ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

MODELING, IDENTIFICATION AND SIMULATION OF A QUADROTOR USING REAL-TIME FLIGHT DATA

M.Sc. THESIS

Atakan SARIOĞLU

Department of Mechanical Engineering

System Dynamics and Control Programme

MAY 2015

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

MODELING, IDENTIFICATION AND SIMULATION OF A QUADROTOR USING REAL-TIME FLIGHT DATA

M.Sc. THESIS

Atakan SARIOĞLU
(503121604)

Department of Mechanical Engineering

System Dynamics and Control Programme

Thesis Advisor: Assist. Prof. Ayhan KURAL

MAY 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

BİR DÖRT ROTORLU HAVA ARACININ GERÇEK ZAMANLI UÇUŞ VERİSİ İLE MODELLEMESİ, TANILAMASI VE SİMÜLASYONU

YÜKSEK LİSANS TEZİ

Atakan SARIOĞLU
(503121604)

Makine Mühendisliği Anabilim Dalı

System Dinamiği ve Kontrol Programı

Tez Danışmanı: Y. Doç. Dr. Ayhan KURAL

MAYIS 2015

Atakan SARIOĞLU, a **M.Sc.** student of ITU **Graduate School of Science, Engineering and Technology** student ID **503121604**, successfully defended the **thesis** entitled "**MODELING, IDENTIFICATION AND SIMULATION OF A QUADROTOR USING REAL-TIME FLIGHT DATA**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assist. Prof. Ayhan KURAL** İstanbul Technical University	.....
Jury Members :	**Prof. Dr. Recep KOZAN** Sakarya University	.....
	Assoc. Prof. Kenan Refah KUTLU İstanbul Technical University	.....

Date of Submission : 04 May 2015
Date of Defense : 29 May 2015

FOREWORD

This master's thesis is written for the completion of System Dynamics and Control programme in Istanbul Technical University. Without ambition and patience, it could not be succeeded.

May 2015

Atakan SARIOĞLU
(Electronics Engineer)

TABLE OF CONTENTS

Page

ABBREVIATIONS

DOF	**:** Degrees of freedom
IMU	**:** Inertial measurement unit
LiPo	**:** Lithium polymer
DC	**:** Direct current
PEM	**:** Prediction error method (or prediction error minimization)
ARX	**:** Auto regressive exogenous
OE	**:** Output error
PID	**:** Proportional, integral, derivative
UAV	**:** Unmanned aerial vehicle
Hz	**:** Hertz
kbps	**:** Kilobits per second
kBps	**:** Kilobytes per second
RMS	**:** Root mean square

LIST OF TABLES

Page

LIST OF FIGURES

Page

MODELING, IDENTIFICATION AND SIMULATION OF A QUADROTOR USING REAL-TIME FLIGHT DATA

SUMMARY

Quadrotor is the name given to a special kind of aircraft of which thrust is generated by four independent propellers. Although some manned quadrotors exist, most of the quadrotors are unmanned aerial vehicles (UAV). This work is about mini unmanned quadrotors so called quadrotor mAV.

Today, quadrotors are very popular for researcher for their great ability to hover and vertical takeoff and landing (VTOL). Especially for defense industry applications, rescue and exploration purposes they become a matter the choice.

In this work, operation principles of are given in details from an engineering point of view. A detailed inertial frame based nonlinear dynamical model for plus-type quadrotor is obtained using Euler Lagrange method. It contains gyroscopic torques as well as Coriolis effects. After that, a body frame based nonlinear dynamical model is obtained using Newton Euler method. The rotational part of the Newton Euler based model is linearized to obtain a simpler model to use for the rest of this work. A motor (driving a propeller) is also modeled as a nonlinear dynamical system.

For experimental parts of this work, Crazyflie quadrotor platform is used. Crazyflie is a mini sized quadrotor that can be flown by a human operator via Crazyradio wireless communication. To be able to collect real-time flight data, a telemetry system is implemented as an add-on for Crazyflie quadrotor. Both the software and the hardware are explained in details.

Real-time motor speed versus motor armature voltage data is collected. A grey box model is defined and the parameters of the model are identified using prediction error method (PEM).

Real-time flight data including attitude angles of the quadrotor versus motor armature voltages is collected and the voltages are converted to motor speeds using identified motor model. Grey-box models are defined for three Euler angles and their parameters are found using PEM. Then as a black-box approach, auto regressive exogenous (ARX) technique is used to identify linear models.

Finally, a simulation scheme using the identified model is designed and model verification is done. A proportional, integral, derivative (PID) based controllers are designed and the quadrotor simulation model is controlled with success.

BİR DÖRT ROTORLU HAVA ARACININ
GERÇEK ZAMANLI UÇUŞ VERİSİ İLE
MODELLEMESİ, TANILAMASI VE SİMÜLASYONU

ÖZET

Dört rotorlu hava aracı, yani quadrotor, itkinin dört bağımsız pervane tarafından oluşturulduğu özel bir hava aracı türüne verilen addır. Temelde rotorlu hava aracı yani helikopter sınıfında yer almaktadırlar. İnsansız türlerini uçan robotlar olarak da adlandırmak mümkündür.

Kullanım şekline göre temel olarak ikiye ayrılır. İnsan kontrollü ve kendinden kontrollü türleri vardır. İnsan kontrollü olan quadrotorlar aracın içindeki bir sürücü tarafından veya uzaktan kontrol edilebilir.

Uzaktan bir insan tarafından kontrol edilen veya kendinden otomatik olarak kontrol edilen quadrotorlar insansız hava aracı sınıfına (İHA) girer. Üzerinde insan taşıyan ve o şekilde kontrol edilen şekilleri de mevcut olmasına rağmen genellikle insansız türleri daha yaygındır.

Quadrotorlar günümüzde savunma sanayi başta olmak üzere çok yaygın kullanım alanlarına sahiptir. Genellikle uzaktan insan pilotlar tarafından uçurulurlar. Fakat tekniğin geldiği son noktada bazı kendinden kontrollü şekilleri de geliştirilmiştir.

Kullanım alanlarına örnek vermek istersek, insanların ulaşmasının zor ya da tehlikeli olduğu alanlar; mağaralar, yıkıntılar, bina enkazları, radyoaktif veya kimyasal madde tehlikesi olan alanlar olabileceği gibi, ulaşmanın güç olduğu yüksek yerler, derin çukurlar ve buna benzer yerler için idealdir.

Başlıca quadrotor kullanım alanlarında biri de fotoğrafçılık ve haritacılıktır. Havadan keşif ya da ölçüm amaçlı fotoğraf çekme imkânı sunan quadrotorlar sayesinde bu alanda büyük kolaylık sağlanmıştır.

Örneğin 2011 yılında Japonya'daki Fukuşima nükleer güç reaktöründeki radyoaktif sızıntı ve patlamalar sonrasında, insan sağlığına zararlı durumlara sebebiyet vermemek amacıyla insansız hava araçları ve robotlar ile alan incelenmiş, uzaktan bilgi toplanmış, fotoğraflar çekilmiştir.

Quadrotorların savunma sanayinde de yaygın kullanımı vardır. Birçok büyük ülke ordusu son yıllarda bu araçlardan çok sayıda alarak ya da üreterek bünyesine katmıştır.

Savunma sanayindeki kullanım amaçlarından bahsetmek istersek, güvenlik amaçlı olarak havadan fotoğraf ve video çekme özelliği ile ön plana çıkmaktadırlar. Hudut güvenliğinde insanlarla gözlemlenmesi zor olan geniş alanlar, bu araçlar sayesinde sürekli takip edilebilmektedir.

Son yıllarda kullanımları o kadar yaygınlaşmıştır ki, Amazon internetten alışveriş şirketi müşterilerine siparişlerini bu araçlarla havadan ulaştırarak bu alanda bir ilke imza atmıştır.

Bu çalışmada mini quadrotor sınıfında yer alan Crazyflie quadrotorun modellenmesi, gerçek zamanlı uçuş verileri kullanılarak model parametrelerinin kestirimi, son olarak da bu modelin bilgisayarda benzetimi yer almaktadır.

1. bölümde quadrotor hava araçları tanıtılmış, kullanım alanları, tarihçesi, çeşitleri anlatılmıştır. Daha sonra kapsamlı bir literatür taraması yapılarak, quadrotor tasarımı, modellemesi, sistem tanılaması ve kontrolü alanındaki öne çıkan çalışmalar incelenmiştir.

2. bölümde quadrotor hava aracının yapısı ve çalışma mantığı anlatılmış, hareket çeşitleri ve nasıl yönetildikleri incelenmiştir.

3. bölüm quadrotor modellemesi üzerinedir. Quadrotor 6 serbestlik derecesine sahip bir araçtır. Ayrıca havada hareket ettiğinden doğrusal olmayan ve yüksek mertebeden bir dinamiğe sahiptir.

Literatürde quadrotor modelleri temel olarak ikiye ayrılmaktadır; Newton Euler tabanlı modeller ile Euler Lagrange temelli modeller. Bunlara ek olarak bazı kaynaklarda yer alan Quaternion tabanlı modeller de mevcuttur. Saydığımız tüm modeller doğrusal olmayan modellerdir.

Bu çalışmada önce aerodinamik etkileri ve jiroskop momentlerini içeren Euler Lagrange tabanlı, kapsamlı bir model yer almaktadır. Yer koordinat ekseni tabanlı olan bu model ötelemeli ve döngüsel model olarak iki alt modelde incelenmiştir. Ötelemeli kısım sade olmasına karşın döngüsel kısım oldukça uzun ve karmaşık denklemler içermesi sebebiyle kullanımı bazı varsayımlar olmadan zordur.

Bu çalışmadaki bir diğer model ise Newton Euler tabanlı daha basit fakat yine doğrusal olmayan bir modeldir. Bu model aslında gövde koordinat eksenine göre elde edilmesine rağmen bazı gerçekçi kabuller sayesinde yer koordinat sistemine aktarılmıştır.

Her iki model de karmaşık bulunmuştur. Bu sebeple modelin ötelemeli kısmı olduğu gibi bırakılırken, döngüsel kısmı doğrusallaştırılmış, böylece çalışmanın geri kalanında kullanılacak olan basit bir quadrotor dinamik modeli elde edilmiştir.

Quadrotor üzerinde ayrıca dört adet motor-pervane çifti yer almaktadır. Dinamik modelin gerçeğe yakın olabilmesi motor dinamik modelinin doğruluğundan geçmektedir. Bu sebeple elektrik motorları ve pervane yükü doğrusal olmayan bir dinamik denklem olarak ifade edilmiştir.

Tüm model, 4 adet motor modeli ve quadrotor dinamik modeli birleştirilerek elde edilmektedir.

4. bölümde deneysel çalışmalarda kullanılacak olan quadrotor donanımının gereksinimleri anlatılarak, bu doğrultuda yapılan çalışmalara yer verilmiştir.

Bu çalışmada Crazyflie ismindeki mini quadrotor altyapısı kullanılmıştır. Crazyflie avuç içine sığacak boyutta, üzerindeki ARM Cortex-M mikroişlemcisi ile yüksek

başarımlı yerleşik kontrolcüye sahip, Crazyradio ismindeki kablosuz haberleşme altyapısı ile bir bilgisayar üzerinden kontrol edilebilen quadrotor platformudur.

Crazyflie quadrotorun yazılım ve donanım yapısı ayrıntılı olarak incelenmiştir.

5. bölümde anlatıldığı gibi, Crazyflie üzerine, uzaktan gerçek-zamanlı veri toplama amaçlı bir telemetri sistemi tasarlanmış ve üretilmiştir. Bu sistem 2.4GHz radyo frekansında çalışarak, quadrotor dinamik sistemine ilişkin pek çok veriyi yüksek örnekleme hızında kumanda bilgisayarına aktarmaktadır.

Bu bölümde telemetri sisteminin yazılımı ve donanımsal yapısı ayrıntılı olarak işlenmiştir.

6. bölümde quadrotor üzerinde kullanılan ivmeölçer ve jiroskop sensörleri anlatılmış, çıkışlarının nasıl işleneceğine değinilmiştir.

7. bölümde ilk olarak toplanan gerçek zamanlı verilerin anlamlı hale getirilmesi yani işlenmesi daha sonra model tanılama ve parametre kestirimi çalışmaları anlatılmıştır.

Öncelikle motor hızının armatür gerilimine göre değişimini temsil eden veriler toplanmış, bu veriler işlendikten sonra gri-kutu olarak verilen sistem modeli üzerindeki parametreler kestirim hatası yöntemi kullanılarak elde edilmiştir.

Sonrasında quadrotordan eğim açılarının motor armatür gerilimlerine göre değişimini temsil eden, gerçek zamanlı veriler, uçuş esnasında toplanmıştır. Motor giriş gerilimleri yine bu bölümde kestirilmiş olan motor modeli sayesinde motor hızlarına çevrilmiştir.

Eğim açılarının motor hızlarına göre değişimini temsil eden veriler kullanılarak, gri kutu yaklaşımı ile verilmiş olan quadrotor modeline ait parametreler, kestirim hatası yöntemi ile kestirilmeye çalışılmıştır.

Sonrasında ARX yaklaşımı ile model tanılama yapılmış, çeşitli model mertebeleri için doğrusal modeller elde edilmiştir.

8. bölümde elde edilen model ile bir benzetim oluşturulmuş ve model doğrulaması yapılmıştır. Daha sonra bir PID kontrolcü ile model test edilmiş, böylece daha sonraki kontrolcü tasarım çalışmaları için bu altyapının kullanılabileceği ispatlanmıştır.

9. bölümde yapılan çalışmalar üzerine yorum yapılmış ve çeşitli öneriler ele alınmıştır.

1. INTRODUCTION

Quadrotor is a vertical takeoff and landing (VTOL) aerial vehicle that has four rotors. There are manned and unmanned quadrotor types. Especially, the unmanned quadrotors gained a big popularity today.

This work is about mathematical modeling, identification and simulation of a quadrotor unmanned aerial vehicle (UAV), based on real-time flight data that is collected from a modified mini-sized quadrotor.

Now, brief information on unmanned aerial vehicles and quadrotors will be given.

1.1 Unmanned Aerial Vehicles

Unmanned aerial vehicles (UAV), so-called drones, are remote piloted or autonomous vehicles with no pilots onboard.

The first versions were built during World War I, generally in missile form. Some of them were radio controlled and some of them have mechanical gyroscopes.

With the rapid improvements in electronic systems, especially the sensor and microcontroller technologies, UAVs become more and more powerful. Today, most of the world armies are using remote controlled drones (Figure 1.1).

Figure 1.1 : MQ-9 Reaper UAV.

1.2 Quadrotors

The first quadrotors appeared in early 20th century and all of them were manned. Louis Breguet designed a four-rotor aerial vehicle, Gyroplane No.1, that is powered by a piston engine in 1907 but the result was not good. In 1920, Etienne Oehmichen built a quadrotor, Oehmichen No.2, which was the first stable and controllable rotorcraft. Oehmichen No.2 (Figure 1.2) completed a test flight successfully.

