under wind disturbance

First, the comparison between the performance of a neural PD controller and a traditional PD controller under constant wind disturbance is presented. Here, during different simulations, the magnitude of wind velocity is changed from 1m/s to 20m/s, and the direction of the wind is fixed in the positive x, y and z axes. In Figure (7.1), the wind disturbance is added in positive x, y, and z directions at the same time, and for each simulation, the magnitudes of wind velocity are the same in three directions. Similarly, in Figure (7.2), the rotational wind disturbance is added around the x, y, and z directions by the same way of adding the translational wind disturbance.
Figure (7.1) Root mean square error of the position under translational wind disturbance

Figure (7.2) Root mean square error of the position under rotational wind disturbance
In Figure (7.1), for the plot of RMSE-x, the RMSE of neural PD controller in x direction is as same as that of PD controller except the interval between 12 and 18. In the interval [12, 18], the RMSE of neural PD controller in x direction is a little larger than that of PD controller under constant translational wind disturbance, which means the accuracy of neural PD controller is lower than that of PD controller during that interval. However, for the plots of RMSE-y and RMSE-z, the RMSE of neural PD controller in y and z directions are smaller than that of PD controller in the interval [0, 60] under constant translational wind disturbance. In another way, the accuracy of neural PD controller is improved compared to that of PD controller obviously in y and z directions under constant translational wind disturbance. Since there is only a very small interval in the plot of RMSE-x where the RMSE of neural PD controller is larger than that of PD controller, thus, a conclusion is pulled out as the accuracy of neural PD controller is improved compared to that of PD controller under constant translational wind disturbance in total.

In Figure (7.2), for the plot of RMSE-x, the RMSE of neural PD controller in x direction is similar with that of PD controller under constant rotational wind disturbance. For the plots of RMSE-y and RMSE-z, the RMSEs of neural PD controller in y and z directions are smaller than those of PD controller under constant rotational wind disturbance except during certain small intervals as [1, 4] and [2, 9]. Thus, the accuracy of neural PD controller under constant rotational wind disturbance is improved compared to that of PD controller, but not pronounced.
Next, the errors, another type of measurement for control accuracy, between the desired path and actual path under random translational and rotational wind disturbances are simulated based on neural PD controller and PD controller, which is more practical for simulating the environment. The results are showed in Figures (7.3) and (7.4).
Figure (7.3) Error between desired and actual path under random translational wind disturbance

68
Random Rotational Wind Disturbance

Time(s) vs. Error Plot

(c)

Random Rotational Wind Disturbance

Time(s) vs. Error Plot

(d)
In Figure (7.3), the random translational wind disturbance are added to x, y and z directions. The directions of random translational wind disturbance are fixed to the positive x, y and z directions and the magnitudes of the translational wind disturbance are random, which are sampled from uniform distribution. For an iteration in one simulation, the magnitudes of the wind velocity are generated from the uniform distribution and are the same in x, y and z directions. In Figure (7.3), five subfigures (a, b, c, d, e) are listed. The difference among them is the interval of the random translational wind velocity that is added to the quad-rotor. The intervals of these five subfigures change from [0, 1] to [0, 5]. And for each subfigure, the neural PD controller and PD controller are under the same random translational wind disturbance. The green plots in the figures (a)-(e) present the
magnitude of the random translational wind disturbance during the flight. From the five subfigures (a)-(e) in Figure (7.3), the errors between the desired path and actual path are similar for neural PD controller and PD controller in x direction. The errors between the desired path and actual path of neural PD controller in y and z directions under random translational wind disturbance are obviously improved compared to those of PD controller.

In Figure (7.4), the random rotational wind disturbance is added around x, y and z directions by the same way of adding random translational wind disturbance in Figure (7.3). The same conclusions are obtained as in Figure (7.3).

Thus, the position-track accuracy of neural PD controller under wind disturbances are higher than that of PD controller; that is, neural PD controller has a better wind disturbances rejection in position control.
Chapter 8

Conclusions and future work

8.1 Conclusions

This thesis is focused on exploring a new control method that can improve the performance of quad-rotors under wind disturbance. This thesis mainly includes five parts, a dynamic model, an adaptive controller, a PD controller, a neural PD controller, and the comparison between the PD controller and the neural PD controller under wind disturbance.

The dynamic model offers the model of the dynamics, or equations of motions, of the quad-rotor. Actually, for any control problem, obtaining the dynamic model of the object is indispensable and a very important task. A mathematical model was developed in order to describe and analyze the UAV motion, which can help make UAVs move as desired.

These adaptive control and traditional PD control techniques have already been used to control the quad-rotor. These two controllers show the method works as well as the difference between these two controllers. They present two different theories of stability, and based on the PD controller, the neural PD controller is derived for controlling the quad-rotor. The neural PD controller actually changes the structure of the
control law (as an adaptive controller) as the error changes between the desired path and the actual path. Inside the neural network, there are two cost functions. The gradients of the cost functions with respect to different parameters are used in order to minimize the error, and as the parameters are updated, the structure of the neural network is changed. This is a process of learning to achieve better performance of the quad-rotor.

The performances in position tracking of the PD controller and the neural PD controller are compared under wind disturbance. Extensive simulation studies were carried out that included both constant as well as random disturbances added to the system. Different intensities of disturbances were added for different case studies for both constant and random disturbance models. It was found that the neural PD controller performed better in terms of positional error with respect to the traditional PD controller for the constant and random disturbance cases.

8.2 Future work

Future work would include exploration of some other neural network models to see if the performance can be improved more. At the same time, the structure of the control block also has some space for modification. In addition, establishing an accurate wind disturbance model is necessary for simulation because this can help the simulation be more realistic. Another future work is implementation of the proposed technique on a real-world quad-rotor.

The quad-rotor includes frames, actuators, a communication system, hardware computers, and an intelligent algorithm. The intelligent algorithm has two layers. One is the decision layer, and the other is the control layer (including the dynamic model).
level of intelligence, such as avoiding obstacles, cooperation among a group of quad-rotors, or completing tasks independently, also mainly depends on the decision layer.

