Linear vs. Nonlinear Control Techniques for a Quadrotor Vehicle

Younes M. Al-Younes • Mohammed A. Al-Jarrah •
Ali A. Jhemi

American University of Sharjah
Department of Mechanical Engineering
P.O. Box 26666, Sharjah, UAE
mjarrah@aus.edu

ABSTRACT

The aim of this paper is to propose a nonlinear control algorithm that is able to stabilize the attitude and altitude of a quadrotor vehicle, and then to compare it with the classical techniques, such as PID and LQR. The nonlinear algorithm is based on the backstepping methodology integrated with the integral and adaptation schemes. It's called "Adaptive Integral Backstepping Controller" (AIBC). The recursive Lyapunov methodology in the backstepping technique will ensure the system stability, the integral action will increase the system robustness against disturbances and model uncertainties, and the adaptation law will estimate the modeling errors caused by assumptions in simplifying the complexity of the quadrotor model. Furthermore, a quadrotor vehicle is designed and modeled in this paper to perform the controlling task,

Keywords Backstepping · Adaptation · Lyapunov · Attitude · Altitude

1 Introduction

Recently, unmanned aerial vehicles (UAVs) gained increased attention for their civil and scientific research applications. Progress in sensing elements, control theories, and data processing techniques lead to the rapid development of the UAV applications in different fields. Important class of UAVs is the Vertical Take-Off and Landing (VTOL) rotorcraft that has four lift-generating propellers known as the quadrotor.

The quadrotor is a six degrees of freedom (6DOF) vehicle: yaw, pitch, roll, x (longitudinal motion), y (lateral motion), and z (altitude). These states are controlled by controlling the thrust of the four rotors.

Due to the complexity of the dynamic model, and the nonlinearity of the quadrotor system, its control becomes quite a challenge. The stability issues of the attitude and altitude are the main objectives for most studies conducted in this field. Sliding mode control, basic PID, and LQR control are some of the control methods that have been applied to the quadrotor platform.

In most of the studies the control algorithm is based on a linearized version of the quadrotor model. The linearization will restrict the control to be valid for a certain conditions like the hover flight condition. The drifting and fluctuating of the quadrotor in hover flight were some of the problems that faced the researcher in their work. So, a nonlinear controller is suitable to overcome these challenging. [1]

This paper will be going through five chapters:
Chapter 2: "*Quadrotor Platform*": This chapter concerns the development of the quadrotor platform. The quadrotor structure and the high-level architecture
Chapter 3: "Quadrotor Model" : This chapter derives the mathematical non-linear model of the X-Pro quadrotor.
Chapter 4: "*Quadrotor Control*" : Novel control algorithms will be represented in this chapter. As well as, a comparison between the proposed algorithm with the PID and the LQR controllers will be expressed and evaluated.
Chapter 5: "*Test and Validation*" : Mainly, this chapter consists of two sections: the software simulation, and the real implementation test. An intensive simulation and test results will be depicted to validate the proposed control algorithms in chapter 4.
Chapter 6: "*Closure*" : Finally, this chapter concludes upon the work carried out in this master's thesis.

2 Quadrotor Platform

The X-Pro quadrotor from draganfly is used in this project. The structure of the X-Pro quadrotor has been modified to meet the project requirements. The four arms with actuator and the bottom carbon fiber sheet are reused. In addition, two level boards are built to carry the power drive circuit and the sensors.

The main parts of the X-Pro quadrotor platform are shown in figure 1. [2]

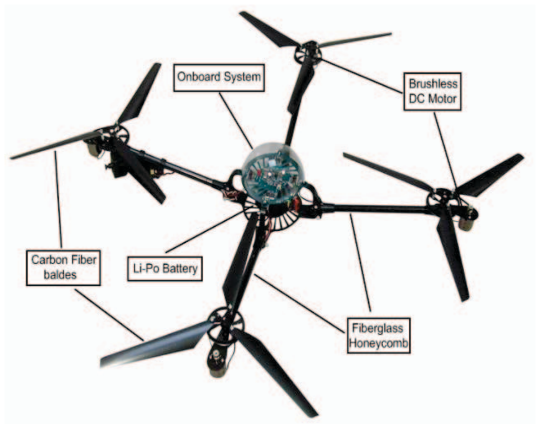

Figure 1 X-Pro quadrotor Platform Structure

On the other hand, the high-level architecture diagram is represented in a connection chart, figure 2. Mainly, the architecture depicts the connections between the ground and the onboard systems.