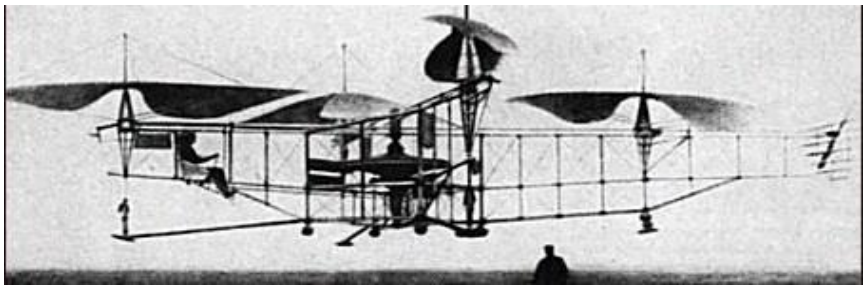
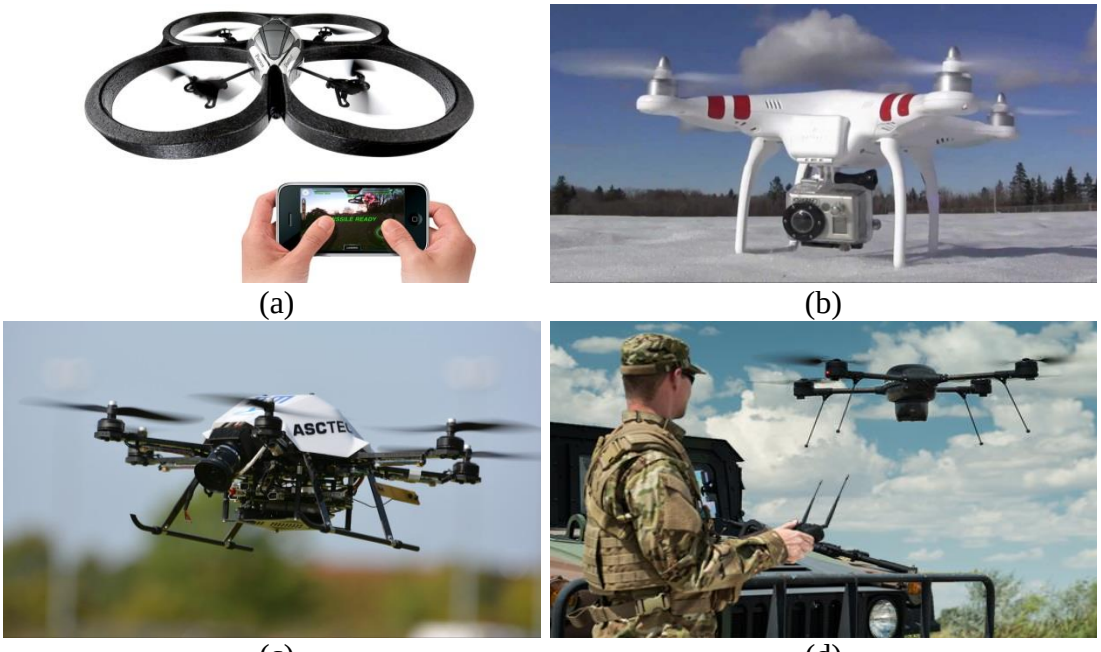


Figure 1.2 : Oehmichen No.2.

In 2000s, the unmanned quadrotors become very popular especially for their ability to hover, which enables aerial photography.

There are too many commercial products available on the market today. Some of the well-known models are; AR. Drone from Parrot, Phantom from DJI, Hummingbird, Pelican and Firefly from Ascending Technologies. In addition some military versions are available i.e. Indago from Lockheed Martin (Figure 1.3).



(a) (b)

(c) (d)

Figure 1.3 : (a) AR.Drone, (b) DJI Phantom, (c) Firefly, (d) Indago.

1.3 State of the Art

Quadrotor UAV's have become very popular to scientists for last few decades. It is possible to say that the first mini quadrotor concept works started to appear in early 2000s and then it turned into a development race. Some recent works will be given.

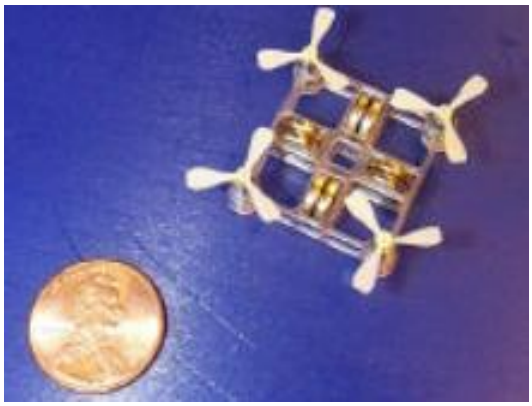


Figure 1.4 : Mesicopter Project.

In 2000, Kroo et al ([1], [2]) from Stanford University has started Mesicopter Project (Figure 1.4) and built a centimeter scale quadrotor. Although the first prototype was controlled by a passive control mechanism, they implemented a visual CMOS camera feedback to control the quadrotor afterwards.

In 2002, Hamel et al. [3] presented a Newton-Euler based, body frame quadrotor model as well as the lift and reactive torque models of the rotors. They also controlled their model using a backstepping controller.

In 2002, Altuğ et al. ([4], [5]) controlled a quadrotor helicopter (Figure 1.5) using a stationary camera to give visual position and orientation feedback. Backstepping and feedback linearization techniques are applied to a cross-type quadrotor model. In 2003, Altuğ et al. also worked with dual camera feedback control.

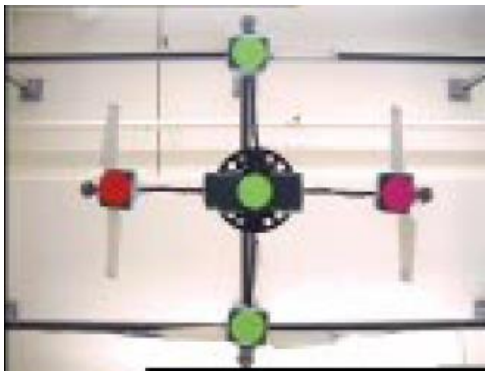


Figure 1.5 : Work of Altuğ et al.

In 2004, Hoffmann et al. [6] from Stanford University developed STARMAC (Stanford Test bed of Autonomous Rotorcraft for Multi-Agent Control) (Figure 1.6) based on Draganflyer III quadrotors. They replaced the onboard electronics of the Draganflyer with a dual PIC microcontroller based controller board, which has also IMU, GPS and ultrasonic range sensor. Then they applied sliding-mode feedback controller.

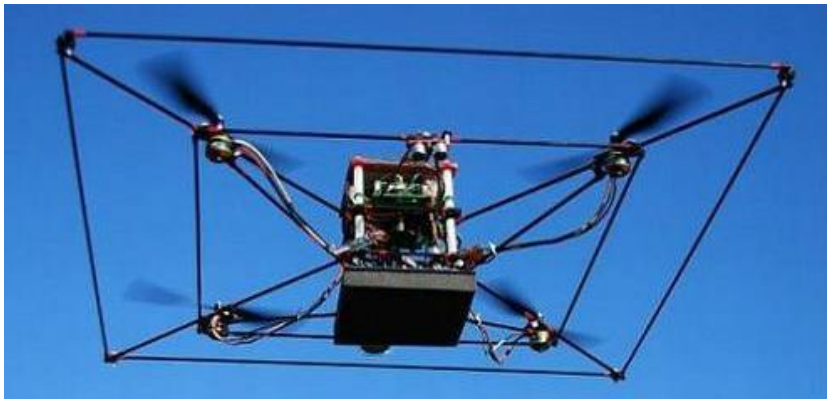


Figure 1.6 : STARMAC from Stanford University.

In 2004, Tayebi et al. [7] presented quaternion based PD^2 feedback controllers to compensate the Coriolis and gyroscopic torques exists on a quadrotor. The work showed that the proposed controller maintains exponential stability.

In 2004, Castillo et al. [8] presented a Lagrange based inertial frame quadrotor model and implemented a Lyapunov based controller then proved global stability of the closed loop system. They also implemented the controller to Draganflyer III (Figure 1.7) quadrotor platform.

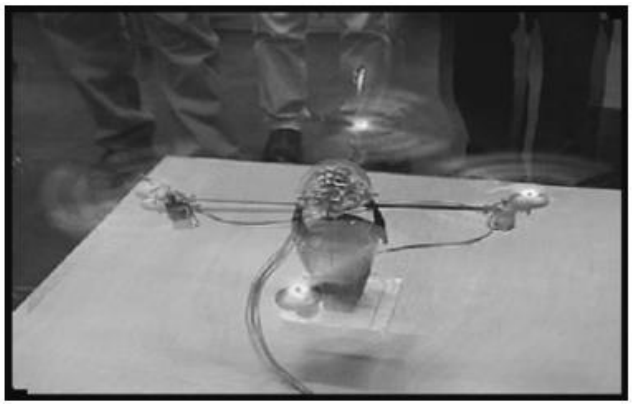


Figure 1.7 : Castillo et al.

In 2004, Mokhtari et al. [9] presented a dynamical model for a quadrotor and evaluated it as decoupled subsystems. They used a feedback linearization based

controller for the rotational subsystem and used an estimator for the translational components because of imprecise measurements. Mokhtari et al. [10] modeled actuator saturation case of a quadrotor and applied a robust, linear $H\infty$ controller in 2005.

In 2004, Bouabdallah et al. [11] presented yet another quadrotor, OS4. A Newton-Euler based dynamical model is evaluated as two subsystems and a Lyapunov based control law is presented. In addition, they tried to implement the controller on a 3 axes locked test bench without success.

In 2005, Bouabdallah et al. [12] equipped their OS4 platform with a single board computer to gain enough processing power and applied backstepping and feedback linearization based control approaches to that platform.

In 2005, Mahony et al. [13] worked on sensor fusion algorithms and presented a nonlinear complementary filter to combine gyroscope and accelerometer sensors that widely used on unmanned quadrotors and other unmanned aircrafts. They also presented on-line bias estimation algorithm for gyroscope data.

In 2006, Pounds et al. [14] developed the X-4 Flyer quadrotor platform (Figure 1.8) and implemented onboard controller to use on it.

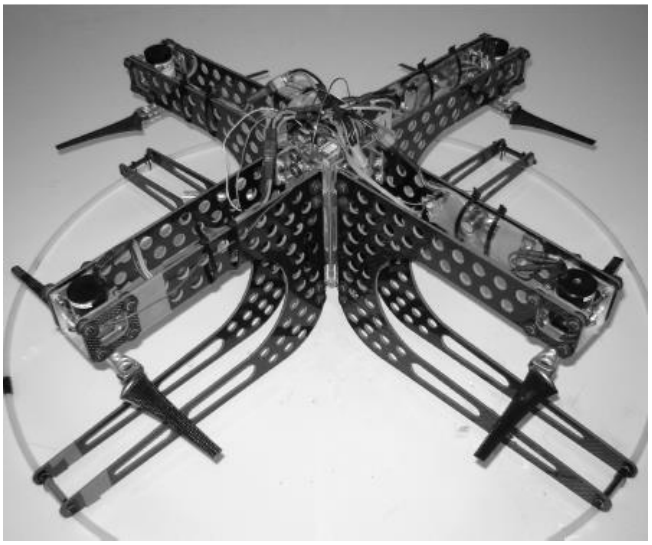
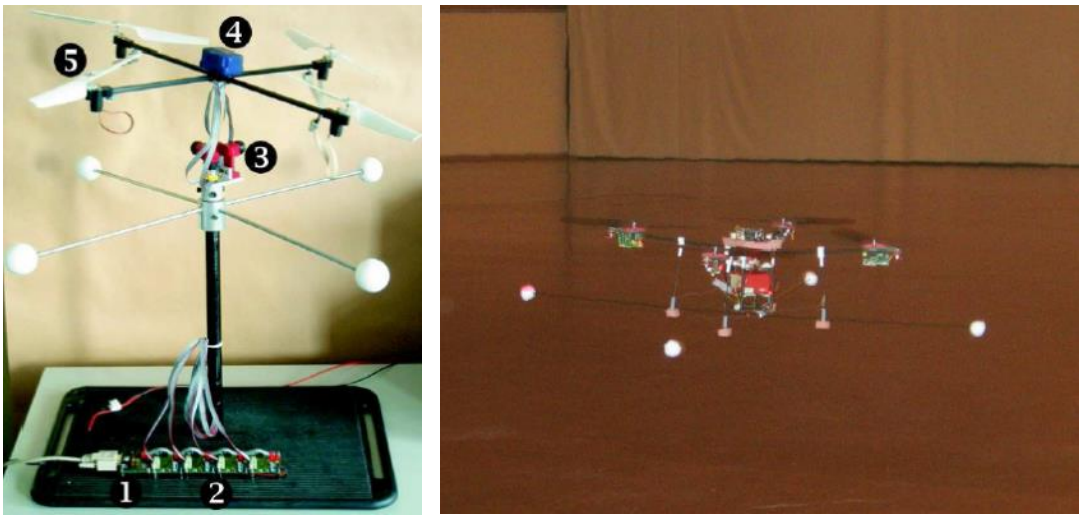


Figure 1.8 : The X-4 Flyer used by Pounds et al.

In 2007, Bouabdallah et al. [15] equipped their OS4 quadrotor (Figure 1.9) with a controller board equipped with a camera system and sonar distance sensors featuring autonomous take-off, hover, landing and collision avoidance. They used integral backstepping based controller.



(a) (b)

Figure 1.9 : Works of Bouabdallah et al. (a) in 2004 (b) in 2007.

In 2008, Bresciani [16] designed a quadrotor platform (Figure 1.10) using Newton-Euler based body frame model. He used different methods to identify the quadrotor constants including propeller lift measurement, rotor and body inertia calculation and motor series resistance. The designed quadrotor was equipped with infrared and sonar distance sensors as well as inertial measurement unit.

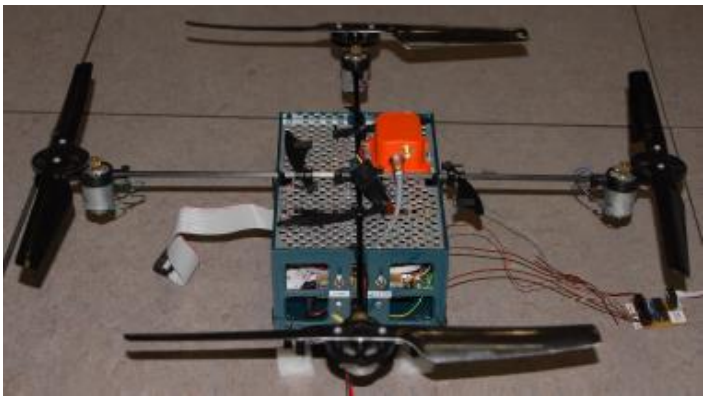


Figure 1.10 : Quadrotor designed by Bresciani.

In 2008, Petersen et al. [17] worked on modeling and identification of Draganflyer X-Pro (Figure 1.11) quadrotor. They presented a cascade quadrotor model that consists of motor dynamics and quadrotor dynamics. To identify the whole system, firstly they identified a linear motor model using ARX estimation approach based on measured motor speed versus PWM input data. Then to estimate the lift and torque constants of the propellers, they placed the quadrotor to a workbench that locks all axes except one, measured generated force versus motor speed and fitted curves for

the unknown parameters. Finally, they implemented a PID controller as well as a pole placement controller.