Future work would include developing the decision layer that can learn from control layer to improve the performance of a quad-rotor and its ability to navigate in not only environment with obstacles but also other possibilities such as high winds or rotor/sensor failure.
References


T. Dierks and S. Jagannathan, “Output Feedback Control of a Quadrotor UAV


Appendix A

Codes for neural PD controller and PD controller

```matlab
x2=zeros(20,1);
y2=zeros(20,1);
z2=zeros(20,1);
v_air=0;
w_air=0;
for sx=1:1:5
    tr=0.40;
    ro=0;
    v_air=v_air+1;
    w_air=w_air+1;
    v_air=sx*rand;
    w_air=sx*rand;
    m=0.468;
g=9.81;
Ir=3.357*10^(-5);Ix=4.856*(10^(-3));Iy=4.856*(10^(-3));Iz=8.801*(10^(-3));
If=[Ix 0 0;0 Iy 0;0 0 Iz];
b=2.98*10^(-6);c=1.14*10^(-7);
l=0.225;
ez=[0 0 1]';

xite=1;
alfa=0.2;
belte=0.53;

x=[0 0 0]';
y=[0 0 0]';
z=[0 0 0]';
cix=10*ones(3,6);
bix=30*ones(6,1);
wx=1.3*ones(6,1);

ciy=10*ones(3,6);
biy=30*ones(6,1);
wy=0.0001*ones(6,1);

ciz=10*ones(3,6);
biz=30*ones(6,1);
wz=20*ones(6,1);

hx=[0 0 0 0 0 0]';
hy=[0 0 0 0 0 0]';
hz=[0 0 0 0 0 0]';
cix_1=cix; cix_2=cix_1;cix_3=cix_2;
```

bix_1=bix; bix_2=bix_1; bix_3=bix_2;
wx_1=wx; wx_2=wx_1; wx_3=wx_2;

ciy_1=ciy; ciy_2=ciy_1; ciy_3=ciy_2;
biy_1=biy; biy_2=biy_1; biy_3=biy_2;
wy_1=wy; wy_2=wy_1; wy_3=wy_2;

ciz_1=ciz; ciz_2=ciz_1; ciz_3=ciz_2;
biz_1=biz; biz_2=biz_1; biz_3=biz_2;
wz_1=wz; wz_2=wz_1; wz_3=wz_2;

ux_1=0; uy_1=0;
xc_x=[0 0 0]';
errorx_1=0; errorx_2=0; errorx=0;

uy_1=0; uy_1=0;
xc_y=[0 0 0]';
errory_1=0; errory_2=0; errory=0;

uz_1=0; uz_1=0;
xc_z=[0 0 0]';
errorz_1=0; errorz_2=0; errorz=0;

% kp=0.03; kdx=0.03;
% kpx_1=kp;
% kdx_1=kd;
% kpy_1=kp;
% kdy_1=kd;
% kpz_1=kp;
% kdz_1=kd;

kpx_1=0.2;
kd_1=0.8;
% kpx_1=kp;
% kdx_1=kd;
% kpy_1=kp;
% kdy_1=kd;
% kpz_1=kp;
% kdz_1=kd;

kpx_1=0.2;
kdx_1=0.8;
kpy_1=0.2;
kdy_1=0.8;
kpz_1=0.2;
kdz_1=0.8;
% kpx_1=0.015;
% kdx_1=0.12;
% kpy_1=0.015;
% kdy_1=0.12;
% kpz_1=0;
% kdz_1=0;

xitekpx=40;
xitekd=10;

xitekpy=90;
xitekdy=60;

xitekpz=90;
xitekdz=60;

% xitekpx=0;
% xitekdx=0;
%
% xitekpy=0;
% xitekdy=0;
%
% xitekpz=0;
% xitekdz=0;


ts=0.1;
t=0;
s=0;
k=0;

ymoutx=zeros(7000,1);
dyoutx=zeros(7000,1);
d_cix=zeros(3,6);
kpx=zeros(700,1);
kdxx=zeros(700,1);
youtx=zeros(7000,1);
dwx=zeros(6,1);

ymouty=zeros(7000,1);
dyouty=zeros(7000,1);
d_ciy=zeros(3,6);
kpy=zeros(700,1);
kdyy=zeros(700,1);
youty=zeros(7000,1);
dwy=zeros(6,1);

ymoutz=zeros(7000,1);
dyoutz=zeros(7000,1);
d_ciz=zeros(3,6);
kpz=zeros(700,1);
kdzz=zeros(700,1);
youtz=zeros(7000,1);
dwz=zeros(6,1);

error=zeros(3,7000);
v=zeros(3,7000);
u=zeros(3,7000);
yout=zeros(3,7000);
rin=zeros(3,7000);
vc=zeros(3,7000);
ac=zeros(3,7000);
e_1=zeros(3,1);
e_2=zeros(3,1);
v_1=zeros(3,1);
y_1=[5 0 0]';
error_1=zeros(3,1);
error_2=zeros(3,1);

85
u_1=zeros(3,1);
k_t=0.01;
UA=zeros(3,7000);
UA_1=zeros(3,7000);
fi_c=zeros(7000,1);
theta_c=zeros(7000,1);
y_c=zeros(7000,1);
angle_c=zeros(3,7000);
T=zeros(7000,1);
angle=zeros(3,7000);
angular_v_c=zeros(3,7000);
dw_fi=zeros(7000,1);
dw_theta=zeros(7000,1);
dw_fy=zeros(7000,1);
wh=0;
dw_f=zeros(7000,1);
w_rotor=zeros(4,7000);
angular_v_a=zeros(3,7000);
angular_v=zeros(3,7000);
angular_v_1=zeros(3,7000);
kr=eye(3)*0.001;

% aa=0.6;
% bb=0.02;
kp_fi=-100;
kd_fi=1;
kp_theta=90;
kd_theta=1;
kp_fy=10;
kd_fy=1;
A=[1 0 -1;1 1 0 -1;1 0 1 1 -1 0 -1];
dt2=2;
xxx=zeros(3,7000);
fuhao1=zeros(3,1);
fuhao2=zeros(3,1);
signw11=0; signw12=0; signw13=0; signw21=0; signw22=0; signw23=0;
sign11=0; sign12=0; sign13=0; sign21=0; sign22=0; sign23=0;
errorp=zeros(3,7000);
errornp=zeros(3,7000);
for i=1:1:600
t=t+0.1;
s=s+1;
rin(:,s)=[5*cos(0.1*t) 5*sin(0.1*t) 0.5]';
vc(:,s)=[-0.5*sin(0.1*t) 0.5*cos(0.1*t) 0]';
ac(:,s)=[-0.05*cos(0.1*t) -0.05*sin(0.1*t) 0]';
k=0;
for as=1:1
    k=k+1;
    h=zeros(6,1);
    for j=1:1:6
        hx(j)=exp(-(norm(x-ci_1(:,j)))^2/(2*bix_1(j)*bix_1(j)));
        hy(j)=exp(-(norm(y-ciy_1(:,j)))^2/(2*biy_1(j)*biy_1(j)));
        hz(j)=exp(-(norm(z-ciz_1(:,j)))^2/(2*biz_1(j)*biz_1(j)));
    end
    ymoutx(k)=wx_1'*hx;
    ymouty(k)=wy_1'*hy;
    ymoutz(k)=wz_1'*hz;
\[d_{\text{wx}} = 0 \times \text{wx};\]
\[d_{\text{wy}} = 0 \times \text{wy};\]
\[d_{\text{wz}} = 0 \times \text{wz};\]