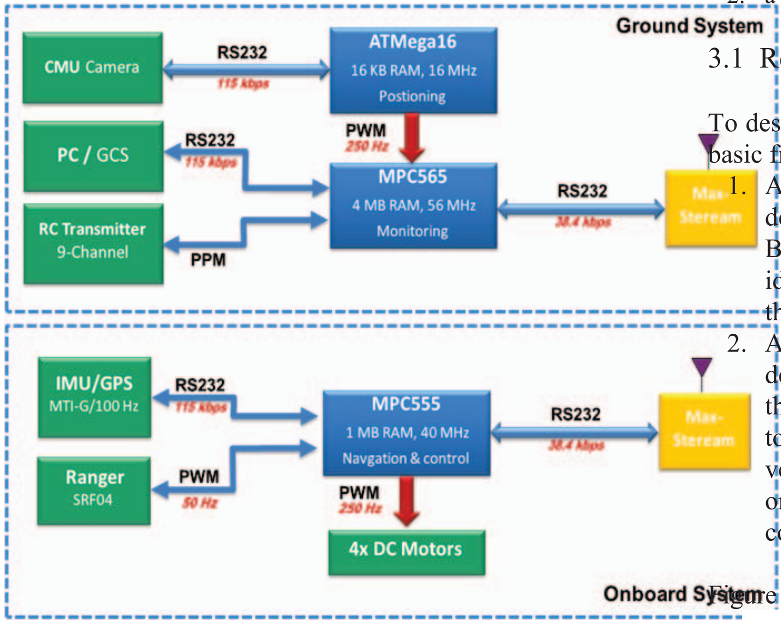

Figure 2 High-Level Architecture Diagram

The onboard system is responsible of controlling the quadrotor autonomously. As shown in figure 2, The MPC555 microcontroller is the brain of the onboard system. The MPC555 collects the data from the sensor devices (IMU/GPS (MTi-G), Ultrasonic sensors (SFR04) and wireless device) to command the thrust of the four actuators. The MPC555 is programmed with an algorithm process to perform the stabilization and the navigation of the quadrotor. In this project three control algorithms are represented, the classical PID, LQR, and AIBC.

The ground system is a crucial part in this project. It gives the functionality to monitor and command the quadrotor during the flight operations. Mainly, the ground system is responsible of:

- Providing the quadrotor with the navigation data (position) from a fixed camera on the ground.
- Changing the flight modes using a Manual Override System.
- Controlling the quadrotor manually or semi-automatically.
- Setting the controller parameters during the flight.
- Monitoring the quadrotor data (e.g. navigation data).

3 Quadrotor Model

The complete mathematical description will be the joint to simulate and design a controller for the quadrotor vehicle. To precisely describe the position and the orientation of the quadrotor, the following is needed:

1. a description of the various reference frames and coordinate systems
2. a transformations between the coordinate systems

3.1 Reference Frames

To describe the equation of motions of a quadrotor, a set of basic frames and notations are defined:

1. A frame with constant inertia of the quadrotor will be designated as a Body Frame (BF). The basis vector of BF will be denoted by (x^b, y^b, z^b). The BF will identify the orientation of the quadrotor according to the right-hand rule at the Centre of Gravity (CG).
2. A fixed frame (not accelerated frame) is required to describe the position and the translational motion of the quadrotor body. This inertial frame will be referred to the earth frame (EF) with the basis vector (x^e, y^e, z^e). The EF could be freely positioned on the earth surface, which is non-accelerated compared to the quadrotor.

Figure 3, shows the reference coordinate systems:

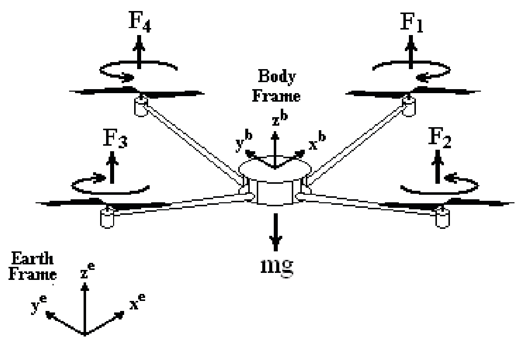

Figure 3 Reference Coordinate Systems

3.2 Transformation

3.2.1 Rotation Matrix

The rotation matrix can be found using Euler angles with rotation arranged in 3-2-1 order. In this sequence, the order of rotation begins with z-axis then y-axis and lastly x-axis, using the right-handed rotation of the coordinate system. [3]

$$R_b^e = \begin{bmatrix} c(\theta)\,c(\psi) & c(\psi)\,s(\theta)\,s(\phi) - s(\psi)\,c(\phi) & c(\psi)\,s(\theta)\,c(\phi) + s(\psi)\,s(\phi) \\ c(\theta)\,s\,(\psi) & s(\psi)\,s(\theta)\,s(\phi) + c(\psi)\,c(\phi) & s(\psi)\,s(\theta)\,c(\phi) - c(\psi)\,s(\phi) \\ -s(\theta) & c(\theta)\,s(\phi) & c(\theta)\,c(\phi) \end{bmatrix} \quad (1)$$

Where; R_b^e is the rotation matrix, *c* is the cosine and *s* is the sine, and $\Theta = [\phi\,,\theta\,,\psi]$ are the Euler angles: roll, pitch, and yaw respectively.