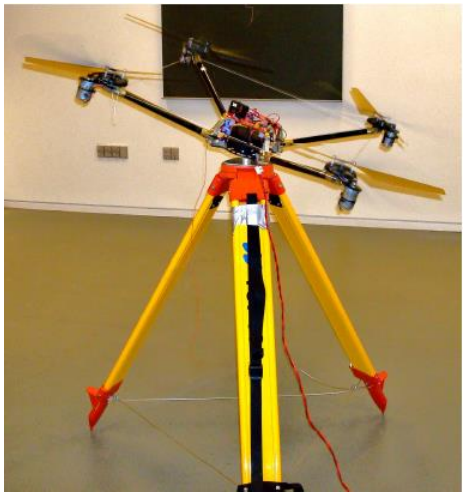


Figure 1.11 : Draganflyer X-Pro used by Petersen et al.

In 2009, Hussein et al. [18] designed a quadrotor and created its full featured CAD model, then calculated the model constants using the CAD software.

In 2010, Luo et al. [19] used Bouabdallah's OS4 Simulator Platform that takes throttle as input and gives attitude and position output. They identified the altitude of the closed-loop simulator using a first order plus time delay ARX model. Finally, they applied different configurations of PID controllers to the identified model and compared the results.

In 2011, Stanculeanu et al. [20] presented yet another quadrotor platform (Figure 1.12) and collected real-time attitude angles versus command input data from the closed-loop system. Then they identified the quadrotor dynamics as a linear state space model using prediction error method (PEM).

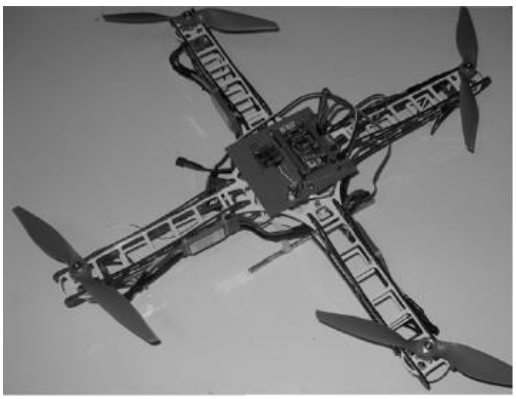


Figure 1.12 : Work of Stanculeanu et al.

In 2012, Falkenberg et al. [21] worked on a quadrotor platform using Newton-Euler based model and calculated the lift and torque constants using blade element momentum theory. Then they fixed the quadrotor axes to a test bench and collected closed-loop input-output data to identify a grey-box quadrotor model. Because of no detailed explanation is given, the identification method is unknown.

In 2012, Rich [22] worked on a GAUI 330X-S quadrotor platform. He used several systematic measurement techniques to find the model parameters including the visual feedback and wireless communication delays as well as the lift, drag and moment of inertia constants. Afterwards he used a nested-loop PID controller and a LQR controller.

In 2013, Vanin [23] built an Arducopter quadrotor (Figure 1.13) then modeled it via Newton-Euler approach and linearized the model. He collected attitude angles versus throttle input data of the quadrotor through the motion capture cameras, which are placed around the lab. Then he used the prediction error method (PEM) to identify a linear output error (OE) model of the quadrotor but its performance was quite limited. He implemented a PID controller and tuned its parameters fixing the quadrotor to a test bench. Finally, he implemented a collision avoidance algorithm thus, enabled autonomous flight.

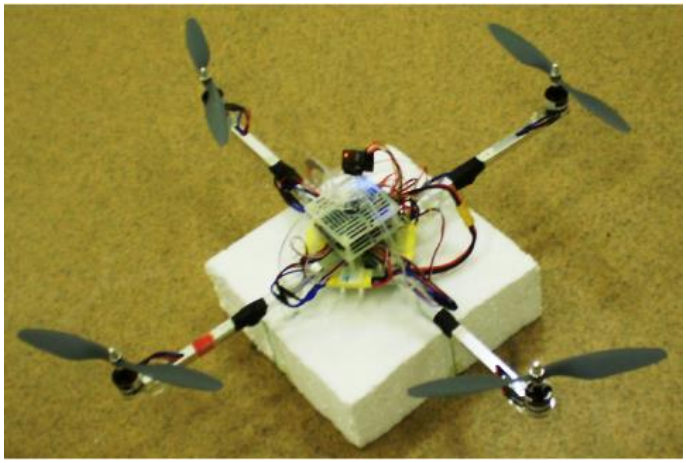


Figure 1.13 : Arducopter used by Vanin.

In 2014, Li et al. [24] from Delft University of Technology worked on an AR. Drone 2.0 quadrotor (Figure 1.14) while a camera based motion capture system was already available in the lab. They presented a Newton-Euler based cross-type quadrotor model and calculated moment of inertia constants of the body and the propellers.

They used motor speed sensors to read motor speed versus throttle input data to obtain motor parameters using the least squares method, then measured the lift and drag coefficients locking the quadrotor axes. Finally, they used attitude angles versus throttle input data to identify a Box-Jenkins (BJ) based linear model using prediction error method (PEM).

Figure 1.14 : AR. Drone 2.0 used by Li et al.

2. QUADROTOR MOTION PRINCIPLES

A quadrotor is a rotorcraft vehicle that is capable of hovering, vertical take-off and landing, as well as four-way (forward/backward/sideward) flight. The lift is generated by four propellers.

Unlike most rotorcrafts, quadrotors use fixed pitched propellers. A couple of the propellers rotate clockwise (CW) as the other couple rotates counter clockwise (CCW). The angular velocity variations between the propellers cause lift and torque thus motion.

There are two different configurations (Figure 2.1) used for quadrotor flight; plus-type configuration and cross-type (x-type) configuration. For photography cross-type configuration is more popular as the propellers can be kept out of the screen and for sport flying, plus-type configuration is better because of being like an airplane and easier to fly [25]. These configurations are given in the following figure.

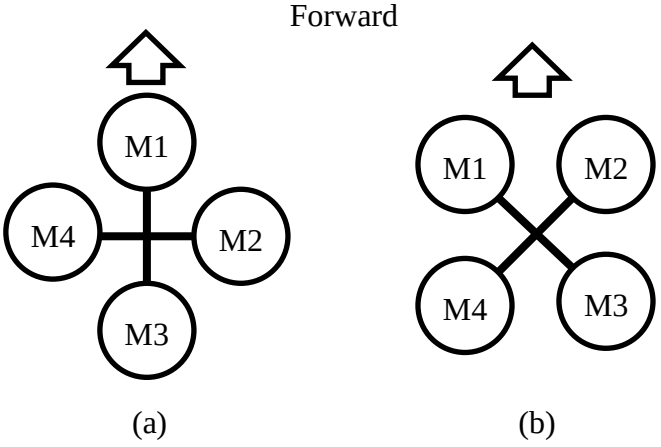


Figure 2.1 : (a) Plus-type and (b) cross-type configurations.

In this work, plus-type configuration is used because of its simplicity and quasi-decoupled dynamic behaviors.

2.1 Hovering

Hovering, take-off and landing are done thanks to the lift generated by the propellers. All of the rotors contribute to hover as their total thrust generates lift.

2.2 Forward and Backward Flight

Forward and backward flight requires angular tilt, which means horizontal thrust. When the angular velocity of M3 is slightly greater than M1 the quadrotor tilts and flies forward. When the angular velocity of M1 is greater than M3 it flies backward.

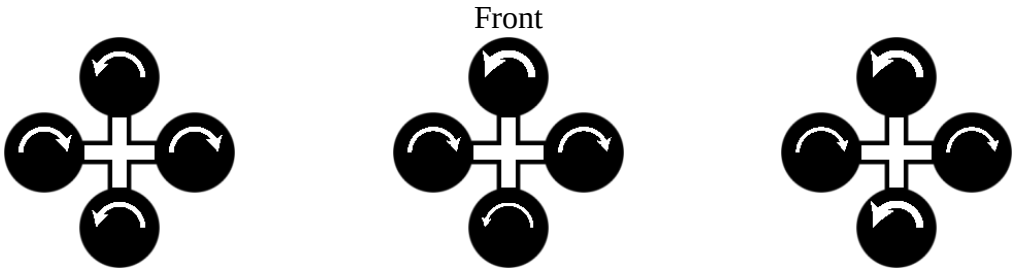
2.3 Sideward Flight

Sideward flight also requires angular tilt. When the angular velocity of M4 is slightly greater than M2, the quadrotor tilts and flies to right and for the opposite it flies left.

2.4 Direction

Each propeller generates a reaction torque due to the moment of inertia of its rotor and air friction. As described before, front and back propellers rotate counter clockwise generating clockwise torque while the others rotate clockwise generating counter clockwise torque.

Therefore, if total torque generated by M1 and M3 is greater than the torque generated by M2 and M4 the quadrotor body starts to rotate clockwise. And for the opposite condition it rotates counter clockwise. This principle is shown below (Figure 2.2, taken from [26]).



(a) (b) (c)

Figure 2.2 : (a) Hover, (b) backward flight and (c) clockwise rotation [26].

3. QUADROTOR MODELING

A quadrotor has 6 degrees of freedom (DOF) and high order dynamics causing a nonlinear mathematical model. Also some coordinate frames and transformations are required to describe its position and orientation. A detailed mathematical model of a quadrotor will be given in this chapter.

3.1 Quadrotor Model Overview

The high order dynamical model of a quadrotor should be separated to the following lower order subsystems.

- Motor dynamics,
- Quadrotor body dynamics,
    - Translational dynamics,
    - Rotational dynamics,

The following block diagram (Figure 3.1) represents the structure that offers great simplicity and independence of the subsystems.

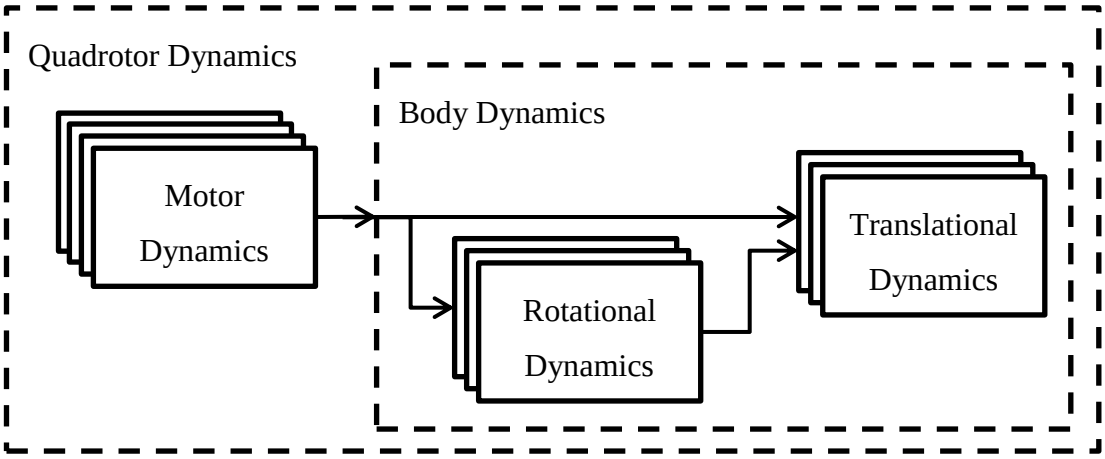


Figure 3.1 : Quadrotor dynamic model overview.

All of the blocks seen in the diagram will be explained in the following sections of this chapter.

3.2 The Inertial Frame

The position and orientation of the quadrotor will be given relative to a fixed coordinate frame, which is called the inertial frame. The X and Y axes of the frame are placed parallel and coincident to the ground while Z axis is pointing upwards in a right handed configuration.

This configuration is also described by East, North, Up (ENU) coordinates.

$$\zeta \triangleq (x, y, z) \in \mathbb{R}^3 \tag{3.1}$$

3.3 The Body Frame

The mobile frame placed to the center of gravity of the quadrotor is called the body frame. The X_B axis points the forward direction of the quadrotor, the Y_B axis points to the left and the Z_B axis points up in a right-handed configuration (Figure 3.2).

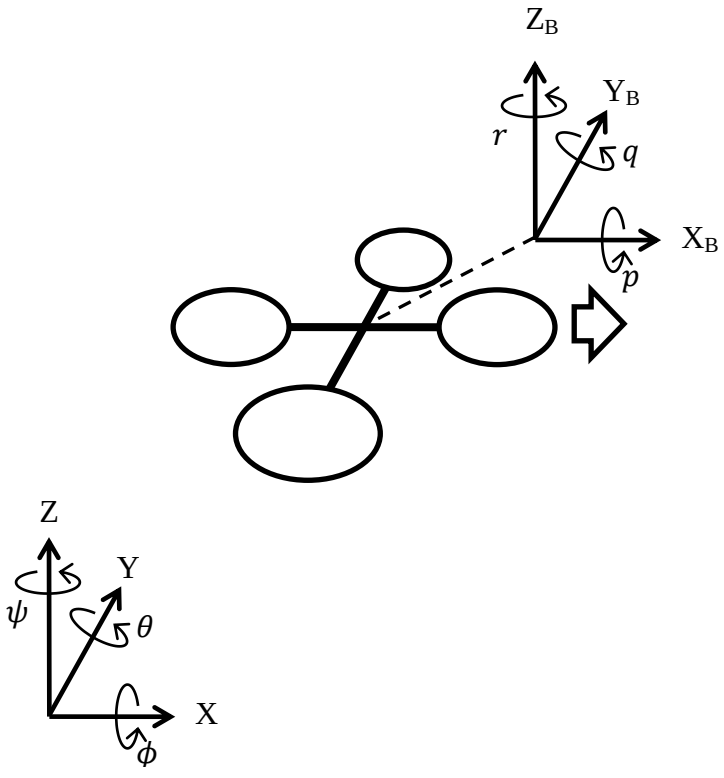


Figure 3.2 : Reference frames, rotation angles and rotation rates.

3.4 Euler Angles

The most important problem in quadrotor control is its orientation in space. The well-known Euler angles representation is suitable for this purpose.

$$\eta \triangleq (\psi, \theta, \phi) \in \mathbb{R}^3 \tag{3.2}$$

The Euler angles ϕ (roll), ψ (pitch) and ψ (yaw) are the rotation angles about the axes X, Y and Z respectively (shown in Figure 3.2).

3.5 Euler Rotations

The fundamental rotation matrices [27] that transform an orientation from the body frame to the inertial frame are given below.