\textbf{for} \ j = 1:1:6
\[
\begin{align*}
  d_{\text{wx}}(j) &= \text{xite} \times (\text{youtx}(k) - \text{ymoutx}(k)) \times h_x(j); \\
  d_{\text{wy}}(j) &= \text{xite} \times (\text{youty}(k) - \text{ymouty}(k)) \times h_y(j); \\
  d_{\text{wz}}(j) &= \text{xite} \times (\text{youtz}(k) - \text{ymoutz}(k)) \times h_z(j); \\
\end{align*}
\]
\textbf{end}

\[\text{wx} = \text{wx}_1 + d_{\text{wx}} + \alpha \times (\text{wx}_1 - \text{wx}_2) + \beta \times (\text{wx}_2 - \text{wx}_3);\]
\[\text{wy} = \text{wy}_1 + d_{\text{wy}} + \alpha \times (\text{wy}_1 - \text{wy}_2) + \beta \times (\text{wy}_2 - \text{wy}_3);\]
\[\text{wz} = \text{wz}_1 + d_{\text{wz}} + \alpha \times (\text{wz}_1 - \text{wz}_2) + \beta \times (\text{wz}_2 - \text{wz}_3);\]

\[d_{\text{bix}} = 0 \times \text{bix};\]
\[d_{\text{biy}} = 0 \times \text{biy};\]
\[d_{\text{biz}} = 0 \times \text{biz};\]

\textbf{for} \ j = 1:1:6
\[
\begin{align*}
  d_{\text{bix}}(j) &= \text{xite} \times (\text{youtx}(k) - \text{ymoutx}(k)) \times \text{wx}_1(j) \times h_x(j) \times (\text{bix}_1(j)^{-3}) \times \text{norm}(x - \text{cix}_1(:,j))^2; \\
  d_{\text{biy}}(j) &= \text{xite} \times (\text{youty}(k) - \text{ymouty}(k)) \times \text{wy}_1(j) \times h_y(j) \times (\text{biy}_1(j)^{-3}) \times \text{norm}(y - \text{ciy}_1(:,j))^2; \\
  d_{\text{biz}}(j) &= \text{xite} \times (\text{youtz}(k) - \text{ymoutz}(k)) \times \text{wz}_1(j) \times h_z(j) \times (\text{biz}_1(j)^{-3}) \times \text{norm}(z - \text{ciz}_1(:,j))^2; \\
\end{align*}
\]
\textbf{end}

\textbf{for} \ j = 1:1:6
\textbf{for} \ i = 1:1:3
\[
\begin{align*}
  d_{\text{cix}}(i,j) &= \text{xite} \times (\text{youtx}(k) - \text{ymoutx}(k)) \times \text{wx}_1(j) \times h_x(j) \times 2 \times (x(i) - \text{cix}_1(i,j)) \times (\text{bix}_1(j)^{-2}); \\
  d_{\text{ciy}}(i,j) &= \text{xite} \times (\text{youty}(k) - \text{ymouty}(k)) \times \text{wy}_1(j) \times h_y(j) \times 2 \times (y(i) - \text{ciy}_1(i,j)) \times (\text{biy}_1(j)^{-2}); \\
  d_{\text{ciz}}(i,j) &= \text{xite} \times (\text{youtz}(k) - \text{ymoutz}(k)) \times \text{wz}_1(j) \times h_z(j) \times 2 \times (z(i) - \text{ciz}_1(i,j)) \times (\text{biz}_1(j)^{-2}); \\
\end{align*}
\]
\textbf{end}
\textbf{end}

\[\text{cix} = \text{cix}_1 + \text{xite} \times \text{d}_{\text{cix}} + \alpha \times (\text{cix}_1 - \text{cix}_2) + \beta \times (\text{cix}_2 - \text{cix}_3);\]
\[\text{ciy} = \text{ciy}_1 + \text{xite} \times \text{d}_{\text{ciy}} + \alpha \times (\text{ciy}_1 - \text{ciy}_2) + \beta \times (\text{ciy}_2 - \text{ciy}_3);\]
\[\text{ciz} = \text{ciz}_1 + \text{xite} \times \text{d}_{\text{ciz}} + \alpha \times (\text{ciz}_1 - \text{ciz}_2) + \beta \times (\text{ciz}_2 - \text{ciz}_3);\]

\[yux = 0;\]
\[yuy = 0;\]
\[yuz = 0;\]
\textbf{for} \ j = 1:1:6
\[
\begin{align*}
  yux &= yux + \text{wx}(j) \times h_x(j) \times 2 \times (\text{cix}(1,j) - x(1)) / (\text{bix}(j)^2); \\
  yuy &= yuy + \text{wy}(j) \times h_y(j) \times 2 \times (\text{ciy}(1,j) - y(1)) / (\text{biy}(j)^2); \\
  yuz &= yuz + \text{wz}(j) \times h_z(j) \times 2 \times (\text{ciz}(1,j) - z(1)) / (\text{biz}(j)^2); \\
\end{align*}
\]
\textbf{end}

\[\text{dyoutx}(k) = yux;\]
\[\text{dyouty}(k) = yuy;\]
\[\text{dyoutz}(k) = yuz;\]

\[\text{error}(::,k) = \text{rin}(::,s) - y_1;\]
\[\text{vc}(::,k) = \text{error}(::,k) / \text{ts};\]
\% \[\text{ac}(::,k) = (\text{vc}(::,k) - \text{v}(::,k)) / \text{ts};\]
\[\text{kpx}(k) = \text{kpx}_1 \times \text{dk}_{\text{kpx}} \times \text{error}(1,k) \times \text{dyoutx}(k) \times e_1(1);\]
\[
kdx(k) = kdx_{-1} + xitekdx * \text{error}(1,k) * \text{dyout}(k) * e_2(1);
\]

\[
kpy(k) = kpy_{-1} + xitekpy * \text{error}(2,k) * \text{dyout}(k) * e_1(2);
\]

\[
kdy(k) = kdy_{-1} + xitekdy * \text{error}(2,k) * \text{dyout}(k) * e_2(2);
\]

\[
kpz(k) = kpz_{-1} + xitekpz * \text{error}(3,k) * \text{dyout}(k) * e_1(3);
\]

\[
kdz(k) = kdz_{-1} + xitekdz * \text{error}(3,k) * \text{dyout}(k) * e_2(3);
\]

\[
u(1,k) = kpx(k) * \text{error}(1,k) + kdx(k) * (vc(1,k) - v_1(1));
\]

\[
u(2,k) = kpy(k) * \text{error}(2,k) + kdy(k) * (vc(2,k) - v_1(2));
\]

\[
u(3,k) = kpz(k) * \text{error}(3,k) + kdz(k) * (vc(3,k) - v_1(3));
\]

\[
fi_c(k) = \text{asin}((u(1,k) * \sin(fy_c(k)) - u(2,k) * \cos(fy_c(k))) / \sqrt{u(1,k)^2 + u(2,k)^2 + (u(3,k) + g)^2});
\]