The functionality of the rotation matrix can be utilized by expressing the position vector (X^b) in BF to its equivalent vector (X^e) in EF:

$$X^e = R_b^e \,.\, X^b \quad (2)$$

3.2.2 Euler Rates

$\omega^b = [p \;\; q \;\; r]^T$ are the body angular rates of quadrotor about the BF axes. The relationship between the body angular rates and the Euler rates $\dot{\Theta} = [\dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi}]^T$ is complicated by the fact that they are from different coordinate systems. Equation (3) represents the relation between them:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan(\theta)\sin(\phi) & \tan(\theta)\cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sec(\theta)\sin(\phi) & \sec(\theta)\cos(\phi) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (3)$$

3.3 Rigid Body Dynamics and Kinematics

The rigid body model of the quadrotor will be derived based on Newton-Euler formalism. Accordingly, a superior representation of the internal and external forces on the quadrotor structure makes the model more realistic.

3.3.1 Force Dynamics and Kinematics

In this model we will include the thrust and the gravity forces. The dynamics of the translational motions is well-versed in [1].
Equation (4) shows the translational equations of motion.

$$\begin{bmatrix} \ddot{x}^e \\ \ddot{y}^e \\ \ddot{z}^e \end{bmatrix} = \frac{1}{m} \begin{bmatrix} (\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi))\,.\,\sum_{i=1}^{4}(f_{lift,i}) \\ (\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi))\,.\,\sum_{i=1}^{4}(f_{lift,i}) \\ (\cos(\theta)\cos(\phi)) - mg \end{bmatrix} \quad (4)$$

From the kinematics of the translational equations of motion, the velocity and the position can be described by integrating the linear acceleration in (4)

$$\vec{v}^e = \int \vec{a}^e$$
$$\vec{X}^e = \int \vec{v}^e \quad (5)$$

Where;
$\vec{v}^e = [\dot{x}^e, \dot{y}^e, \dot{z}^e]$ & $\vec{X}^e = [x^e, y^e, z^e]$

For controlling purposes, the translational equations of motion are derived with respect to the EF frame. Where, the navigation sensory devices provide the measurements in the inertial frame, which is here the EF.

3.3.2 Moment Dynamics and Kinematics

The derivation of the rotational equations of motion is well described in [1].

$$I_{xx}\,\dot{p} = \tau^b_{lift,x} + (I_{yy} - I_{zz})\,q\,r$$

$$I_{yy}\,\dot{q} = \tau^b_{lift,y} + (I_{zz} - I_{xx})\,p\,r$$

$$I_{zz}\,\dot{r} = \tau^b_{drag} + (I_{xx} - I_{yy})\,p\,q \quad (6)$$

Where, $\dot{\omega}^b = [\dot{p} \;\; \dot{q} \;\; \dot{r}]^T$ is the angular acceleration of the Euler angles.
The Euler angle can be found by integrating the angular acceleration as follows:

$$\omega^b = \int \dot{\omega}^b$$

$$\dot{\Theta} = T.\,\omega^b$$

$$\Theta = \int \dot{\Theta} \quad (7)$$

3.3.3 Equations of Motion

The equations of motion for the translational and rotational subsystems are derived in equations (4) and (7). The state space representation of the nonlinear model could be written as follows:

$$\dot{X} = f(x, u) \tag{8}$$

$$\begin{bmatrix} \ddot{x}^e \\ \ddot{y}^e \\ \ddot{z}^e \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} {}^1/_m\left(\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi)\right).\sum_{i=1}^{4}(f_{lift,i}) \\ {}^1/_m\left(\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi)\right).\sum_{i=1}^{4}(f_{lift,i}) \\ \sum_{i=1}^{4}(f_{lift,i})/_m\left(\cos(\theta)\cos(\phi)\right) - g \\ p + \sin(\phi)\tan(\theta)\, q + \cos(\phi)\tan(\theta)\, r \\ \cos(\phi)\, q - \sin(\phi)\, r \\ (\sin(\phi)/\cos(\theta))\, q + (\cos(\phi)/\cos(\theta))\, r \\ \left(\tau^b_{lift,x} + (I_{yy} - I_{zz})\, q\, r\right)/I_{xx} \\ \left(\tau^b_{lift,y} + (I_{zz} - I_{xx})\, p\, r\right)/I_{yy} \\ \left(\tau^b_{drag} + (I_{xx} - I_{yy})\, p\, q\right)/I_{zz} \end{bmatrix}$$