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & Cos\phi & -Sin\phi \\ 0 & Sin\phi & Cos\phi \end{bmatrix} \tag{3.3}$$

$$R_y(\theta) = \begin{bmatrix} Cos\theta & 0 & Sin\theta \\ 0 & 1 & 0 \\ -Sin\theta & 0 & Cos\theta \end{bmatrix} \tag{3.4}$$

$$R_z(\psi) = \begin{bmatrix} Cos\psi & -Sin\psi & 0 \\ Sin\psi & Cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

The yaw-pitch-roll (YPR or ZYX) composite rotation matrix [27] that transforms an orientation from the body frame to the inertial frame is given below.

$$R(\phi, \theta, \psi) = R_z(\psi) R_y(\theta) R_x(\phi)$$

$$= \begin{bmatrix} Cos\theta Cos\psi & Cos\psi Sin\theta Sin\phi - Cos\phi Sin\psi & Cos\phi Cos\psi Sin\theta + Sin\phi Sin\psi \\ Cos\theta Sin\psi & Cos\phi Cos\psi + Sin\theta Sin\phi Sin\psi & -Cos\psi Sin\phi + Cos\phi Sin\theta Sin\psi \\ -Sin\theta & Cos\theta Sin\phi & Cos\theta Cos\phi \end{bmatrix} \tag{3.6}$$

It is also possible to write an inverse rotation matrix by simply transposing [27] the matrix (3.6).

$$R^{-1}(\phi, \theta, \psi) = R^T(\phi, \theta, \psi) \tag{3.7}$$

3.6 Angular Velocities

It is also required deal with the angular velocities (attitude rates) of the quadrotor. The angular velocities p, q and r are shown in Figure 3.2.

$$\Omega \triangleq (p, q, r) \in \mathbb{R}^3 \tag{3.8}$$

3.7 Angular Rate Transformations

The matrix that transforms the angular rates from the inertial frame to the body frame is obtained below [28].

$$\Omega = W_\eta \dot{\eta} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R^{-1}(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R^{-1}(\phi) R^{-1}(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}$$

$$= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & Cos\phi & Sin\phi \\ 0 & -Sin\phi & Cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & Cos\phi & Sin\phi \\ 0 & -Sin\phi & Cos\phi \end{bmatrix} \begin{bmatrix} Cos\theta & 0 & -Sin\theta \\ 0 & 1 & 0 \\ Sin\theta & 0 & Cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{3.9}$$

$$W_\eta = \begin{bmatrix} 1 & 0 & -Sin\theta \\ 0 & Cos\phi & Cos\theta Sin\phi \\ 0 & -Sin\phi & Cos\theta Cos\phi \end{bmatrix} \tag{3.10}$$

It is also possible to write an inverse transformation matrix by inverting matrix (3.10).

$$\dot{\eta} = W_\eta^{-1} \Omega = \begin{bmatrix} 1 & Sin\phi Tan\theta & Cos\phi Tan\theta \\ 0 & Cos\phi & -Sin\phi \\ 0 & Sec\theta Sin\phi & Cos\phi Sec\theta \end{bmatrix} \Omega \tag{3.11}$$

3.8 Thrust Generation

All motor on the quadrotor contribute to the main thrust (lift) relative to their angular velocities. It is possible to say that the motor M_i produces the force f_{Mi}, which is proportional to the square of the angular speed [29] and the total thrust is the sum of the individual thrusts.

$$f_{Mi} \triangleq k\omega_{Mi}^2 \tag{3.12}$$

$$f_z = \sum f_{Mi} \tag{3.13}$$

The main thrust vector (Figure 3.3) acts upwards on the Z axis of the quadrotor body and is shown below.

$$F_B = \begin{bmatrix} 0 \\ 0 \\ f_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ k\sum\omega_{Mi}^2 \end{bmatrix} \tag{3.14}$$

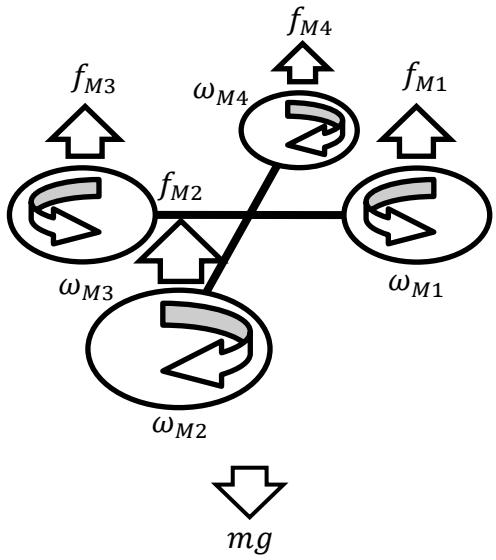


Figure 3.3 : Forces generated by the motors and their speeds.

3.9 Torque Generation

All of the quadrotor maneuvers are done thanks to the torques generated by the motors as described in chapter 2.

The differences in motor M2 and M4 thrusts i.e. different angular velocities causes a torque is about the X axis of the body frame. Similarly motor M1 and M3 causes a torque about the Y axis of the body frame (Figure 3.4).

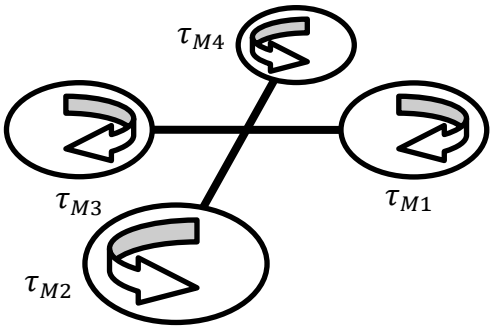


Figure 3.4 : Torques generated by the motors.

$$\tau_{\phi} = f_{M4} - f_{M2} \tag{3.15}$$

$$\tau_{\theta} = f_{M3} - f_{M1} \tag{3.16}$$

The torque acting about the Z axis of the body frame is relative to the air drags and the moment of inertia of the rotors. The air drag acting on one of the motors' blades is given below where ρ is the air density, A is the frontal area of the blade and r is its radius [29].

$$\tau_{drag} = \frac{1}{2}\rho A r^2 \omega^2 \triangleq b\omega^2 \tag{3.17}$$

If the rotor moment of inertia is I_r the torque equation for one of the motors is given below.

$$\tau_{Mi} = I_r \dot{\omega}_{Mi} + b\omega_{Mi}^2 \tag{3.18}$$

As the rotor speed is quasi-stationary [29], equation (3.18) becomes;

$$\tau_{Mi} = b\omega_{Mi}^2 \tag{3.19}$$

Finally knowing that the motors M1 and M3 generates clockwise torque while M2 and M4 generates counter clockwise torque the equation below gives the torque acting about Z axis of the body frame.

$$\tau_{\psi} = b(-\omega_{M1}^2 + \omega_{M2}^2 - \omega_{M3}^2 + \omega_{M4}^2) \tag{3.20}$$

Gathering, the torque vector acting about the body axes is given below.

$$\tau_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} k(\omega_{M4}^2 - \omega_{M2}^2) \\ k(\omega_{M3}^2 - \omega_{M1}^2) \\ b(-\omega_{M1}^2 + \omega_{M2}^2 - \omega_{M3}^2 + \omega_{M4}^2) \end{bmatrix} \quad \textbf{(3.21)}$$

3.10 Inertial Frame Model

An Euler-Lagrange modeling approach will be used to obtain the inertial frame based quadrotor model. The 6 DOF generalized coordinates vector is obtained by combining the translational and rotational components.

$$q \triangleq (\zeta, \eta) = (x, y, z, \psi, \theta, \phi) \in \mathbb{R}^6 \quad \textbf{(3.22)}$$

The Lagrangian is made up translational and rotational kinetic energy and potential energy.

$$L(q, \dot{q}) = T_{trans} + T_{rotational} - U_{potential} \quad \textbf{(3.23)}$$

The Euler-Lagrange equation is given below.

$$\begin{bmatrix} F \\ \tau \end{bmatrix} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} \quad \textbf{(3.24)}$$

3.10.1 Translational motion

Translational kinetic energy of a quadrotor is given below.

$$T_{tans} = \frac{1}{2} m \dot{\zeta}^T \dot{\zeta} \quad \textbf{(3.25)}$$

Potential energy is given below. Luckily, it has no rotational components.

$$U_{potential} = mg\zeta \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = mgz \quad \textbf{(3.26)}$$

The generalized force comes from the main thrust and it is given relative to the body frame before. It will be transformed to the inertial frame by the rotation matrix.

$$U \triangleq k\sum\omega_{Mi}^2 \tag{3.27}$$

$$F = RF_B = U\begin{bmatrix} Cos\phi Cos\psi Sin\theta + Sin\phi Sin\psi \\ -Cos\psi Sin\phi + Cos\phi Sin\theta Sin\psi \\ Cos\theta Cos\phi \end{bmatrix} \tag{3.28}$$

Finally gathering all together;

$$L_{trans}(\zeta, \dot{\zeta}) = T_{trans} - U_{potential} \tag{3.29}$$

$$L_{trans}(\zeta, \dot{\zeta}) = \frac{1}{2}m\dot{\zeta}^T\dot{\zeta} - mgz \tag{3.30}$$

$$F = \frac{d}{dt}\left(\frac{\partial L_{trans}}{\partial \dot{\zeta}}\right) - \frac{\partial L_{trans}}{\partial \zeta} \tag{3.31}$$

$$F = m\ddot{\zeta} - mg\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.32}$$

After some arrangements, the nonlinear translational dynamics are obtained below.

$$m\ddot{x} = U(Cos\phi Cos\psi Sin\theta + Sin\phi Sin\psi) \tag{3.33}$$

$$m\ddot{y} = U(-Cos\psi Sin\phi + Cos\phi Sin\theta Sin\psi) \tag{3.34}$$

$$m\ddot{z} = U(Cos\theta Cos\phi) - mg \tag{3.35}$$

3.10.2 Rotational motion

It can be said that a traditional quadrotor body is symmetrical for X_B, Y_B and Z_B axes. Thus, its moment of inertia matrix will be symmetrical.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.36}$$

Rotational kinetic energy of a quadrotor is given below.

$$T_{rotational} = \frac{1}{2}\Omega^T I \Omega \tag{3.37}$$

Angular rate transformation matrix could be used to write the Jacobian.

$$J \triangleq W_\eta^T I W_\eta =$$

$$\begin{bmatrix} I_{xx} & 0 & -I_{xx}Sin\theta \\ 0 & I_{yy}Cos^2\phi + I_{zz}Sin^2\phi & (I_{yy} - I_{zz})Cos\theta Cos\phi Sin\phi \\ -I_{xx}Sin\theta & (I_{yy} - I_{zz})Cos\theta Cos\phi Sin\phi & I_{xx}Sin\theta^2 + Cos\theta^2(I_{zz}Cos\phi^2 + I_{yy}Sin\phi^2) \end{bmatrix} \tag{3.38}$$

Now the expression (3.37) could be written explicitly.

$$T_{rotational} = \frac{1}{2}\dot{\eta}^T W_\eta^T I W_\eta \dot{\eta} \tag{3.39}$$

$$T_{rotational} = \frac{1}{2}\dot{\eta}^T J \dot{\eta} \tag{3.40}$$

The generalized torque vector is given before and it is the same for the inertial frame.

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \triangleq \tau = \tau_B \tag{3.41}$$

The Euler-Lagrange equation becomes;

$$L_{rotational}(\eta, \dot{\eta}) = T_{rotational} \tag{3.42}$$

$$\tau = \frac{d}{dt}\left(\frac{\partial L_{rotational}}{\partial \dot{\eta}}\right) - \frac{\partial L_{rotational}}{\partial \eta} \tag{3.43}$$

$$\tau = J\ddot{\eta} + \frac{dJ}{dt}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T J \dot{\eta}) \tag{3.44}$$

It is possible to group (3.44) by the Coriolis term that includes the gyroscopic and centripetal forces [30].

$$\tau = J\ddot{\eta} + C(\eta, \dot{\eta})\dot{\eta} \tag{3.45}$$

After some arrangements, the nonlinear rotational dynamics are obtained below.

$$J\ddot{\eta} = \begin{bmatrix} U_1 + (-Iyy + Izz)Cos\phi Sin\phi\theta'^2 \\ +Cos\theta(Ixx + (Iyy - Izz)Cos2\phi)\theta'\psi' \\ +(Iyy - Izz)Cos\theta^2 Cos\phi Sin\phi\psi'^2 \\ \\ U_2 + (Iyy - Izz)Sin2\phi\theta'\phi' \\ +Cos\theta\psi'(-(Ixx + (Iyy - Izz)Cos2\phi)\phi' \\ +Sin\theta(Ixx - IzzCos\phi^2 - IyySin\phi^2)\psi') \\ \\ U_3 + \frac{1}{2}((Iyy - Izz)Sin\theta Sin2\phi\theta'^2 + 2(-Iyy + Izz)Cos\theta^2 Sin2\phi\phi'\psi' \\ +2Cos\theta\theta'((Ixx + (-Iyy + Izz)Cos2\phi)\phi' \\ +(-2Ixx + Iyy + Izz + (-Iyy + Izz)Cos2\phi)Sin\theta\psi')) \end{bmatrix} \quad \textbf{(3.46)}$$

To simplify the model the following assumption [6] will be reasonable.

$$I \triangleq \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{xx} & 0 \\ 0 & 0 & 2I_{xx} \end{bmatrix} \quad \textbf{(3.47)}$$

$$J\ddot{\eta} = \begin{bmatrix} U_1 + IxxSin\phi\left(Cos\phi\theta'^2 + 2Cos\theta Sin\phi\theta'\psi' - Cos\theta^2 Cos\phi\psi'^2\right) \\ \\ U_2 - Ixx\left(Sin2\phi\theta'\phi' + Cos\theta\psi'(2Sin\phi^2\phi' + Cos\phi^2 Sin\theta\psi')\right) \\ \\ U_3 - IxxCos\phi Sin\theta Sin\phi\theta'^2 + IxxCos\theta^2 Sin2\phi\phi'\psi' \\ +2IxxCos\theta Cos\phi^2\theta'(\phi' + Sin\theta\psi') \end{bmatrix} \quad \textbf{(3.48)}$$

Although some simplifications are made, the inertial frame rotational model is still too complicated to go with.

3.11 Body Frame Model

As the translational part of the body frame model is the same as of the inertial frame model (3.33) (3.34) (3.35), only the rotational part will be given.