\[
theta_c(k) = \text{atan}((u(1,k) * \cos(fy_c(k)) + u(2,k) * \sin(fy_c(k))) / (u(3,k) + g));
\]

\[
\text{angle}_c(:,k) = [fi_c(k) \quad theta_c(k) \quad fy_c(k)]';
\]

\[
\text{angular}_v_c(:,k) = \text{angle}_c(:,k) / \text{ts};
\]

\[
dw_fi(k) = kp_fi * (\text{angle}_c(1,k) - \text{angle}(1,k)) + kd_fi * (\text{angular}_v_c(1,k) - \text{angular}_v(1,k));
\]

\[
dw_theta(k) = kp_theta * (\text{angle}_c(2,k) - \text{angle}(2,k)) + kd_theta * (\text{angular}_v_c(2,k) - \text{angular}_v(2,k));
\]

\[
dw_fy(k) = kp_fy * (\text{angle}_c(3,k) - \text{angle}(3,k)) + kd_fy * (\text{angular}_v_c(3,k) - \text{angular}_v(3,k));
\]

\[
wh = \sqrt{m * g / (4 * b)};
\]

\[
w_rotor(:,k) = A * [wh + dw_f(k) \quad dw_fi(k) \quad dw_theta(k) \quad dw_fy(k)]';
\]

\[
C = [1 * (b * (w_rotor(4,k)^2) - b * (w_rotor(2,k)^2)) \quad 1 * (b * (w_rotor(3,k)^2) - b * (w_rotor(1,k)^2)) \quad c * (w_rotor(1,k)^2) + c * (w_rotor(4,k)^2) - c * (w_rotor(3,k)^2)]';
\]

\[
\text{angular}_v_1(:,k) = \text{angular}_v(:,k);
\]

\[
\text{if } w_air == 0
\]

\[
\text{signw11} = \text{sign}(\text{angular}_v(1,k)); \text{signw12} = -1;
\]

\[
\text{elseif } \text{angular}_v(1,k) >= 0 && \text{abs}(w_air) >= \text{abs}(\text{angular}_v(1,k))
\]

\[
\text{signw11} = -1; \text{signw12} = 1;
\]

\[
\text{elseif } \text{angular}_v(1,k) >= 0 && \text{abs}(w_air) <= \text{abs}(\text{angular}_v(1,k))
\]

\[
\text{signw11} = -1; \text{signw12} = +1;
\]

\[
\text{elseif } \text{angular}_v(1,k) < 0
\]

\[
\text{signw11} = 1; \text{signw12} = -1;
\]

\[
\text{end}
\]

\[
\text{if } w_air == 0
\]

\[
\text{signw21} = \text{sign}(\text{angular}_v(2,k)); \text{signw22} = -1;
\]

\[
\text{elseif } \text{angular}_v(2,k) >= 0 && \text{abs}(w_air) >= \text{abs}(\text{angular}_v(2,k))
\]

\[
\text{signw21} = -1; \text{signw22} = 1;
\]

\[
\text{elseif } \text{angular}_v(2,k) >= 0 && \text{abs}(w_air) <= \text{abs}(\text{angular}_v(2,k))
\]

\[
\text{signw21} = -1; \text{signw22} = +1;
\]

\[
\text{elseif } \text{angular}_v(2,k) < 0
\]

\[
\text{signw21} = 1; \text{signw22} = -1;
\]

\[
\text{end}
\]

\[
\text{if } w_air == 0
\]

\[
\text{signw31} = \text{sign}(\text{angular}_v(3,k)); \text{signw32} = -1;
\]

\[
\text{elseif } \text{angular}_v(3,k) >= 0 && \text{abs}(w_air) >= \text{abs}(\text{angular}_v(3,k))
\]

\[
\text{signw31} = -1; \text{signw32} = 1;
\]

\[
\text{elseif } \text{angular}_v(3,k) >= 0 && \text{abs}(w_air) <= \text{abs}(\text{angular}_v(3,k))
\]

\[
\text{signw31} = -1; \text{signw32} = +1;
\]

\[
\text{elseif } \text{angular}_v(3,k) < 0
\]

\[
\text{signw31} = 1; \text{signw32} = -1;
\]

\[
\text{end}
\]
signw31=-1;signw32=+1;

elseif angular_v(3,k)<0
    signw31=1;signw32=-1;
end

fuhao1=sign(C);

angular_v_a(:,k)=inv(If)*(-cross(angular_v(:,k),(If*angular_v(:,k))-
Ir*cross(angular_v(:,k),ez)*(w_rotor(1,k)-w_rotor(2,k)+w_rotor(3,k)-
w_rotor(4,k))+[fuhao1(1) 0 0;0 fuhao1(2) 0;0 0 fuhao1(3)]*[abs(C)+[signw12 0 0;0 signw22 0;0 0 signw32]*kr*([w_air
w_air w_air]'+[signw11*angular_v(1,k) signw21*angular_v(2,k)
signw31*angular_v(3,k)]'))

%f % angular_v_a(:,k)=inv(If)*(-cross(angular_v(:,k),(If*angular_v(:,k))-
Ir*cross(angular_v(:,k),ez)*(w_rotor(1,k)-w_rotor(2,k)+w_rotor(3,k)-
w_rotor(4,k))+C);
angular_v(:,k)=angular_v(:,k)+angular_v_a(:,k)*ts;

angle(:,k)=angle(:,k)+angular_v(:,k)*ts;

R=[cos(angle(2,k))*cos(angle(3,k))
   sin(angle(2,k))*cos(angle(3,k))*sin(angle(1,k))-
   sin(angle(3,k))*cos(angle(1,k))
   sin(angle(2,k))*cos(angle(3,k))*cos(angle(1,k))+sin(angle(3,k))]*sin(angle(1,k))

% R=[cos(angle_c(2,k))*cos(angle_c(3,k))
% sin(angle_c(2,k))*cos(angle_c(3,k))*sin(angle_c(1,k))-
% sin(angle_c(3,k))*cos(angle_c(1,k))
% sin(angle_c(2,k))*cos(angle_c(3,k))*cos(angle_c(1,k))+sin(angle_c(3,k))]*sin(angle_c(1,k))