By defining the control inputs as follows:

$$U_1 = \sum_{i=1}^{4}(f_{lift,i})$$

$$U_2 = \tau^b_{lift,x}$$

$$U_3 = \tau^b_{lift,y}$$
$$U_4 = \tau^b_{drag} \tag{9}$$

Then the state space representation will be as follows:

$$\begin{bmatrix} \ddot{x}^e \\ \ddot{y}^e \\ \ddot{z}^e \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} {}^1/_m\left(\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi)\right).U_1 \\ {}^1/_m\left(\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi)\right).U_1 \\ {}^1/_m\left((\cos(\theta)\cos(\phi)) - g\right).U_1 \\ p + \sin(\phi)\tan(\theta)\, q + \cos(\phi)\tan(\theta)\, r \\ \cos(\phi)\, q - \sin(\phi)\, r \\ (\sin(\phi)/\cos(\theta))\, q + (\cos(\phi)/\cos(\theta))\, r \\ \left(U_2 + (I_{yy} - I_{zz})\, q\, r\right)/I_{xx} \\ \left(U_3 + (I_{zz} - I_{xx})\, p\, r\right)/I_{yy} \\ \left(U_4 + (I_{xx} - I_{yy})\, p\, q\right)/I_{zz} \end{bmatrix} \tag{10}$$

4 Quadrotor Control

Three control approaches will be represented: classical PID, LQR, and our proposed nonlinear controller, AIBC.

4.1 Linear Control

4.1.1 PID Controller

The PID controller is considered to be the most basic controller that is commonly used in different applications. In this paper, the PID controller is designed for attitude and altitude stabilization. Where the thrust, roll, pitch, and yaw are controlled independently by using different PID loops. Figure 4 depicts the feedback loops of the attitude and altitude states.

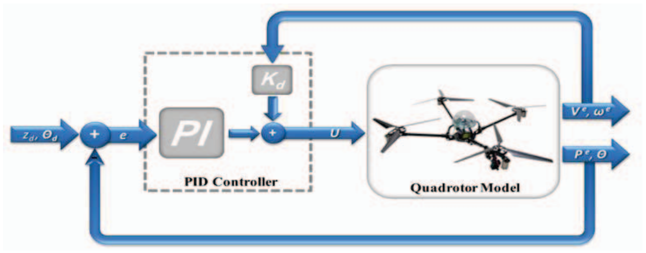

Figure 4 Block diagram of the Attitude and Altitude PID controllers.

The controller does not take the model into account and as such can only be seen as a preliminary controller. Hence, finding the PID gains is performed using one of the heuristic tuning methods, such as Ziegler-Nicholas technique, and/or Tyreus & Luyblen method. The Ziegler-Nicholas tuning method is designed to provide a good system response to the load disturbances. Since the primary design was to reduce the load disturbances, it is necessary to choose a set of gains that gives satisfactory results for our application. Tyreus & Luyblen (TLC) retuned the classical Ziegler-Nicholas method by introducing new rules for determining the PID controller gains, which are called the TLC tuning rules. These values tend to reduce oscillatory effects and improve robustness. Table 1 shows the PID controllers gains that are used after performing the TLC retuning method.

Table 1 Controllers parameters for attitude and altitude control using TLC method

Controller Parameters	*Roll/Pitch Controller*	*Yaw Controller*	*Altitude Controller*
K_p	9.999	5.454	9.090
K_i	5.737	3.482	2.070
K_d	1.304	0.639	2.985

More details about the procedures of performing the Ziegler-Nicholas and TLC methods are described in [1].

4.1.2 Integral LQR Controller

The LQR control algorithm is an optimal control that is concerned with operating a dynamic system at minimum cost. The cost of the system that represented by a linear differential equations is described by a quadratic functional. The quadratic cost function defined as:

$$J = \int_0^\infty (x^T Q\, x + u^T R\, u)\, dt \qquad (11)$$

The feedback control law that minimizes the value of the cost is

$$u = -Kx \qquad (12)$$

where K is given by

$$K = R^{-1} B^T P \qquad (13)$$

And P is found by solving the continuous time differential Riccati equation.

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \qquad (14)$$

The LQR control can be applied as a trajectory follower to minimize the small errors between the reference state x_d and the measured state x, such that the applied control Input becomes:

$$u = -K(x - x_d) \qquad (15)$$

<u>Bryson's rule:</u> matrices Q and R is given by the Bryson's rule. Where Q and R selected to be diagonal matrices:

$$Q_{ii} = \frac{1}{maximmum\ acceptable\ value\ x_i^2} \quad , \quad i \in \{1,2,\dots,l\}$$

$$R_{jj} = \frac{1}{maximmum\ acceptable\ value\ u_i^2} \quad , \quad j \in \{1,2,\dots,m\}$$
(16)

This corresponds to the following criteria

$$J = \int_0^\infty \left(\sum_{i=1}^{l} Q_{ii}\, x_i^2 + \rho \sum_{j=1}^{m} R_{jj}\, u_j^2\right) dt$$
(17)