Body frame model is constructed based on a Newton-Euler approach [31] which is given below, where I_r represents the total moment of inertia of a rotor.

$$I\dot{\Omega} = \tau_B - \Omega \times I\Omega - I_r\Omega \times \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \quad \textbf{(3.49)}$$

$$\omega = -\omega_{M1} + \omega_{M2} - \omega_{M3} + \omega_{M4} \quad \textbf{(3.50)}$$

After substitutions and some arrangements, the body frame rotational model will be obtained as follows.

$$\dot{p} = \frac{U_1}{I_{xx}} + \frac{I_{yy} - I_{zz}}{I_{xx}} qr - \frac{I_r}{I_{xx}} q\omega \quad \textbf{(3.51)}$$

$$\dot{q} = \frac{U_2}{I_{yy}} + \frac{I_{zz} - I_{xx}}{I_{yy}} pr + \frac{I_r}{I_{yy}} p\omega \quad \textbf{(3.52)}$$

$$\dot{r} = \frac{U_3}{I_{zz}} + \frac{I_{xx} - I_{yy}}{I_{zz}} pq \quad \textbf{(3.53)}$$

If the Euler angles ϕ, θ, ψ assumed to be small [32], then expression (3.54) will be reasonable.

$$\dot{\eta} \approx \Omega \quad \textbf{(3.54)}$$

Now the body frame angles can be replaced with the Euler angles. Thus, the body frame nonlinear model is obtained.

$$m\ddot{x} = U(Cos\phi Cos\psi Sin\theta + Sin\phi Sin\psi) \quad \textbf{(3.33)}$$

$$m\ddot{y} = U(-Cos\psi Sin\phi + Cos\phi Sin\theta Sin\psi) \quad \textbf{(3.34)}$$

$$m\ddot{z} = U(Cos\theta Cos\phi) - mg \quad \textbf{(3.35)}$$

$$\ddot{\phi} = \frac{U_1}{I_{xx}} + \frac{I_{yy} - I_{zz}}{I_{xx}} \dot{\theta}\dot{\psi} - \frac{I_r}{I_{xx}} \dot{\theta}\omega \quad \textbf{(3.55)}$$

$$\ddot{\theta} = \frac{U_2}{I_{yy}} + \frac{I_{zz} - I_{xx}}{I_{yy}} \dot{\phi}\dot{\psi} + \frac{I_r}{I_{yy}} \dot{\phi}\omega \quad \textbf{(3.56)}$$

$$\ddot{\psi} = \frac{U_3}{I_{zz}} + \frac{I_{xx} - I_{yy}}{I_{zz}} \dot{\phi}\dot{\theta} \quad \textbf{(3.57)}$$

3.12 Simplified Model

The nonlinear quadrotor model obtained before includes gyroscopic effects and moment of inertia of the motors. To simplify the model they can be assumed small and neglected [32]. The most basic quadrotor model is given below.

$$m\ddot{x} = U(Cos\phi Cos\psi Sin\theta + Sin\phi Sin\psi) \tag{3.33}$$

$$m\ddot{y} = U(-Cos\psi Sin\phi + Cos\phi Sin\theta Sin\psi) \tag{3.34}$$

$$m\ddot{z} = U(Cos\theta Cos\phi) - mg \tag{3.35}$$

$$\ddot{\phi} = \frac{U_1}{I_{xx}} \tag{3.58}$$

$$\ddot{\theta} = \frac{U_2}{I_{yy}} \tag{3.59}$$

$$\ddot{\psi} = \frac{U_3}{I_{zz}} \tag{3.60}$$

The rotational part is linear but the translational part is still nonlinear.

3.13 State Equations

To simplify a controller design, it will be useful to represent the model using state equations. There are 6 DOF, thus 6 second order equations are found previously. The whole model can be represented using the 12 state variables given below.

$$x_1 = \dot{\phi}, \quad x_2 = \dot{\theta}, \quad x_3 = \dot{\psi}, \quad x_4 = \phi, \quad x_5 = \theta, \quad x_6 = \psi \tag{3.61}$$

$$x_7 = \dot{x}, \quad x_8 = \dot{y}, \quad x_9 = \dot{z}, \quad x_{10} = x, \quad x_{11} = y, \quad x_{12} = z \tag{3.62}$$

When the state variables are substituted into the model, following state equations that represent the quadrotor dynamics will be obtained.

$$\dot{x}_1 = \frac{U_1}{I_{xx}} \tag{3.63}$$

$$\dot{x}_2 = \frac{U_2}{I_{yy}} \tag{3.64}$$

$$\dot{x}_3 = \frac{U_3}{I_{zz}} \tag{3.65}$$

$$\dot{x}_4 = x_1 \tag{3.66}$$

$$\dot{x}_5 = x_2 \tag{3.67}$$

$$\dot{x}_6 = x_3 \tag{3.68}$$

$$\dot{x}_7 = \frac{U}{m}(Cosx_4 Cos\psi Sinx_5 + Sinx_4 Sinx_6) \tag{3.69}$$

$$\dot{x}_8 = \frac{U}{m}(-Cosx_6 Sinx_4 + Cosx_4 Sinx_5 Sinx_6) \tag{3.70}$$

$$\dot{x}_9 = \frac{U}{m}(Cosx_5 Cosx_4) - g \tag{3.71}$$

$$\dot{x}_{10} = x_7 \tag{3.72}$$

$$\dot{x}_{11} = x_8 \tag{3.73}$$

$$\dot{x}_{12} = x_9 \tag{3.74}$$

3.14 Model Decoupling

When the state equations are inspected, it can be clearly seen that the rotational part of the model (equations (3.69) to (3.74)) does not depend on the translational part (equations (3.69) to (3.74)) while the translational part depends on the rotational part. Using this property, the model can be inspected as two subsystems [15] (Figure 3.5).

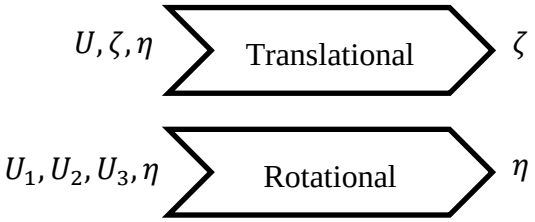


Figure 3.5 : Decoupled model.

3.15 Motor Model

The most important component that affects the whole system dynamics is the motor. Although different kind of motors are used in different quadrotor types, a generalized DC motor model will be sufficient to cover most of them.

3.15.1 DC motor model

A quadrotor model can be taught as an armature controlled DC motor driving a propeller. The following diagram (Figure 3.6) will be used. There is no inductance modeled to keep the model as simple as possible.

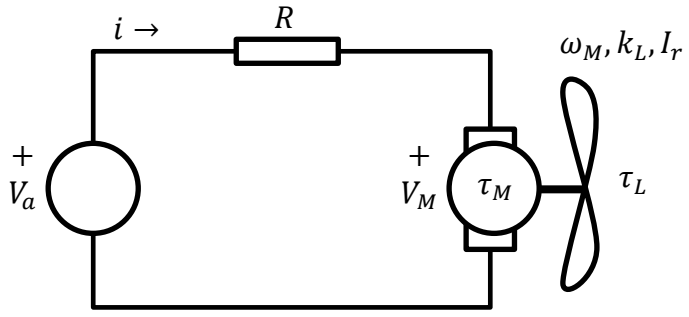


Figure 3.6 : Motor model.

The torque, load torque and back EMF relations are given below, respectively.

$$\tau_M = k_T i \tag{3.75}$$

$$\tau_L = k_L \omega_M^2 \tag{3.76}$$

$$V_M = k_M \omega_M \tag{3.77}$$

Then using Kirchhoff's voltage law the following equation can be written.

$$i = \frac{V_a - k_M \omega_M}{R} \tag{3.78}$$

And the torque equation is given below where k_L is the total load coefficient of the propeller, and the load is relative to the square of the motor speed [4].

$$k_L = k + b \tag{3.79}$$

$$\tau = I_r \dot{\omega}_M + k_L \omega_M^2 \tag{3.80}$$

After substitutions, the following model is obtained.

$$\dot{\omega}_M = \frac{k_T}{R I_r} V_a - \frac{k_M k_T}{R I_r} \omega_M - \frac{k_L}{I_r} \omega_M^2 \tag{3.81}$$

If x is chosen as the only state variable and the input is u, then the following first order, nonlinear state equation is obtained.

$$\dot{x} = a_0 u - a_1 x - a_2 x^2 \tag{3.82}$$

3.15.2 Driver stage

This work will cover a real time quadrotor platform. Thus, modeling a simple DC motor driver stages is required. In real-time systems, the most basic structure is pulse width modulation (PWM) based single switch driver (Figure 3.7).

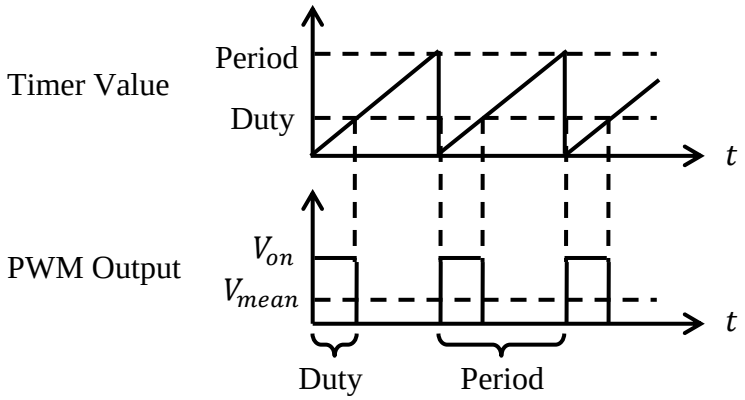


Figure 3.7 : PWM signal.

PWM is a high frequency (20kHz to sub-MHz) signal of which on-time (duty) is controlled by a periodical timer. The frequency of PWM signal is higher than the load (in this case a motor) can catch up. Thus, only a mean voltage is seen by the load.

The PWM driven single switch motor driver is shown below (Figure 3.8).

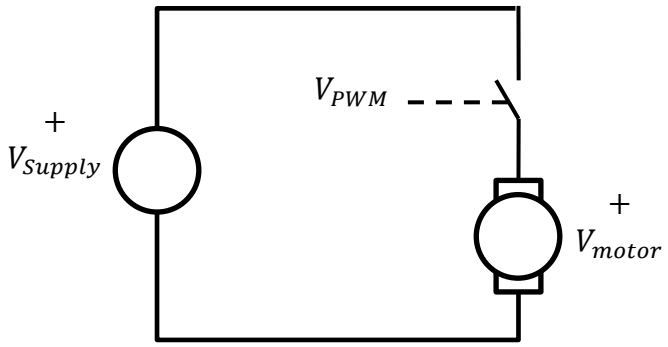


Figure 3.8 : Single switch motor driver.

The mean voltage applied to the motor is given by the equation below, where duty is a ratio which is not greater than unity.

$$0 \leq Duty \leq 1 \tag{3.83}$$

$$V_{motor} = V_{Supply} * Duty \tag{3.84}$$

4. HARDWARE PLATFORM

One of the goals of this work includes experimental real-time data collecting. For that purpose, a quadrotor platform is used.

4.1 Hardware Requirements

There are several quadrotor platforms readily available in the market. The requirements for the experimental platform may be listed as follows.

- Available for indoor use,
- Ready to fly,
- Small in size,
- Inexpensive,
- Accessible spare parts,
- High performance microcontroller,
- Low-level programming available,
- High precision sensors,
- Efficient motors,
- High quality propellers,
- Computer connectivity,

4.2 The Crazyflie Platform

Crazyflie mini quadrotor (Figure 4.1) platform is chosen as it meets all of the requirements.

- Small and lightweight, only19g,
- 90mm motor to motor distance,
- Flight time up to 7 minutes,
- Open-source design,
- Single microcontroller reduces model complexity,

- Spare parts included (Figure 4.2),
- Remote control over computer,

Figure 4.1 : Crazyflie.

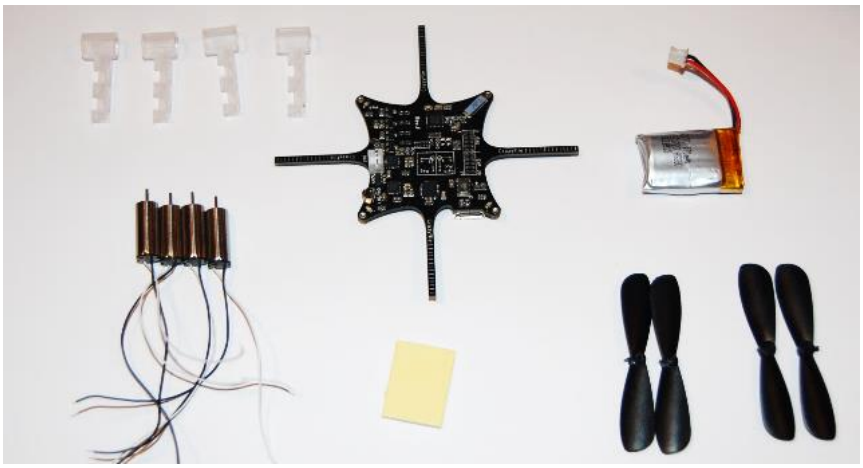


Figure 4.2 : Crazyflie parts.

4.3 Structural Overview

Crazyflie platform consists of the following parts, which are also illustrated below (Figure 4.3).

- Crazyflie quadrotor itself,
- Crazyradio radio link,
- Computer running Crazyflie client software,
- 4-axis analog Joystick.

User controls Crazyflie using a 4-axis analog joystick, which is connected to a computer running Crazyflie Client Software. The software receives input commands

from the joystick over USB and sends them to Crazyradio wireless transceiver that is communicating to Crazyflie quadrotor. The Quadrotor responds to Crazyradio with an acknowledgement message.

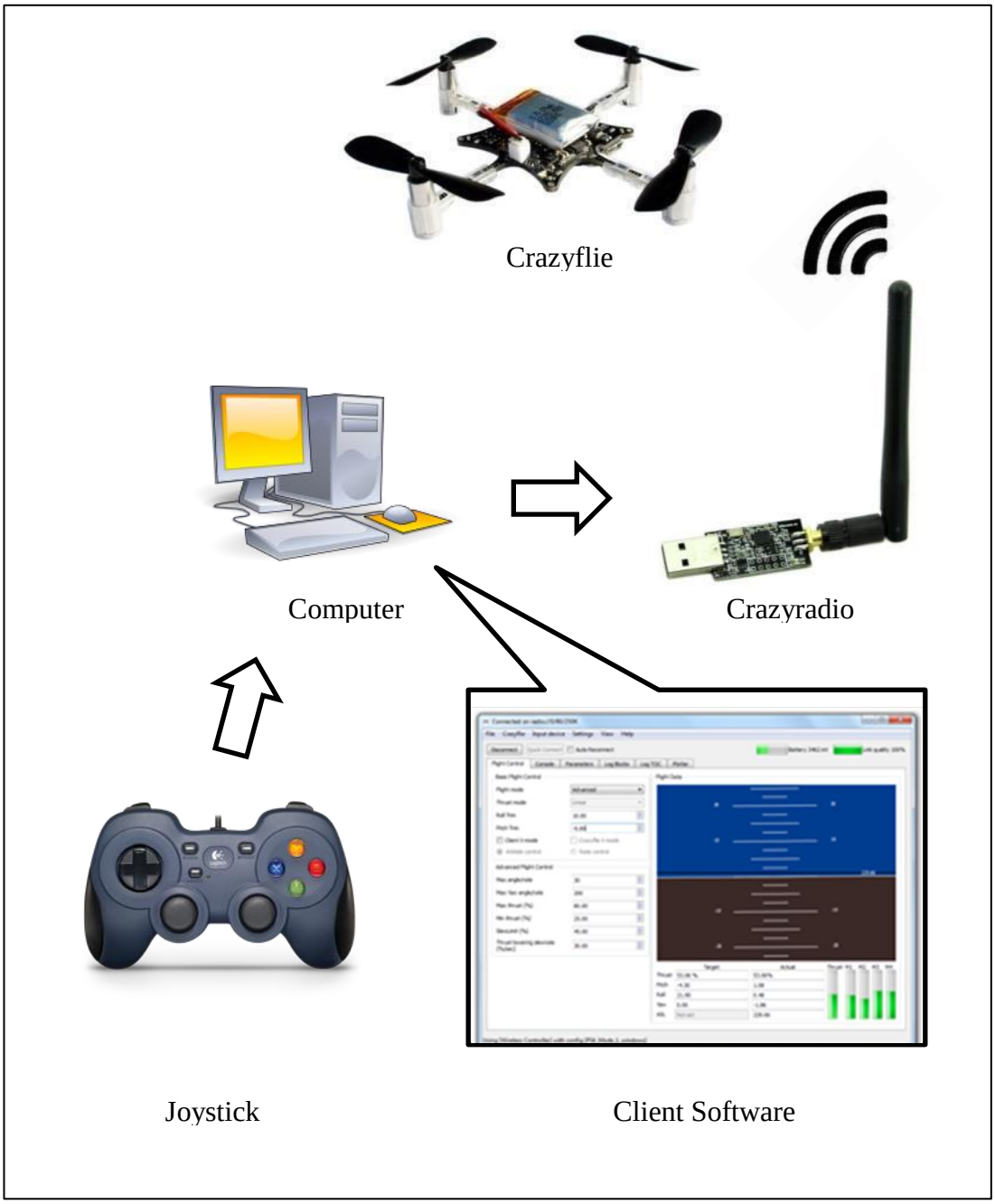


Figure 4.3 : The operating scheme.