T(k)=m*(u(1,k)*(sin(angle(2,k))*cos(angle(3,k))*cos(angle(1,k))+sin(angle(3,k))*sin(angle(1,k)))+
   u(2,k)*(sin(angle(2,k))*sin(angle(3,k))*cos(angle(1,k))-
   cos(angle(3,k))*sin(angle(1,k)))+(u(3,k)+g)*cos(angle(2,k))*cos(angle(1,k))];

fuhao2=sign(R*(T(k)*[0 0 1]'))

if v_air==0
    sign11=sign(v_1(1));sign12=-1;
elseif v_1(1)>=0&&abs(v_air)>=abs(v_1(1))
    sign11=-1;sign12=1;
elseif v_1(1)>=0&&abs(v_air)<=abs(v_1(1))
    sign11=-1;sign12=+1;
elseif v_1(1)<0
    sign11=1;sign12=-1;
end
if v_air==0
    sign21=sign(v_1(2));sign22=-1;
elseif v_1(2)>=0 && abs(v_air)>=abs(v_1(2))
    sign21=-1;sign22=1;
elseif v_1(2)>=0 && abs(v_air)<=abs(v_1(2))
    sign21=-1;sign22=+1;
elseif v_1(2)<0
    sign21=1;sign22=-1;
end
if v_air==0
    sign31=sign(v_1(3));sign32=-1;
elseif v_1(3)>=0 && abs(v_air)>=abs(v_1(3))
    sign31=-1;sign32=1;
elseif v_1(3)>=0 && abs(v_air)<=abs(v_1(3))
    sign31=-1;sign32=+1;
elseif v_1(3)<0
    sign31=1;sign32=-1;
end
TH=R*(T(k)*[0 0 1]');
UA(1,k)=(1/m)*fuhao2(1)*(abs(TH(1))+sign12*kt*(v_air+sign11*v_1(1)));
UA(2,k)=(1/m)*fuhao2(2)*(abs(TH(2))+sign22*kt*(v_air+sign21*v_1(2)));
UA(3,k)=-g+(1/m)*fuhao2(3)*(abs(TH(3))+sign32*kt*(v_air+sign31*v_1(3)));
v(:,k)=v_1+UA(:,k)*ts;
\% yout(:,k)=y_1+v(:,k)*ts;
x(1)=u(1,k);
x(2)=yout(1,k);
x(3)=y_1(1);
\% UA_1(:,k)=UA(:,k);
y(1)=u(2,k);
y(2)=yout(2,k);
y(3)=y_1(2);
z(1)=u(3,k);
z(2)=yout(3,k);
z(3)=y_1(3);
\% UA_1(1)=UA(1,k);
y_1=yout(:,k);
v_1=v(:,k);
cix_3=cix_2;
cix_2=cix_1;
cix_1=cix;
w_3=w_2;
w_2=w_1;
w_1=w;
ciy_3=ciy_2;
ciy_2=ciy_1;
ciy_1=ciy;
w_3=w_2;
wy_2=wy_1;
wy_1=wy;

ciz_3=ciz_2;
ciz_2=ciz_1;
ciz_1=ciz;
wz_3=wz_2;
wz_2=wz_1;
wz_1=wz;

%     u_1=u(:,k);

e_2=error(:,k)-2*error_1+error_2;
e_1=error(:,k)-error_1;
error_2=error_1;
error_1=error(:,k);

kpx_1=kpx(k);
kdx_1=kdx(k);

kpy_1=kpy(k);
kdy_1=kdy(k);

kpz_1=kpz(k);
kdz_1=kdz(k);

end

xxx(:,k)=R*ez;
%
plot3(yout(1,k),yout(2,k),yout(3,k),'+');hold on;
%
plot3(rin(1,s),rin(2,s),rin(3,s),'-');hold on;
%
grid on;
% errorp(:,k)=abs(yout(:,k)-rin(:,k));
errornp(:,k)=abs(yout(:,k)-rin(:,k));

end

% x2(sx)=sqrt(mean((yout(1,200:350)-rin(1,200:350)).^2));
% y2(sx)=sqrt(mean((yout(2,200:350)-rin(2,200:350)).^2));
% z2(sx)=sqrt(mean((yout(3,200:350)-rin(3,200:350)).^2));
% if sx==1
%     save('D:\adaptivecontrol15x\errorpp1','errorp');
% elseif sx==2
%     save('D:\adaptivecontrol15x\errorpp2','errorp');
%     else
%         save('D:\adaptivecontrol15x\errorpp3','errorp');
%     end
% if sx==3
%     save('D:\adaptivecontrol15x\errorpp4','errorp');
% else
%         save('D:\adaptivecontrol15x\errorpp5','errorp');
%     end
% if sx==4
%     save('D:\adaptivecontrol15x\errorpp6','errorp');
% else
%         save('D:\adaptivecontrol15x\errorpp7','errorp');
% end
% if sx==5
%     save('D:\adaptivecontrol15x\errorpp8','errorp');
% else
%         save('D:\adaptivecontrol15x\errorpp9','errorp');
% end
% if sx==6
%     save('D:\adaptivecontrol15x\errorpp10','errorp');
% else
%     save('D:\adaptivecontrol15x\errorpp11','errorp');
% end

91
end

if sx==1
    save('D:\adaptivecontrol15x\errornp1','errornp');
elseif sx==2
    save('D:\adaptivecontrol15x\errornp2','errornp');
elseif sx==3
    save('D:\adaptivecontrol15x\errornp3','errornp');
elseif sx==4
    save('D:\adaptivecontrol15x\errornp4','errornp');
elseif sx==5
    save('D:\adaptivecontrol15x\errornp5','errornp');
    elseif sx==6
        save('D:\adaptivecontrol15x\errornp6','errornp');
        elseif sx==7
            save('D:\adaptivecontrol15x\errornp7','errornp');
            elseif sx==8
                save('D:\adaptivecontrol15x\errornp8','errornp');
                elseif sx==9
                    save('D:\adaptivecontrol15x\errornp9','errornp');
                    elseif sx==10
                        save('D:\adaptivecontrol15x\errornp10','errornp');
end

end

% ssx=1:1:20;
% subplot(3,1,1);
% plot(ssx,x1,'r',ssx,x2,'b');legend('x-PD','x-Neural PD');
% xlabel('t-air(m/s)');
% ylabel('RMSE-x');
% subplot(3,1,2);
% plot(ssx,y1,'r',ssx,y2,'b');legend('x-PD','x-Neural PD');
% xlabel('t-air(m/s)');
% ylabel('RMSE-y');
% subplot(3,1,3);
% plot(ssx,z1,'r',ssx,z2,'b');legend('x-PD','x-Neural PD');
% xlabel('t-air(m/s)');
% ylabel('RMSE-z');

ssx=1:1:20;
% subplot(3,1,1);
% plot(ssx,x1,'r',ssx,x2,'b');legend('x-PD','x-Neural PD');
% % xlabel('r-air(m/s)');
% ylabel('RMSE-x');
% subplot(3,1,2);
% plot(ssx,y1,'r',ssx,y2,'b');legend('x-PD','x-Neural PD');
% % xlabel('r-air(--*rand*m/s)');
% ylabel('RMSE-y');
% subplot(3,1,3);
% plot(ssx,z1,'r',ssx,z2,'b');legend('x-PD','x-Neural PD');
% % xlabel('r-air(--*rand*m/s)');
% ylabel('RMSE-z');