For attitude stabilization, the linearization version of the equations of motion in (10) for the rotational subsystem is:

$$\dot{\phi} = U_2 / I_{xx}$$

$$\dot{\theta} = U_3 / I_{yy}$$

$$\dot{\psi} = U_4 / I_{zz} \qquad (18)$$

Bryson's rule specifies the weight of the Q and R matrices. The gain matrix K is obtained by choosing the Q and R matrices, so that the system performance are chosen appropriately (e.g. <0.5sec settling time, and 10% overshoot). This is done by using the LQR Matlab toolbox.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

$$K = \begin{bmatrix} 10 & 2.4698 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 2.4698 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 2.3874 \end{bmatrix}$$
(19)

The substitution of the gain matrix K on the equation (15) leads us to conclude that the LQR controller is a PD controller. In addition, the integral part is added to the LQR control to reject the disturbances and the model uncertainties. Consequently, the integral LQR control was another choice to find the PID control gains.

4.2 Nonlinear Control, AIBC

The recursive Lyapunov methodology in the backstepping technique will ensure the system stability, the integral action will increase the system robustness against disturbances and model uncertainties, and the adaptation law will estimate the modeling errors caused by assumptions in simplifying the complexity of the quadrotor model.
This methodology has been proposed to overcome these challenges:

1. Stabilize the attitude and the altitude of the quadrotor
2. Control the nonlinear model of the quadrotor by ensuring the stability based on Lyapunov criterion.
3. Use the adaptation scheme to estimate the translation matrix between the Euler angles rates and the inertial orientation angles rates.
4. Enhance the stabilization performance and compare it with the PID and LQR controllers.

First and foremost, a second order system will be analyzed:

$$\ddot{y} = a + bu + A \tag{20}$$

Where, y is the state variable, a represents the body gyroscopic effect, b is constant and represents the inertia, u is the control input, and A is an estimator for modeling errors and variations.
The design starts with the definition of the position tracking error and its derivative as follows:

$$e_1 = y_d - y_x \tag{21}$$

$$\frac{de_1}{dt} = \dot{y}_d - \dot{y}_x \tag{22}$$

The subscripts in y (into y_x) is for differentiating between the derivative of the desired value $\dot{y}_d$, and the desired virtual input $\dot{y}_{xd}$ as will be shown later.
Explicitly, this definition specifies our control objective, where the recursive methodology will systematically drive the tracking error to zero.
Now, let us consider the Lyapunov function V, which is positive definite around the desired position:

$$V = \frac{1}{2} e_1^2 \tag{23}$$

$$\dot{V} = e_1(\dot{y}_d - \dot{y}_x) \tag{24}$$

If the velocity $\dot{y}_x$ was our control input, it would be straightforward to choose $\dot{y}_x$ to have exponential convergence for the system. One example is:

$$\dot{y}_x = \dot{y}_d + c_1 e_1 \tag{25}$$

Where c_1 is positive number that determines the convergence speed of the error. Thus the derivative of the Lyapunov function will be semi-negative definite and consequently the error will converge exponentially to zero.

$$(\dot{V} = -c_1 e_1^2 \leq 0)$$

Since the velocity $\dot{y}_x$ is only a system variable and not a control input, we can't specify its value as we did in (24). However, we can still choose the desired behavior for $\dot{y}_x$ and consider it as our "virtual" control input. In the backstepping design, this desirable dynamic behavior is called the stabilizing function.
Furthermore, the integral action will be introduced by choosing the virtual input as follows:

$$\dot{y}_{xd} = c_1 e_1 + \dot{y}_d + \lambda_1 \mathcal{X}_1 \tag{26}$$

$$\mathcal{X}_1 = \int_0^t e_1(\tau)\, d\tau \tag{27}$$

The integral action in the backstepping design is to ensure the convergence of the tracking error to zero at the steady state, despite the presence of the disturbances and model uncertainties.
Since, $\dot{y}_x$ is not our control input, there exists a dynamic error between it and its desired behavior $\dot{y}_{xd}$. Therefore, we will compensate this dynamic error through our next design step by defining the velocity tracking error and its derivative as follows:

$$e_2 = \dot{y}_{xd} - \dot{y}_x \tag{28}$$

$$\frac{de_2}{dt} = c_1(\dot{y}_d - \dot{y}_x) + \ddot{y}_d + \lambda_1 e_1 - \ddot{y}_x \tag{29}$$

We can rewrite the derivative of the position error:

$$\frac{de_1}{dt} = -c_1 e_1 - \lambda_1 \mathcal{X}_1 + e_2 \qquad (30)$$

Therefore,

$$\frac{de_2}{dt} = c_1(\dot{y}_d - \dot{y}_x) + \ddot{y}_d + \lambda_1 e_1 - a - bu - A$$

(31)

The Integral Action: By adding the integral action, the augmented Lyapunov function will be:

$$V = \frac{\lambda_1}{2}\mathcal{X}_1^2 + \frac{1}{2}e_1^2 + \frac{1}{2}e_2^2 \qquad (32)$$