4.4 Onboard Electronics

Crazyflie hardware may be inspected in sub partitions; power circuitry, sensors, microcontroller, actuators and radio transceiver circuitry. The onboard electronics block diagram is given in the following figure (Figure 4.4, taken from [33]).

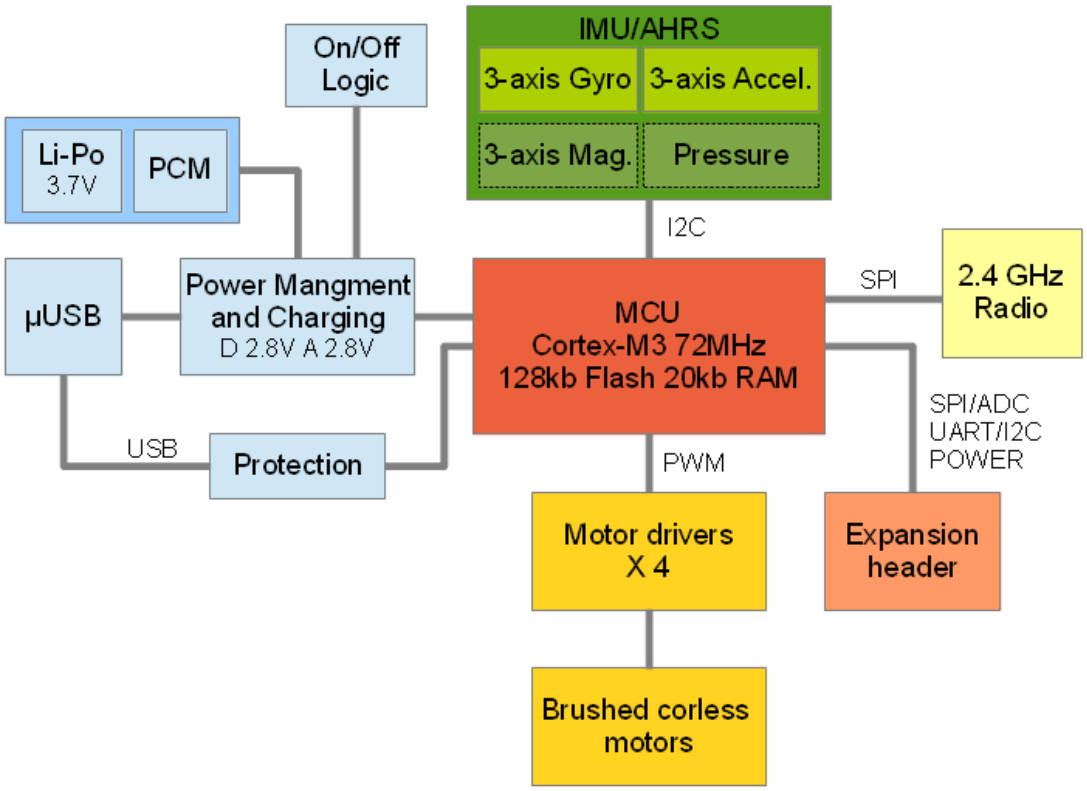


Figure 4.4 : Quadrotor electronics overview [33].

3-axis Gyroscope and 3-axis accelerometer sensors are integrated into a single package. The pressure sensor and 3-axis magnetometer are optional and not included in our platform. Thus, only 6 degrees of freedom sensing is available.

The expansion header is used for telemetry and speed-sensing add-on circuits, which will be explained afterwards.

4.4.1 Microcontroller

An onboard, high performance STM32F103 microcontroller controls Crazyflie. Its basic features are:

- 32-bit ARM Cortex-M3 architecture,
- 72MHz core speed,
- Low power operation,
- 128kB Flash,
- 20kB RAM,
- 37 x GPIO pin,
- 4 x 32-bit Timer,

- 10-channel, 12-bit ADC,
- 2 x I^2C/SPI communication interface,
- Full speed USB,

The microcontroller is responsible for all of the real-time tasks; reading the sensors, data filtering and sensor fusion, control input reception from Crazyradio, running a closed-loop controller and driving the actuators with respect to control loop, charging the lithium-polymer onboard battery.

The software running on this microcontroller will be explained in details afterwards.

4.4.2 Sensors

Inertial measurement sensors are crucial for a quadrotor platform. Crazyflie has single–chip, high-precision MPU6050 sensor of which block diagram is shown below (Figure 4.5, taken from [34]).

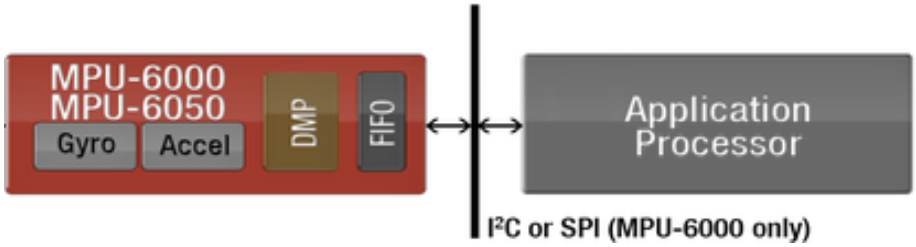


Figure 4.5 : MPU6050 block diagram [34].

It is a micro-electro-mechanical sensor offering 6 degrees of freedom inertial sensing, which is required for the quadrotor to stabilize in the air. The sensor hosts a 3-axis MEMS accelerometer as well as a 3-axis MEMS gyroscope of which axes are illustrated below (Figure 4.6, taken from [34]).

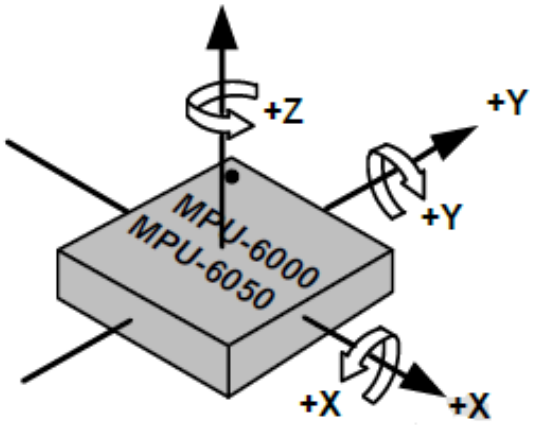


Figure 4.6 : MPU6050 axes [34].

4.4.2.1 Accelerometer

Accelerometers are one of the most common inertial measurement sensors. MEMS based accelerometers are widely used by unmanned vehicles and mobile devices. An accelerometer measures the proper acceleration, so called the g-force.

The operation of a MEMS accelerometer is based on displacement of a proof mass placed in a silicon chip and suspended by small beams. It works in consistence with Newton's law of motion. [35]

The proof mass carries capacitive fingers, which causes capacitance change while the proof mass moves. An example MEMS accelerometer is given below (Figure 4.7, taken from [36]).

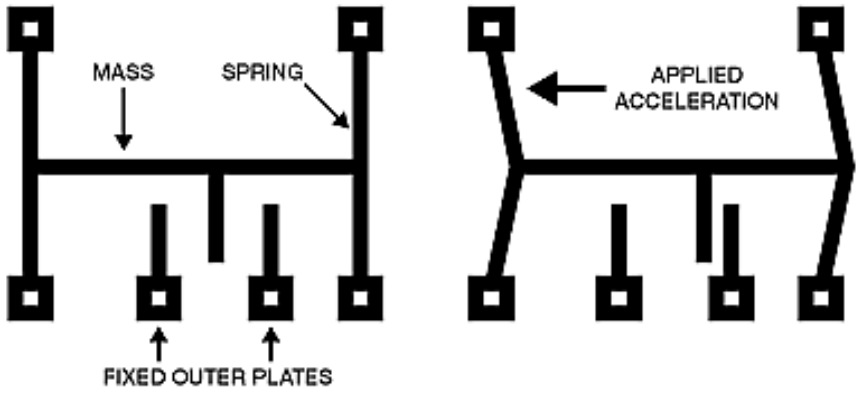


Figure 4.7 : Capacitive sensing MEMS accelerometer [36].

Basic features of MPU6050 accelerometer are given below.

- Digital-output with a programmable range of ±2g, ±4g, ±8g and ±16g,
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers,
- Low power operation,
- 1kHz output data rate,
- I^2C communication with data-ready interrupts,

The onboard accelerometer detects the acceleration caused by the gravity and motion of the body in three axes.

4.4.2.2 Gyroscope

A gyroscope is a device that measures angular velocity about its operating axes. The onboard MEMS gyroscope detects the angular velocities about three axes.

Fundamental principle of operation of a MEMS gyroscope depends on the Coriolis forces. A MEMS gyroscope contains a proof mass that is connected to a wheel by micro springs. The wheel is suspended via orthogonal springs. [37]

This concept is illustrated below (Figure 4.8, taken from [38]).

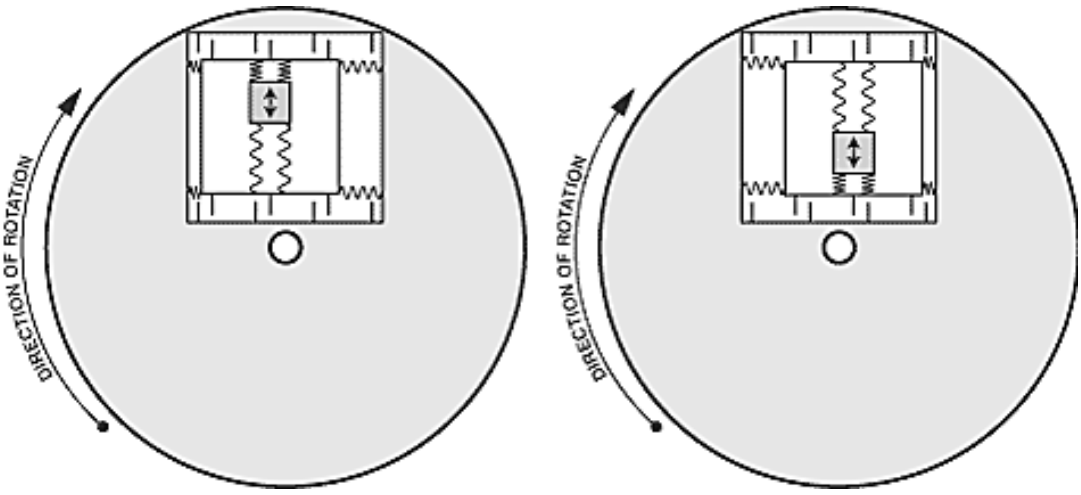


Figure 4.8 : Capacitive sensing MEMS gyroscope [38].

Basic features of MPU6050 gyroscope are given below.

- Digital-output 3-Axis angular rate sensors,
- User-programmable full scale range of ±250, ±500, ±1000, and ±2000°/sec,
- Integrated 16-bit ADCs enable simultaneous sampling of gyros,
- Enhanced bias and sensitivity temperature stability,
- Digitally programmable low-pass filter,
- Low-power operation,
- 8kHz output data rate,

4.4.3 Battery

Crazyflie is designed to be lightweight. So it uses a high power density lithium-polymer battery of which basic features are given below.

- 170mAh full charge capacity,
- 3.7v nominal, 1S configuration,
- Voltage range from 3.0v to 4.2v,

- 1.7A (10C) continuous discharge current,
- Weight 5.2g,

Figure 4.9 : LiPo battery.

The battery is charged over a standard USB cable thanks to the onboard, intelligent battery charge circuitry.

To maintain battery health a PCM (Protection Circuit Module) that prevent the user from under/overcharging or shorting the battery is used. It opens the circuit if a current limit of 4A is exceeded. The PCM is located inside the battery.

4.4.4 Motors

Crazyflie uses four miniature-sized, high-performance, super-efficient, coreless brushed DC motors.

Figure 4.10 : Crazyflie motor.

Motor specifications are listed below.

- Rated voltage 4.2V Max
- Nominal voltage 3.7V
- No load speed 45000 ±15% RPM

- No load current 80mA Max
- Starting voltage 0.8V Max
- Rated load speed 21000 ±15% RPM
- Rated load current 810mA Max
- Resistance 2.3 ±20% Ω
- Shaft diameter 0.8mm
- Motor diameter 6 ± 0.05mm
- Motor length 15mm
- Weight 1.7g (each)

The microcontroller applies a high-frequency PWM signal to the single-switch DC motor drive circuitry. The drive frequency is higher than the frequency motors can respond. Thus, a ratio of the battery voltage is seen by the motors, enabling armature voltage control.

4.5 Radio Link

Command inputs from user to the quadrotor are sent over wireless radio link, Crazyradio (Figure 4.11).

Figure 4.11 : Crazyradio.

Crazyradio and onboard radio link circuitry communicates over NRF24 chips. Some basic properties of the chip are listed below.

- 2.4GHz ISM communication band,
- 125 selectable channels,

- Up to 2Mbps air data rate,
- 0dBm output power (1mW),
- Tested up to 80m range with the Crazyflie at 250Kbps,
- Sends and receives data packets of up to 32 bytes,
- Auto CRC calculation and verification,
- SPI communication to microcontroller,
- USB interface (only for Crazyradio),
- Receive/transmit interrupts,

Communication is done in 32-byte packet basis. A format called CRTP is used between Crazyflie and Crazyradio. The structure is given below (Figure 4.12, taken from [33]).

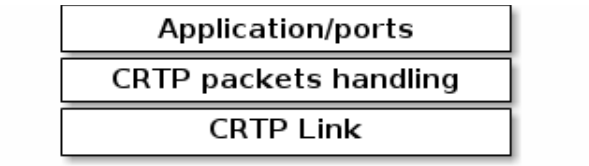


Figure 4.12 : The communication protocol [33].

4.6 Joystick Controller

A standard USB joystick is used for controlling the desired quadrotor angles. 4 analog axes of the joystick are assigned to thrust and Euler angles as illustrated below.