92
% ssx=1:1:20;
% subplot(3,1,1);
% plot(ssx,x2,'b');legend('x-PD','x-Neural PD');
% subplot(3,1,2);
% plot(ssx,y2,'b');legend('x-PD','x-Neural PD');
% subplot(3,1,3);
% plot(ssx,z2,'b');legend('x-PD','x-Neural PD');
% x1=xlabel('x(m)');
% x2=ylabel('y(m)');
% x3=zlabel('z(m)');
% d=1:1:60;
% plot(l2h(1,d),'+');hold on;
% plot(l2h(2,d),'*');hold on;
% plot(l2h(3,d),'+');hold on;
x=1:1:600;
% subplot(2,1,1);
% plot(0.1*x,angle_c(1,x),'linewidth',3);hold on;
% plot(0.1*x,angle_c(2,x),'linewidth',2);hold on;
% plot(0.1*x,angle_c(3,x));hold on;
% text(25,0.08,'desired angle');
% subplot(2,1,2);
% plot(0.1*x,angle(1,x),'linewidth',3);hold on;
% plot(0.1*x,angle(2,x),'linewidth',2);hold on;
% plot(0.1*x,angle(3,x));hold on;
% text(25,0.12,'real angle');
% plot(0.01*x,toque(1,x),'-');hold on;
% plot(0.01*x,toque(2,x),'-');hold on;
% plot(0.01*x,toque(3,x));hold on;
% text(50,0.0008,'Toque');
% % plot(0.01*x,T(x));
% axis([0 60 4.5 5]);
% text(45,4.95,'Thrust');
% plot(0.1*x,l1h(1,x),'-');hold on;
% plot(0.1*x,l1h(2,x),red');hold on;
% plot(0.1*x,l1h(3,x),red');hold on;
% text(40,0.0045,'l1 estimated');
% plot(0.01*x,l2h(1,x),'-');hold on;
% plot(0.01*x,l2h(2,x),'-');hold on;
% plot(0.01*x,l2h(3,x),'-');hold on;
% text(30,1.8*10^-5,'l2 estimated');
% plot(0.1*x,kth_p(1,x),'-');hold on;
% plot(0.1*x,kth_p(2,x),'black');hold on;
% plot(0.1*x,kth_p(3,x),red');hold on;
% text(45,0.025,'kt estimated');
% plot(0.01*x,krh_p(1,x),'-');hold on;
% plot(0.01*x,krh_p(2,x),'green');hold on;
% plot(0.01*x,krh_p(3,x),red');hold on;
% % axis([0 60 -1 0.01]);
% text(40,-0.1*10^-3,'kr estimated');
% plot(0.1*x,yout(1,x)-rin(1,x));hold on;
% plot(0.1*x,yout(2,x)-rin(2,x));hold on;
% plot(0.1*x,yout(3,x)-rin(3,x));hold on;
% plot(0.1*x,angle(1,x)-angle_c(1,x));hold on;
% plot(0.1*x,angle(2,x)-angle_c(2,x));hold on;
% plot(0.1*x,angle(3,x)-angle_c(3,x));hold on;
Appendix B

Codes for adaptive controller

clear all;
close all;
clc;
v_air=0;
w_air=0;
m=0.468; g=9.81;
I_r=3.357*10^(-5); I_x=4.856*(10^(-3)); I_y=4.856*(10^(-3)); I_z=8.801*(10^(-3));
I_f=[I_x 0 0; 0 I_y 0; 0 0 I_z];
b=2.98*10^(-6); k=1.14*10^(-7);
l=0.225;
k_1=eye(3)*0.8;
k_3=eye(3)*0.25;
k_2=eye(3)*1.2;
k_4=eye(3)*5;
\theta_0_1=eye(3)*0.15;
\theta_0_2=eye(3)*0.04;
\theta_0_3=eye(3)*0.001;
\theta_0_4=eye(3)*0.0001;
t_b=eye(3)*15;
k_r=eye(3)*0.001;
a=2*t_b;
t=0;
\text{% dt2=0.01; }
x11=0;
x12=0;
x13=0;
x21=0;
x22=0;
x23=0;
y1=0;
y2=0;
y3=0;
f_y_c=zeros(7000,1);
U=zeros(3,7000);
U_A=zeros(3,7000);
dt=0.1;
p_c=zeros(3,7000);
v_c=zeros(3,7000);
a_c=zeros(3,7000);
z_1=zeros(3,7000);
z_2=zeros(3,7000);
z_3=zeros(3,7000);
z_4=zeros(3,7000);
\theta_c_c=zeros(7000,1);
f_i_c=zeros(7000,1);
\theta_c=zeros(7000,1);
f_i=zeros(7000,1);
f_y=zeros(7000,1);

p=zeros(3,7000);
T=zeros(7000,1);
% kt=[0.02 0 0;0 0.04 0;0 0 0.01];
kt=[0.01 0 0;0 0.01 0;0 0 0.01];
angle_c=zeros(3,7000);
angle_r=zeros(3,7000);
angle_re=zeros(3,60);
angle_c_es_diff=zeros(3,7000);
angular_v_d=zeros(3,7000);
angular_v_c=zeros(3,7000);
% from angle controller loop
-----------------------------------------------

------------------------

angular_v=zeros(3,7000);
angular_v_1=zeros(3,7000);
% for m=1:1
%     angular_v(:,1)=[0.001 0.001 0.001]';
% end
angular_v_a=zeros(3,7000);
lameda=zeros(3,7000);
lameda_con=zeros(3,7000);
dt1=0.1;
krh=zeros(3,3);
l2h=zeros(3,7000);
angular_v_con=zeros(3,7000);
step_a=0;

---------------------------------------------------------------------

---------------------------------------------

% from position controller
step=1;
kth=zeros(3,3);

  kth_d=zeros(3,3);
l1h=zeros(3,7000);

---------------------------------------------------------------------

s=0;