The derivative will satisfy ($\dot{V} = -c_1 e_1^2 - c_1 e_2^2 \leq 0$), when the control input is:

$$u = \frac{1}{b}[(1 - c_1^2 + \lambda_1)e_1 + (c_1 + c_2)e_2 - c_1\lambda_1\mathcal{X}_1 + \ddot{y}_d - a - A]$$

(33)

Adaptation Scheme: Since we do not know the real value of A, we will replace it with the estimated value $\hat{A}$ and the adaptation law will be designed:

$$u = \frac{1}{b}[(1 - c_1^2 + \lambda_1)e_1 + (c_1 + c_2)e_2 - c_1\lambda_1\mathcal{X}_1 + \ddot{y}_d - a - \hat{A}]$$

(34)

The design is not complete since we still need to derive the update law for the estimated parameter $\hat{A}$. For this purpose, we will define the parameter estimation error signal in (34) as follows:

$$\tilde{A} = \hat{A} - A \qquad (35)$$

After this definition, by substituting (35) into (31) the derivative of the velocity tracking error is:

$$\frac{de_2}{dt} = -c_2 e_2 - e_1 + \tilde{A} \qquad (36)$$

The Lyapunov function is used a second time to augment the estimated parameter error:

$$V = \frac{\lambda_1}{2}\mathcal{X}_1^2 + \frac{1}{2}e_1^2 + \frac{1}{2}e_2^2 + \frac{1}{2\gamma}\tilde{A}^2 \qquad (37)$$

$$\dot{V} = \lambda_1\mathcal{X}_1\dot{\mathcal{X}}_1 + e_1\dot{e}_1 + e_2\dot{e}_2 + \frac{\tilde{A}}{\gamma}\frac{d\tilde{A}}{dt} \qquad (38)$$

$$\dot{V} = -c_1 e_1^2 - c_2 e_2^2 + \tilde{A}\left\{e_2 + \frac{1}{\gamma}\frac{d\tilde{A}}{dt}\right\} \qquad (39)$$

Where γ is a positive design constant that determines the convergence speed of the estimate. In order to render the non-positivity of the Lyapunov derivative in the above equation, we will choose the adaptation law for the estimated parameter $\hat{A}$ as:

$$\frac{d\hat{A}}{dt} = -\gamma e_2 \qquad (40)$$

$$\hat{A} = -\gamma \mathcal{X}_2 \qquad (41)$$

Where;

$$\mathcal{X}_2 = \int_0^t e_2(\tau)\, d\tau \qquad (42)$$

Thus,

$$\dot{V} = -c_1 e_1^2 - c_2 e_2^2 \leq 0 \qquad (43)$$

Finally, the control input that contains the adaptation law and the integral action is derived as follows:

$$\boldsymbol{u = \frac{1}{b}[(1 - c_1^2 + \lambda_1)e_1 + (c_1 + c_2)e_2 - c_1\lambda_1\mathcal{X}_1 + \gamma\mathcal{X}_2 + \ddot{y}_d - a]}$$

(44)

By applying the AIB algorithm on each equation of motion, the control inputs are going to be: [4]

$$U_1 = \frac{-m}{\cos(x_1)\cos(x_3)}\left((1 - c_7^2 + \lambda_4)e_7 + (c_7 + c_8)e_8 - c_7\lambda_4\chi_7 + \gamma_4\chi_8 + \ddot{x}_{7d} - g\right)$$

(45)

$$U_2 = \frac{I_{xx}}{l}\left((1 - c_1^2 + \lambda_1)e_1 + (c_1 + c_2)e_2 - c_1\lambda_1\chi_1 + \gamma_1\chi_2 + \ddot{x}_{1d} - \left(\frac{I_{yy} - I_{zz}}{I_{xx}}\right)x_4 x_6\right)$$

$$U_3 = \frac{I_{yy}}{l}\left((1 - c_3^2 + \lambda_2)e_3 + (c_3 + c_4)e_4 - c_3\lambda_2\chi_3 + \gamma_2\chi_4 + \ddot{x}_{3d} - \left(\frac{I_{zz} - I_{xx}}{I_{yy}}\right)x_2 x_6\right)$$

$$U_4 = \frac{I_{zz}}{l}\left((1 - c_5^2 + \lambda_3)e_5 + (c_5 + c_6)e_6 - c_5\lambda_3\chi_5 + \gamma_3\chi_6 + \ddot{x}_{5d} - \left(\frac{I_{xx} - I_{yy}}{I_{zz}}\right)x_2 x_4\right)$$

5 Test and Validation

In this paper, PID and Integral-LQR controllers are simulated and implemented to be the benchmark for AIBC control algorithm.