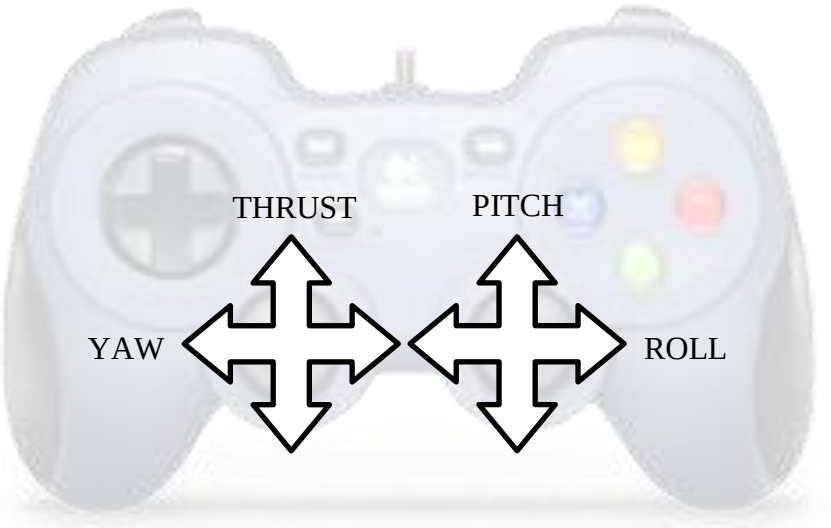


Figure 4.13 : Controller axes.

Ratio-metric input to the axes can be applied using the joystick axes. For example pulling up the left stick increases the thrust while pulling it down does the opposite.

4.7 Embedded Software

The onboard microcontroller is running a real-time operating system called FreeRTOS, thus multiple tasks run simultaneously.

- Commander task;

  This part of software is responsible for communicating to the Crazyradio over the onboard NRF24 radio circuitry, and receiving control inputs from ground. Every time it receives a command, acknowledges.
- Power management task;

  It is responsible for monitoring the battery voltage and shutting the system down if, it falls below a critical limit. It also monitors the battery charging.
- IMU sensor task;

  This task communicates MPU6050 chip over I^2C. While Crazyflie is on the ground, initializes and reads the sensor, then calculates sensor bias values. When Crazyflie is flying, it reads accelerometer and gyroscope data, then combines them. Thus, it produces Euler angles and angular rates.
- Stabilizer task;

  Stabilizer is the most important task running or the Crazyflie. There are two cascaded proportional integral and derivative (PID) closed-loop controllers running on the system. This structure is also known as nested-loop PID.

  The outer loop, which runs at 250Hz, takes the desired Euler angles from the commander task and the Euler angles from the IMU task. Then it produces desired angular rates for the inner loop.

  The inner control loop which runs at 500Hz, takes the desired angular rates from the outer loop and the angular rates from the IMU task, then produces PWM signals for each actuator.

  This task also takes desired thrust input from the commander task and uses it as an offset for the PWM signals calculated by the closed-control loops.

Software block diagrams are given (Figure 4.14, Figure 4.15, taken from [33]).

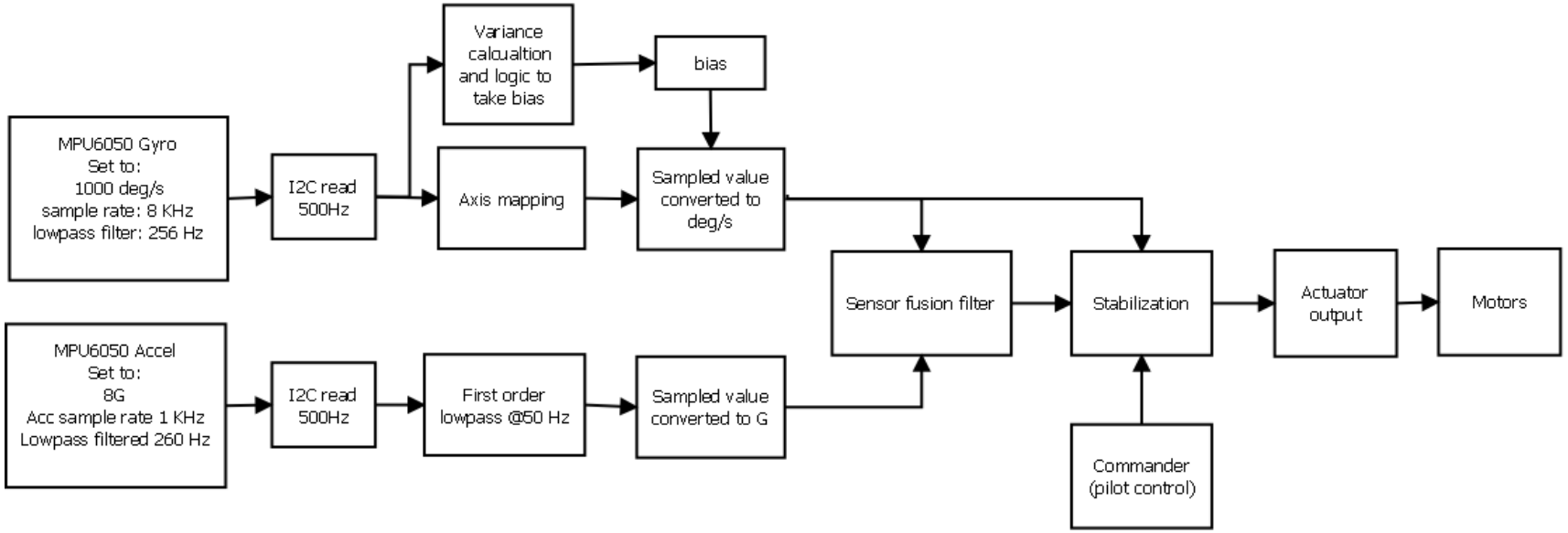


Figure 4.14 : Block diagram of the stabilization system [33].

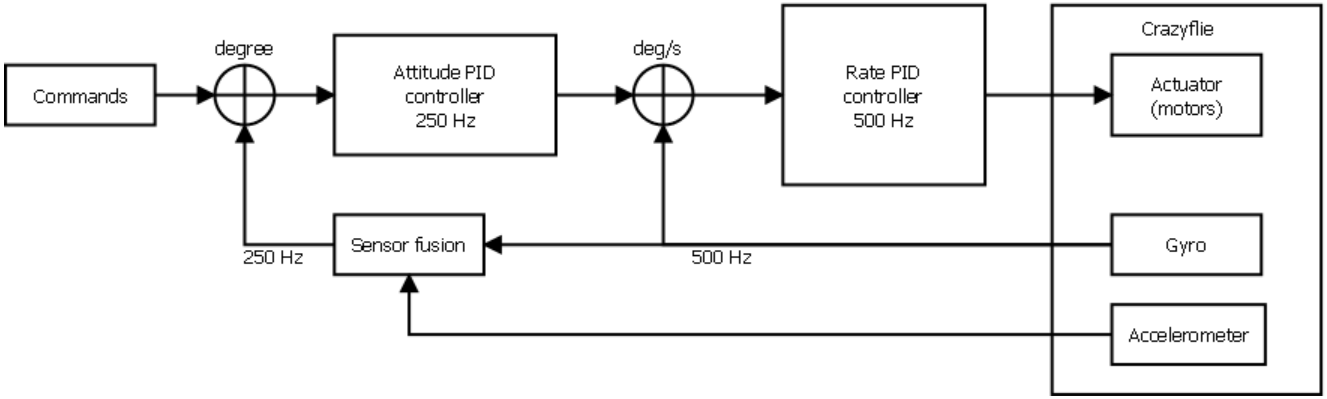


Figure 4.15 : The nested-loop PID controller [33].

5. REAL-TIME DATA ACQUISITION

One of the goals of this study is real-time data acquisition. For that purpose, a telemetry system is implemented on Crazyflie platform (Figure 5.1).

Figure 5.1 : Crazyflie and telemetry circuitry.

5.1 Telemetry Hardware

5.1.1 Transmitter hardware

For real-time data acquisition, a transmitter circuitry is implemented on Crazyflie quadrotor. The additional payload is critical as the quadrotor itself weights 19g. However NRF24L01+ wireless transceiver module (Figure 5.2) is only 4g, because it uses a PCB antenna. The module is connected to the expansion port of the Crazyflie and placed on top of the battery.

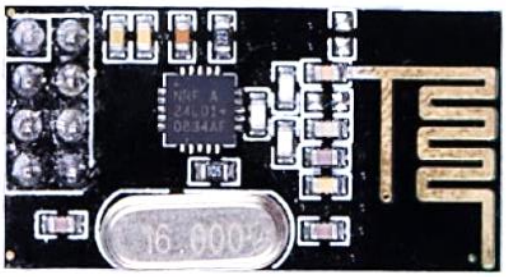


Figure 5.2 : Onboard transceiver module.

The module requires 6 I/O pins for full functionality. But there are only 4 I/O pins are free at the expansion connector. Fortunately, this is enough if the sensor is used as transmitter only.

Thus, SCK, MOSI, CE and CS signals are connected to the microcontroller I/O pins.

5.1.2 Receiver hardware

Although there is no weight limit on the receiver side, the communication distance is critical. A long-range NRF24L01+PA module with antenna, which is fully compatible with the transmitter module, is used (Figure 5.3).

Figure 5.3 : Receiver module.

The module has no USB interface but an SPI interface, so it can not directly communicate to the computer (Figure 5.4).

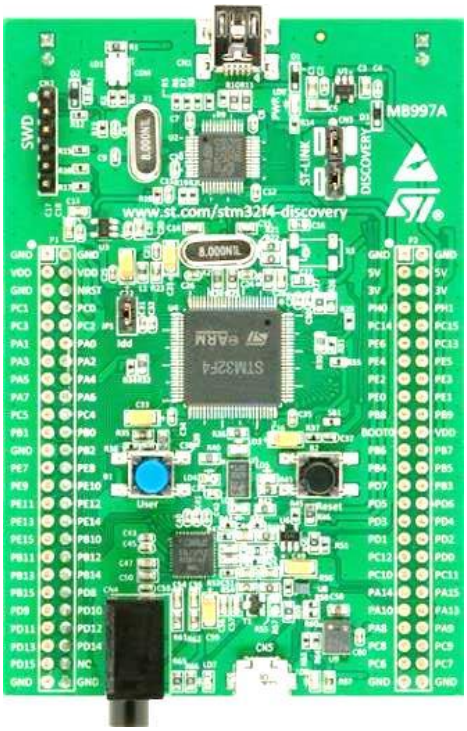


Figure 5.4 : STM32F4-Discovery board.

It is mounted to a STM32F4-Discovery board, which is based on STM32F407 microcontroller that supports Full-Speed USB communication.

Reception requires all the signals; SCK, MOSI, MISO, IRQ, CE and CS are connected to the microcontroller I/O pins.

5.2 Motor Speed Measurement Hardware

For modeling purposes motor speed is crucial. As Crazyflie drives the motors in an open-loop manner (motor speed is not used by the stabilizer), it has no speed sensing circuitry. A simple but high resolution circuitry is implemented using TCST2103 optocoupler (Figure 5.5).

Figure 5.5 : Motor speed sensor.

The optocoupler output is connected to the microcontroller's I/O pin through the debug connector of Crazyflie.

Because of this add-on is mounted to one of the motors, it affects the balance of Crazyflie. For this reason, the motor speed sensing circuitry is kept experimental. It is used to identify the motor model then it is removed.

5.3 Data Structure

The implemented telemetry system is capable of transmitting 32-byte data packs with up to 64kBps data rate. All required data is organized to fit into a single 32-byte data pack. The data structure is given below (Table 5.1).

Table 5.1 : Telemetry data structure.

Data	Unit	Bytes
Timestamp	microseconds	2
Battery voltage	millivolts	2
Motor PWM	1:4096	1.5 x 4
Gyroscope axes	degrees per second	2 x 3
Accelerometer axes	g-force	2 x 3
Roll, pitch, yaw	degrees	2 x 3
Debug / motor speed	-	4

5.4 Transmitter Software

The transmitter software is implemented on Crazyflie to interface the telemetry circuitry through the hardware SPI module of the microcontroller.

As declared before, the NRF24L01+ module is configured as 2Mbps transmitter on system startup. Upon the main program loop starts, the following steps are executed on every 2ms.

- A timestamp is been captured from system timers.
- The required sensors are been read and buffered.
- Motor PWM values are been read and buffered.
- The buffered data pack is been repeatedly transmitted with 500us intervals.

Buffer writes and transmissions are done in an asynchronous way. When the main process fills a new buffer, the transmission process transmits a previously written buffer in the background. This prevents possible data corruptions caused by concurrent operations.

5.5 Receiver Software

Two different programs are implemented for the receiver and they work in cooperation.

5.5.1 Microcontroller side

STM32F4-Discovery board is configured as a bridge between the incoming aerial data and the computer. It would be true if we say;

- NRF24L01+PA receives data from Crazyflie Quadrotor,
- It generates an interrupt to the STM32F4 microcontroller,

- The microcontroller connects to the module over SPI and reads incoming data,
- Compares the new data to the previous one,
- If the data is a new one, it is written to a FIFO buffer,
- Another process transmits the buffered data to the computer over USB,
- The Full Speed USB protocol allows data transmission up to 64 bytes per transaction,
- Up to 1k transactions per second,

The receiver board with the RF module is shown below (Figure 5.6).

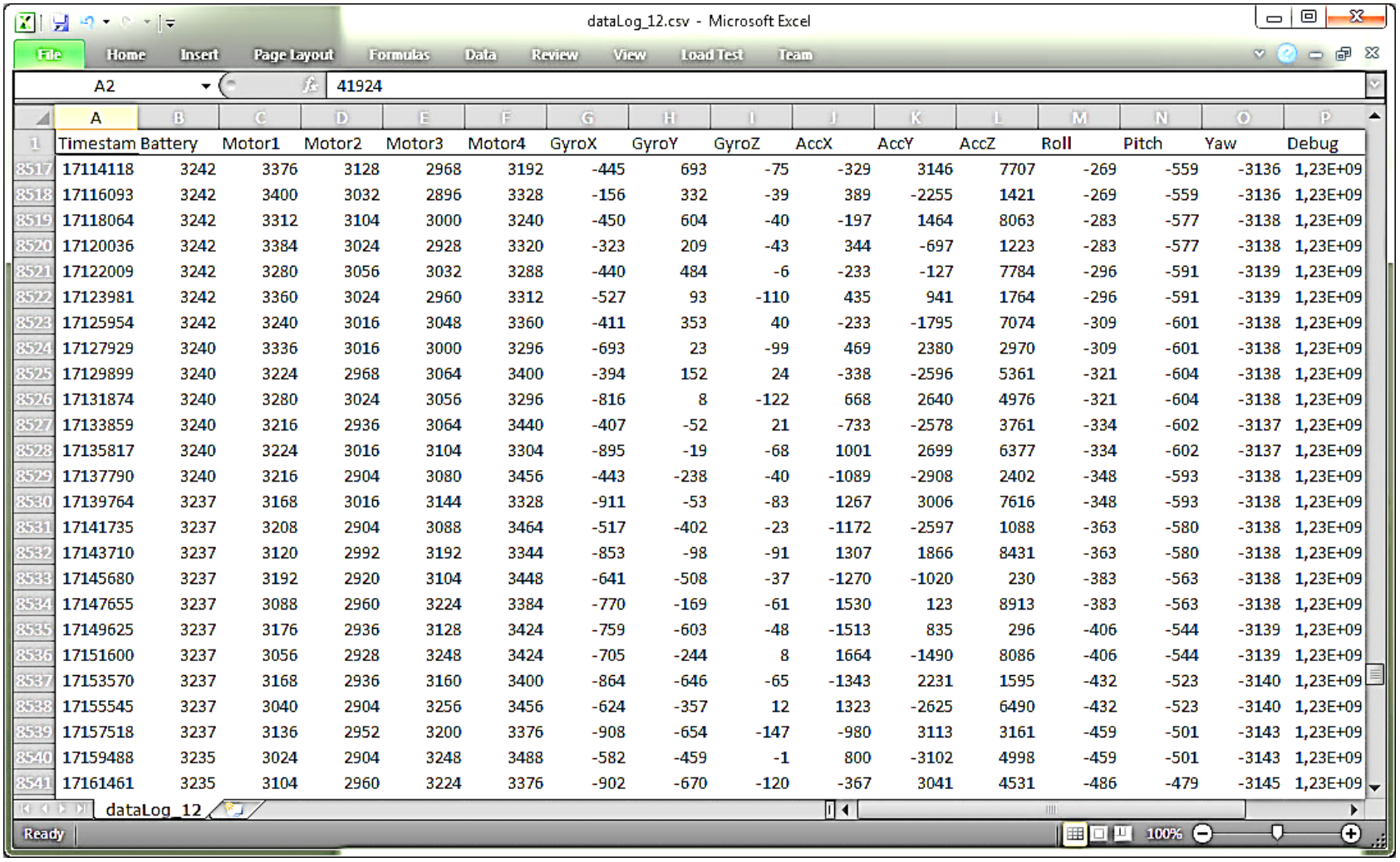
Figure 5.6 : The telemetry receiver.