T_bar=eye(3)*15;
W=zeros(3,3);
data=zeros(3,7000);
length=1;
data_h=zeros(3,60);
tt=0;
Thrust=zeros(1,60);
ez=[0 0 1]';
kth_p=zeros(3,7000);
krh_p=zeros(3,7000);
torque=zeros(3,7000);
w_rotor=zeros(4,7000);
C=zeros(3,1);
A=[1 0 -1;1 1 0 -1;1 0 1 1;1 -1 0 -1];
B=[b b b b;0 -b*l 0 b*l;-b*l 0 b*l 0;-k k -k k];
e_angle=zeros(3,7000);
e_angular_v=zeros(3,7000);
kp_angle=diag([0.001 -0.01 -0.04]);
kd_angle=diag([-0.08 -0.0008 0.01]);
angle_c_es_a=zeros(3,7000);
dt2=0.1;
% dt3=10;
kp_fi=0.0001;
kd_fi=0.000001;
kp_theta=-0.0001;
kd_theta=0.000001;
kp_fy=0;
kd_fy=0;
dw_fi=zeros(7000,1);
dw_theta=zeros(7000,1);
dw_fy=zeros(7000,1);
wh=0;
dw_f=zeros(7000,1);
wh_ror2=zeros(4,7000);
for k=1:1
p(:,k)=[5 0 0]';
end
v=zeros(3,2000);
v_1=zeros(3,1);
fuhao1=zeros(3,1);
fuhao2=zeros(3,1);
for i=1:600
  t=t+0.1;
  s=s+1;
  pc(:,s)=[5*cos(0.1*t) 5*sin(0.1*t) 0.5]';
  ac(:,s)=[-0.5*sin(0.1*t) 0.5*cos(0.1*t) 0]';
for j=1:1
%         tic;
    step=step+1;
    z1(:,step)=p(:,step-1)-pc(:,s);
    vc(:,step)=[-0.5*sin(0.1*t) 0.5*cos(0.1*t) 0]'-z1(:,step);
    z2(:,step)=v(:,step-1)-vc(:,step);
    kth=(1/m)*tao1*z2(:,step)*v(:,step-1)';
    l1h(:,step)=(1/m)*tao2*diag(sign(z2(:,step)))*z2(:,step);%
    l1h(:,step)=l1h(:,step-1)+(1/m)*tao2*diag([tanh(z2(1,step))
        tanh(z2(2,step)) tanh(z2(3,step))])*z2(:,step);
    kth_d=diag([kth(1,1),kth(2,2),kth(3,3)]);
    kth_p(:,step-1)=[kth(1,1) kth(2,2) kth(3,3)]';
    U(:,step)=ac(:,s)-(k1-eye(3))*z1(:,step)-(k2+eye(3))*z2(:,step)+
            (1/m)*kth_d*v(:,step-1)-
            (1/m)*diag(sign(z2(:,step)))*11h(:,step);
    fi_c(step)=asin((U(1,step)*sin(fy_c(step))-
        U(2,step)*cos(fy_c(step)))/sqrt(U(1,step)^2+U(2,step)^2+(U(3,step)+g)^2));
    theta_c(step)=atan(((U(1,step)*cos(fy_c(step))+
        U(2,step)*sin(fy_c(step)))/(U(3,step)+g));
    angle_c(:,step)=[fi_c(step) theta_c(step) fy_c(step)]';
    angle_c_es_diff(:,step)=[(900*fi_c(step)*dt2)/exp(30*dt2)
        (900*theta_c(step)*dt2)/exp(30*dt2) (900* fy_c(step)*dt2)/exp(30*dt2)]';
end
for k=1:1
    step_a=step_a+1;
    z3(:,step_a)=angle(:,step_a)-angle_c(:,step);
    W=[1 sin(angle(1,step_a))*tan(angle(2,step_a))
       cos(angle(1,step_a))*tan(angle(2,step_a));
       0 cos(angle(1,step_a))];
    for k=1:1
        z3(:,step_a)=angle(:,step_a)-
                     angle_c(:,step);
        W=
           [1 sin(angle(1,step_a))*tan(angle(2,step_a))
            cos(angle(1,step_a))*tan(angle(2,step_a));
            0 cos(angle(1,step_a))];
        angular_v_d(:,step_a)=W\(angle_c_es_diff(:,step)-
                       k3*z3(:,step_a));
        angular_v_c(:,step_a+1)=[angular_v_d(1,step_a)-
             angular_v_d(1,step_a)/exp(15*10) angular_v_d(2,step_a)-
             angular_v_d(2,step_a)/exp(15*10) angular_v_d(3,step_a)-
             angular_v_d(3,step_a)/exp(15*10)]';
        z4(:,step_a)=angular_v(:,step_a)-angular_v_c(:,step_a+1);
        lameda_con(:,step_a)=W*(angular_v_c(:,step_a+1)-
             angular_v_d(:,step_a));
        lameda(:,step_a)=[(25*ameda_con(1,step_a))/9 -
                       (25*ameda_con(1,step_a))/9*exp((9*dt1)/25))
                       (25*ameda_con(2,step_a))/9 -
                       (25*ameda_con(2,step_a))/9*exp((9*dt1)/25))
                       (25*ameda_con(3,step_a))/9 -
                       (25*ameda_con(3,step_a))/9*exp((9*dt1)/25)];
        krh=(tao3/If)*z4(:,step_a)*angular_v(:,step_a)'*dt;
        l2h(:,step_a+1)=(tao4/If)*diag(sign(z4(:,step_a)))*z4(:,step_a)*dt;
        krh_p(:,step_a)=krh*angular_v(:,step_a)'
        toque(:,step_a)=cross(angular_v(:,step_a),If*angular_v(:,step_a))'+Ir*cross(angular_v(:,step_a),ez)*(w_rotor(1,step-1)-w_rotor(2,step-1)-w_rotor(3,step-1)-w_rotor(4,step-1)-
             If*tb*angular_v_c(:,step_a)+(If*W)*angle_c_es_diff(:,step)-
             (If*W)*k3*(angle(:,step_a)-angle_c(:,step))-If*W'*(angle(:,step_a)-
             angle_c(:,step))-ameda(:,step_a))-
             If*k4*(angular_v(:,step_a)-
             angular_v_c(:,step_a))-krh*angular_v(:,step_a)-
             diag(sign(z4(:,step_a)))*12h(:,step_a+1);
        angular_v_1(:,step_a)=angular_v(:,step_a);
        if angular_v_1(:,step_a)<=0
            angular_v_1(:,step_a)=-angular_v_1(:,step_a);
        end
    end
    fuhaol=sign(toque(:,step_a));
    angular_v_a(:,step_a+1)=If\(-
                             cross(angular_v(:,step_a),If*angular_v(:,step_a))-
                             Ir*cross(angular_v(:,step_a),ez)*(w_rotor(1,step-1)-w_rotor(2,step-1)-w_rotor(3,step-1)-w_rotor(4,step-1)))+fuhaol(1) 0 0;0 fuhaol(2) 0;0
    fuhaol(3)]*(abs(toque(:,step_a))-kr*angular_v_1(:,step_a)-
                                [w_air w_air w_air]));
    angular_v(:,step_a+1)=angular_v_a(:,step_a+1)*dt1;
    angular_v(:,step_a+1)=angular_v(:,step_a+1)*dt1;
end
R=[cos(angle(2,step_a+1))*cos(angle(3,step_a+1))
   sin(angle(2,step_a+1))*cos(angle(3,step_a+1))*sin(angle(1,step_a+1))+
   sin(angle(2,step_a+1))*cos(angle(3,step_a+1))*cos(angle(1,step_a+1))
   sin(angle(2,step_a+1))*cos(angle(3,step_a+1))*cos(angle(1,step_a+1))+
   sin(angle(3,step_a+1))*cos(angle(1,step_a+1))^2];
\cos(\angle(2, \text{step}_a+1)) \cdot \sin(\angle(3, \text{step}_a+1)) \\
\sin(\angle(2, \text{step}_a+1)) \cdot \sin(\angle(3, \text{step}_a+1)) \cdot \sin(\angle(1, \text{step}_a+1)) + \cos(\angle(3, \text{step}_a+1)) \cdot \cos(\angle(1, \text{step}_a+1)) \\
- \cos(\angle(2, \text{step}_a+1)) \cdot \sin(\angle(3, \text{step}_a+1)) \cdot \cos(\angle(1, \text{step}_a+1)) \\
- \sin(\angle(2, \text{step}_a+1)) \\
\cos(\angle(2, \text{step}_a+1)) \cdot \sin(\angle(1, \text{step}_a+1)) \\
\cos(\angle(2, \text{step}_a+1)) \cdot \cos(\angle(1, \text{step}_a+1))];