5.1 Software Simulation

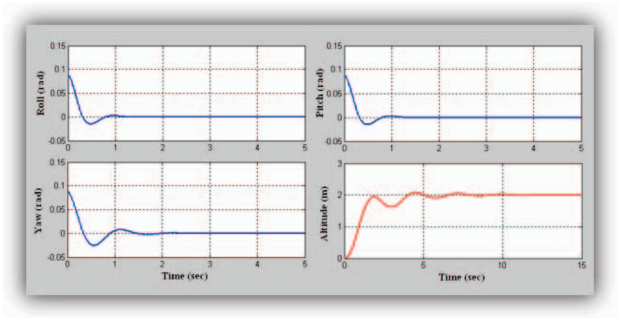

Figure 5 PID Controller System responses using TLC retuning method

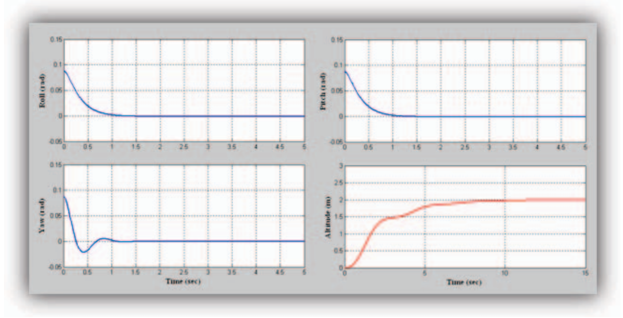

Figure 6 System responses using LQR Control method

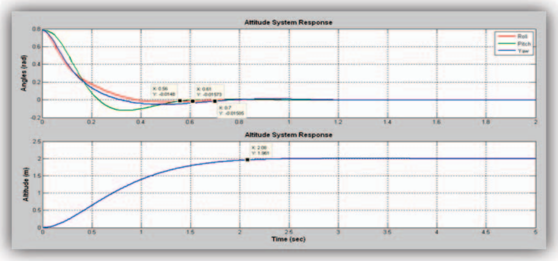

Figure 7 Attitude and Altitude Stabilization of the AIBC

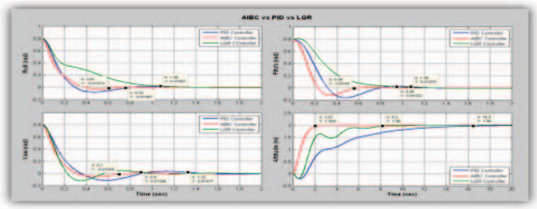

Figure 8 AIBC vs PID vs LQR Controllers

The PID controller gains are tuned after going through two tuning methods: Ziegler-Nichols and TLC methods. In addition, the Integral LQR algorithm considered as another PID tuning method. AIBC shows better performance than the PID and the integral LQR controllers. The time response is faster in the attitude stabilization, and it's not comparable in the altitude control. Table 2 shows the 2% settling time of the three controllers in seconds:

Table 2 2% Settling Time Comparison between AIBC, PID, and LQR Controllers

ariable State	AIBC	PID	Integral-LQR
Roll	0.61	0.76	1.08
Pitch	0.56	1.08	0.95
Yaw	0.7	1.33	0.9
Altitude	2.07	16.5	8.2

The noise rejection and the model error estimations are considered as well in our comparisons. This comparison will be explored in the real time Implementation tests in section.

5.2 Real Time Implementation

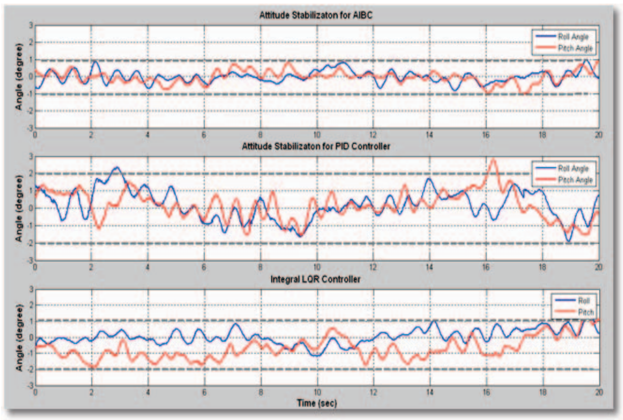

Figure 9 AIBC vs PID vs LQR in Attitude Stabilization (Roll and Pitch Angles)

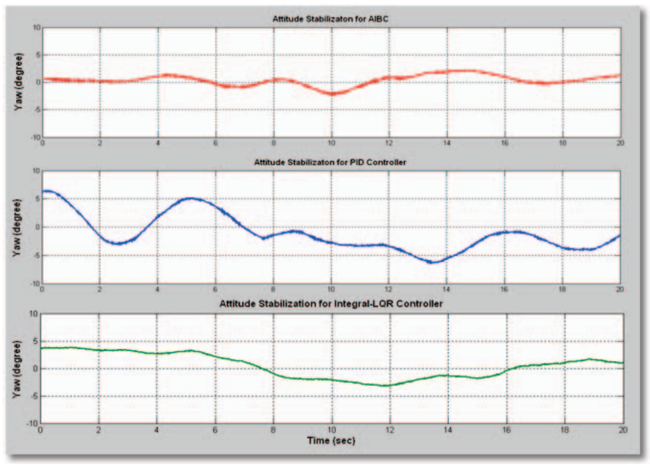

Figure 10 AIBC vs PID vs LQR in Attitude Stabilization (Yaw Angle)

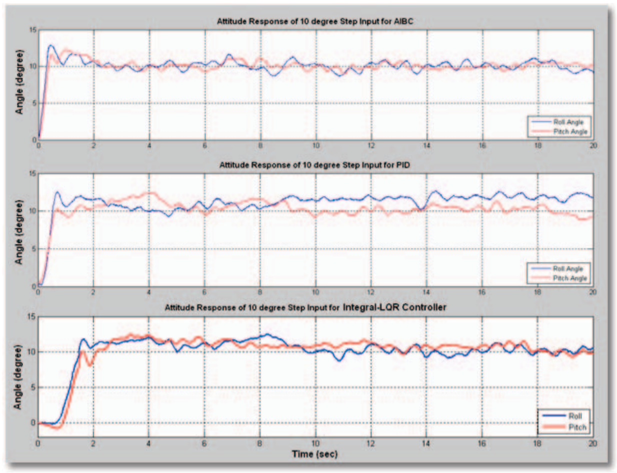

Figure 11 AIBC vs PID vs LQR in Attitude stabilization (10^o step input)

Figure 9 compares the three controllers while maintaining the attitude angles around the origin. For roll and pitch angles, the AIBC shows better stabilization response than the other two controllers. The AIBC stabilizes the angles in a range of less than ±1^o, compared to ±2^o in the PID controller, and ±1.5^o in the Integral-LQR controller. On the other hand, the noise and disturbance rejection are enhanced in the AIBC, where less fluctuation around the origin is depicted in figure 9.
For the yaw angle, figure 10 shows the stabilization comparison as well. Where, the AIBC is performing better than the PID and Integral-LQR in maintaining the yaw angle around the north direction.
On the other hand, a test using 10^o step input is performed. The performance of the time response and the stabilization around the desired value are validated. As shown in figure 11, the AIBC shows faster response and better stabilization among the other controllers. Furthermore, the optimization of the control inputs of the Integral-LQR Controller explains the slower settling time and the smoother stabilization compared to the PID controller.
The effects of considering the nonlinearities in the AIBC control inputs (efforts) are clearly seen from these experiments. The adaptation and integration parts are performing better stabilization results as well.

6. Conclusion

The quadrotor design is different from the conventional helicopter design. It couldn't adjust anything but the rotational velocity of the rotors. Despite of the quadrotor design, the quadrotor is freely moving in 6DOF, like the helicopter.
The goal of this project was to stabilize the attitude and altitude of the quadrotor. To achieve this, the quadrotor was modeled in order to develop and simulate the controller. The model of the quadrotor is a nonlinear model. Therefore, nonlinear control algorithms are proposed to deal with the nonlinearities and the model uncertainties as well.
In this paper, a novel nonlinear control approach is proposed based on a recursive Lyapunov methodology using the backstepping technique. The new approach employs the adaptation scheme to estimate the ignored parts in the dynamic model, and the integral action is used to overcome any disturbances and model uncertainties. Altogether, the Adaptive Integral Backstepping Controller (AIBC) is introduced to control the attitude and the altitude of the quadrotor. Simulation results and test flights had been conducted to validate the control design schemes, and to show that the proposed Adaptation and integral action of the backstepping controller shows better performance in dynamic response.
Finally, all developed algorithms had been tested and validated. Theoretically by showing extensive simulations using Matlab/Simulink Environment, and practically by performing the flight tests to validate our simulation results.

References

[1] Y. Al-Younes, M. Al-Jarrah, and A. Jhemi, "Establishing Autonomous AUS Quadrotor," M. S. thesis, American University of Sharjah, Sharjah, UAE, 2009.

[2] DraganFLY Innovation Inc., "R/C Helicopter," X-Pro Quadrotor, 2003. [Online]. Available: http://www.rctoys.com. [Accessed:

Sep. 2, 2007]

[3] M. F. Pettersen, "Nonlinear Control Approach to Helicopter Autonomy," M. S. thesis, Aalborg university, Aalborg, Denmark, 2005.

[4] Y. M. Alyounes and M. A. Jarrah, "Adaptive Integral Backstepping Controller for an Autonomous Rotorcraft," in *proceeding of the 6th International Conference on Mechatronics and its Applications, ISMA'09*, Sharjah, UAE, 2009.