T(\text{step}) = m \cdot (U(1, \text{step}) \cdot (\sin(\angle(2, \text{step}_a+1)) \cdot \cos(\angle(3, \text{step}_a+1)) \cdot \cos(\angle(1, \text{step}_a+1)) + \sin(\angle(3, \text{step}_a+1)) \cdot \sin(\angle(1, \text{step}_a+1))) + U(2, \text{step}) \cdot (\sin(\angle(2, \text{step}_a+1)) \cdot \sin(\angle(3, \text{step}_a+1)) \cdot \cos(\angle(1, \text{step}_a+1)) - \cos(\angle(3, \text{step}_a+1)) \cdot \sin(\angle(1, \text{step}_a+1)));

w_{\text{rotor2}}(:, \text{step}) = \text{inv}(B) \cdot [T(\text{step}) \cdot \text{toque}(1, \text{step}_a) \cdot \text{toque}(2, \text{step}_a) \cdot \text{toque}(3, \text{step}_a)]';

w_{\text{rotor}}(:, \text{step}) = [\sqrt{w_{\text{rotor2}}(1, \text{step})} \cdot \sqrt{w_{\text{rotor2}}(2, \text{step})} \cdot \sqrt{w_{\text{rotor2}}(3, \text{step})} \cdot \sqrt{w_{\text{rotor2}}(4, \text{step})}]';

v_1 = v(:, \text{step}-1);
if v_1 <= 0
    v_1 = -v_1;
end

fuhao2 = sign(R \cdot (T(\text{step}) \cdot \text{ez}));

UA(:, \text{step}) = g \cdot [0 0 1] + (1/m) \cdot [fuhao2(1) 0 0; fuhao2(2) 0 0; fuhao2(3)] \cdot (\text{abs}(R \cdot (T(\text{step}) \cdot \text{ez}))) - k_t \cdot (v_1 - [v_{\text{air}} v_{\text{air}} v_{\text{air}}]);

v(:, \text{step}) = v(:, \text{step}-1) + UA(:, \text{step}) \cdot \text{dt};

p(:, \text{step}) = p(:, \text{step}-1) + v(:, \text{step}) \cdot \text{dt};
% ylabel('Attitude(rad)');
% subplot(1,2,2);
% plot(0.1*x,angle(1,x), 'r'); hold on;
% plot(0.1*x,angle(2,x), 'g'); hold on;
% plot(0.1*x,angle(3,x)); hold on;
% legend('Pitch', 'Roll', 'Yaw', 'location', 'NorthEast');
% % text(25,0.12,'real angle');
% title('Real Attitude');
% xlabel('Time(s)');
% ylabel('Attitude(rad)');
% plot(0.1*x,toque(1,x),'r'); hold on;
% plot(0.1*x,toque(2,x),'g'); hold on;
% plot(0.1*x,toque(3,x)); hold on;
% % text(50,0.0008,'toque');
% legend('Toque in Pitch', 'Toque in Roll', 'Toque in Yaw', 'location', 'NorthEast');
% xlabel('Time(s)');
% ylabel('Toque(N*m)');

% plot(0.1*x,T(x));
% axis([0 60 4.5 5]);
% % text(45,4.95,'Thrust');
% legend('Thrust', 'location', 'NorthEast');
% xlabel('Time(s)');
% ylabel('Thrust(N)');

% plot(0.1*x,l1h(1,x), 'r'); hold on;
% plot(0.1*x,l1h(2,x), 'g'); hold on;
% plot(0.1*x,l1h(3,x)); hold on;
% % text(40,0.0045,'l1 estimated');
% xlabel('Time(s)');
% ylabel('Estimated Wind Disturbance Upper bound(N*s/m)');
% legend('l1(1)', 'l1(2)', 'l1(3)', 'location', 'NorthEast');

plot(0.01*x,l2h(1,x), 'r'); hold on;
plot(0.01*x,l2h(2,x), 'g'); hold on;
plot(0.01*x,l2h(3,x), 'b'); hold on;
% text(30,1.8*10^(-5),'l2 estimated');
% xlabel('Time(s)');
% ylabel('Estimated Wind Disturbance Upper bound (Nm*rad/m)');
% legend('l2(1)', 'l2(2)', 'l2(3)', 'location', 'NorthEast');

% plot(0.1*x,kth_p(1,x), 'r'); hold on;
% plot(0.1*x,kth_p(2,x), 'g'); hold on;
% plot(0.1*x,kth_p(3,x), 'b'); hold on;
% text(45,0.025,'kt estimated');
% xlabel('Time(s)');
% ylabel('Estimated Aerodynamic Force Coefficient(N*s/m)');
% legend('Kt(1)', 'Kt(2)', 'Kt(3)', 'location', 'NorthEast');

% plot(0.01*x,krh_p(1,x), 'r'); hold on;
% plot(0.01*x,krh_p(2,x), 'g'); hold on;
% plot(0.01*x,krh_p(3,x), 'b'); hold on;
% % text(30,0.025, 'Kr estimated');
% xlabel('Time(s)');
% ylabel('Estimated Aerodynamic Moment Coefficient(Nm*s/rad)');
% legend('Kr(1)', 'Kr(2)', 'Kr(3)', 'location', 'NorthEast');
axis([0 60 -1 0.01]);

text(40,-0.1*10^(-3),'kr estimated');

plot(0.1*x,p(1,x+1)-pc(1,x));hold on;
plot(0.1*x,p(2,x+1)-pc(2,x));hold on;
plot(0.1*x,p(3,x+1)-pc(3,x));hold on;
plot(0.1*x,angle(1,x+1)-angle_c(1,x));hold on;
plot(0.1*x,angle(2,x+1)-angle_c(2,x));hold on;
plot(0.1*x,angle(3,x+1)-angle_c(3,x));hold on;