5.5.2 Computer side

A computer program is implemented to communicate to the STM32F4-Discovery.

- The program uses LibUSB API to communicate over USB,
- It connects to the board and waits for data,
- Isochronous communication is used to maximize the transfer capacity,
- The received data is buffered and written to CSV files,

The CSV spreadsheet format is used to make it possible to open it in EXCEL and MATLAB. An example CSV file part is shown below.



dataLog_12.csv - Microsoft Excel

File Home Insert Page Layout Formulas Data Review View Load Test Team

A2 41924

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Timestam	Battery	Motor1	Motor2	Motor3	Motor4	GyroX	GyroY	GyroZ	AccX	AccY	AccZ	Roll	Pitch	Yaw	Debug
8517	17114118	3242	3376	3128	2968	3192	-445	693	-75	-329	3146	7707	-269	-559	-3136	1,23E+09
8518	17116093	3242	3400	3032	2896	3328	-156	332	-39	389	-2255	1421	-269	-559	-3136	1,23E+09
8519	17118064	3242	3312	3104	3000	3240	-450	604	-40	-197	1464	8063	-283	-577	-3138	1,23E+09
8520	17120036	3242	3384	3024	2928	3320	-323	209	-43	344	-697	1223	-283	-577	-3138	1,23E+09
8521	17122009	3242	3280	3056	3032	3288	-440	484	-6	-233	-127	7784	-296	-591	-3139	1,23E+09
8522	17123981	3242	3360	3024	2960	3312	-527	93	-110	435	941	1764	-296	-591	-3139	1,23E+09
8523	17125954	3242	3240	3016	3048	3360	-411	353	40	-233	-1795	7074	-309	-601	-3138	1,23E+09
8524	17127929	3240	3336	3016	3000	3296	-693	23	-99	469	2380	2970	-309	-601	-3138	1,23E+09
8525	17129899	3240	3224	2968	3064	3400	-394	152	24	-338	-2596	5361	-321	-604	-3138	1,23E+09
8526	17131874	3240	3280	3024	3056	3296	-816	8	-122	668	2640	4976	-321	-604	-3138	1,23E+09
8527	17133859	3240	3216	2936	3064	3440	-407	-52	21	-733	-2578	3761	-334	-602	-3137	1,23E+09
8528	17135817	3240	3224	3016	3104	3304	-895	-19	-68	1001	2699	6377	-334	-602	-3137	1,23E+09
8529	17137790	3240	3216	2904	3080	3456	-443	-238	-40	-1089	-2908	2402	-348	-593	-3138	1,23E+09
8530	17139764	3237	3168	3016	3144	3328	-911	-53	-83	1267	3006	7616	-348	-593	-3138	1,23E+09
8531	17141735	3237	3208	2904	3088	3464	-517	-402	-23	-1172	-2597	1088	-363	-580	-3138	1,23E+09
8532	17143710	3237	3120	2992	3192	3344	-853	-98	-91	1307	1866	8431	-363	-580	-3138	1,23E+09
8533	17145680	3237	3192	2920	3104	3448	-641	-508	-37	-1270	-1020	230	-383	-563	-3138	1,23E+09
8534	17147655	3237	3088	2960	3224	3384	-770	-169	-61	1530	123	8913	-383	-563	-3138	1,23E+09
8535	17149625	3237	3176	2936	3128	3424	-759	-603	-48	-1513	835	296	-406	-544	-3139	1,23E+09
8536	17151600	3237	3056	2928	3248	3424	-705	-244	8	1664	-1490	8086	-406	-544	-3139	1,23E+09
8537	17153570	3237	3168	2936	3160	3400	-864	-646	-65	-1343	2231	1595	-432	-523	-3140	1,23E+09
8538	17155545	3237	3040	2904	3256	3456	-624	-357	12	1323	-2625	6490	-432	-523	-3140	1,23E+09
8539	17157518	3237	3136	2952	3200	3376	-908	-654	-147	-980	3113	3161	-459	-501	-3143	1,23E+09
8540	17159488	3235	3024	2904	3248	3488	-582	-459	-1	800	-3102	4998	-459	-501	-3143	1,23E+09
8541	17161461	3235	3104	2960	3224	3376	-902	-670	-120	-367	3041	4531	-486	-479	-3145	1,23E+09

dataLog_12

Ready 100%

Figure 5.7 : Data log example.

6. INERTIAL MEASUREMENT UNIT

Inertial measurement unit (IMU), the acronym for inertial and navigational sensors, is one of the crucial parts for unmanned aerial vehicles. Its basic task is to provide information for the orientation and position of the vehicle.

Today's UAV systems are equipped with state of the art micro electro-mechanical sensors (MEMS), including accelerometers, gyroscopes and magnetometers.

Sensing the orientation and position is not possible using only one sensor, but requires data from more than one type of sensors and big computational effort. Some of these sensor pairs and triplets are given below (Table 6.1).

Table 6.1 : IMU sensor groups.

Measurement	Required Sensors
6DOF Attitude (yaw drifts)	Accelerometer + Gyroscope
9DOF Attitude	Accelerometer + Gyroscope + Magnetometer
Low Altitude	SODAR (Sonar Range Finder)
All Altitude	SODAR + Barometer
3D Position	SODAR + Barometer + GPS or Camera

There are several techniques, called sensor fusion algorithms, to obtain required measurements using more than one sensor's data, some of which are given below.

- Complementary filter,
- Kalman sensor fusion,
- Numeric fusion algorithms,

The Crazyflie quadrotor has MPU6050 sensor, which includes a 3-axis accelerometer and a 3-axis gyroscope, as explained in chapter 4.

According to Table 6.1, only 6DOF attitude is available; which means roll and pitch axes are referenced to the earth surface while the yaw axis is relative to the initial direction, thus is prone to drifts.

6.1 IMU Sensors Explained

6.1.1 Accelerometer data explained

An accelerometer is a sensor that measures the acceleration parallel with its axes. For a 3-axis accelerometer, the measurement is done in all axes of the Cartesian coordinate system. It represents a proper acceleration vector giving numerical values (a_X, a_Y and a_Z) that correspond to the projections of the vector onto its axes (Figure 6.1).

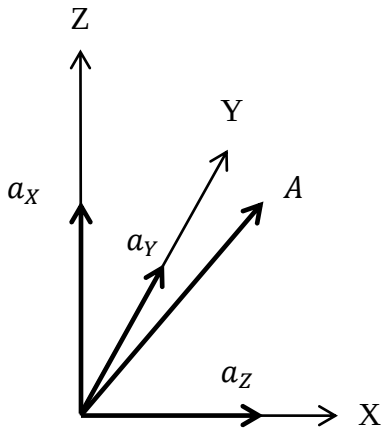


Figure 6.1 : Vector output of a 3-axis accelerometer.

6.1.2 Getting Euler angles

Accelerometers can sense the gravitational acceleration, thus the orientation of the body. Euler rotation angles can be obtained from a 3-axis accelerometer data using the equations below. It is not possible to measure yaw from an accelerometer.

$$\phi_{ACCEL} = tg^{-1}\left(\frac{a_Y}{a_Z}\right) \quad \textbf{(6.1)}$$

$$\theta_{ACCEL} = tg^{-1}\left(\frac{-a_X}{\sqrt{a_Y^2 + a_Z^2}}\right) \quad \textbf{(6.2)}$$

(Only applies to the accelerometers that give $a_Z = +1g$ when the z-axis points up.)

Accelerometers measure any acceleration including the one caused by motion. This makes the use of an accelerometer to measure attitude angles of a non-stationary body (like a quadrotor) impossible. Moreover, accelerometers are prone to high levels of measurement noise and usage of averaging filter that causes delay is crucial.

6.1.3 Gyroscope data explained

A gyroscope is a sensor that measures the rotational velocities (attitude rates) about its axes (assuming the inertial frame is stationary in space). A 3-axis gyroscope measures p, q and r rotational velocities shown in the figure below (Figure 6.2).

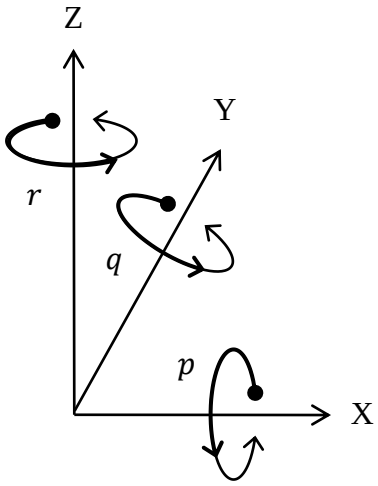


Figure 6.2 : Output of a 3-axis gyroscope.

6.1.4 Getting Euler rates

A gyroscope measures angular velocities about its own axes. A transformation is required to obtain Euler rates about the axes of the inertial frame. Rewriting (3.11):

$$\begin{bmatrix} \dot{\phi}_{GYRO} \\ \dot{\theta}_{GYRO} \\ \dot{\psi}_{GYRO} \end{bmatrix} = \begin{bmatrix} 1 & Sin\phi Tan\theta & Cos\phi Tan\theta \\ 0 & Cos\phi & -Sin\phi \\ 0 & Sec\theta Sin\phi & Cos\phi Sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{6.3}$$

It is possible to integrate Euler rates to obtain relative angles.

$$\Delta\phi_{GYRO} = \int \dot{\phi} dt \tag{6.4}$$

$$\Delta\theta_{GYRO} = \int \dot{\theta} dt \tag{6.5}$$

$$\Delta\psi_{GYRO} = \int \dot{\psi} dt \tag{6.6}$$

Gyroscopes are immune to noise sources, but prone to offset and integral drifts. They cannot be used to measure absolute orientation.

6.2 Sensor Fusion (6DOF)

The Crazyflie quadrotor uses a complex numeric algorithm for sensor fusion, which is not touched in this work. On the other hand, the most basic fusion algorithms will be explained instead.

6.2.1 Complementary filter

The most basic filter that can be used to obtain attitude angles is called complementary filter.

Actually, it is an ordinary averaging filter that uses pure accelerometer data for initial orientation (assuming stationary) with respect to equations (6.1) and (6.2), then uses a weighted average of accelerometer data and time integration of gyroscope data.

The initial state is given below.

$$\begin{bmatrix}\phi_0\\ \theta_0\\ \psi_0\end{bmatrix} = \begin{bmatrix}\phi_{ACCEL}\\ \theta_{ACCEL}\\ 0\end{bmatrix} \tag{6.7}$$

Once initialized, the following equations apply; where $W \in (0,1)$ is filter weight.

$$\phi_k = W(\Delta\phi_{GYRO} + \phi_{k-1}) + (1-W)\phi_{ACCEL} \tag{6.8}$$

$$\theta_k = W(\Delta\theta_{GYRO} + \theta_{k-1}) + (1-W)\theta_{ACCEL} \tag{6.9}$$

$$\psi_k = \Delta\psi_{GYRO} + \psi_{k-1} \tag{6.10}$$

Generally, W is selected close to 1 to suppress accelerometer errors. But this time gyroscope drifts become a major problem.

6.2.2 Kalman sensor fusion

Kalman sensor fusion is a technique based on Kalman Filter that is used for integrating accelerometer and gyroscope data.

The basic structure of a Kalman filter is given below, where K is dynamic Kalman gain, z is the measured and $\hat{x}$ is the estimated value of the physical quantity [38].

$$\hat{x}_k = K_k z_k + (1 - K_k)\hat{x}_{k-1} \tag{6.11}$$

It should be noted that (6.11) is very similar to (6.8) and (6.9).

The heart of the filter that calculates the gain is consists of two parts: prediction and correction.

The prediction step is given below, where u is the input, P is the error covariance and Q is the process noise matrix while A and B represent the system matrices.

$$\hat{x}_k = A\hat{x}_{k-1} + Bu_k \tag{6.12}$$

$$P_k = AP_{k-1}A^T + Q \tag{6.13}$$

The correction step is given below, where R is the measurement noise matrix and H is the output matrix.

$$K_k = P_k H^T \left(HP_k H^T + R\right)^{-1} \tag{6.14}$$

$$\hat{x}_k = \hat{x}_k + K_k\left(z_k - H\hat{x}_k\right) \tag{6.15}$$

$$P_k = (I - K_k H)P_k \tag{6.16}$$

To fuse accelerometer and gyroscope sensors, the low noise gyroscope data is used as the system input (u) while the erroneous accelerometer data is used as the measurement (z). Using (6.1), (6.2) and (6.3):

$$u_\phi = \dot{\phi}_{GYRO}, \qquad u_\theta = \dot{\theta}_{GYRO}, \qquad u_\psi = \dot{\psi}_{GYRO} \tag{6.17}$$

$$z_\phi = \phi_{ACCEL}, \qquad z_\theta = \theta_{ACCEL} \tag{6.18}$$

As accelerometer cannot sense the yaw axis, correction step does not apply to it [38].

$$A = \begin{bmatrix} 1 & -T_S \\ 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} T_S \\ 0 \end{bmatrix} \tag{6.19}$$

$$\hat{x}_\phi = \begin{bmatrix} \phi \\ bias_\phi \end{bmatrix}, \qquad \hat{x}_\theta = \begin{bmatrix} \theta \\ bias_\theta \end{bmatrix}, \qquad \hat{x}_\psi = \begin{bmatrix} \psi \\ bias_\psi \end{bmatrix} \tag{6.20}$$

The system model is kept simple taking only the Euler angles and gyroscope biases [38]. The noise matrices are given below.

$$Q = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}, \qquad R = 5 \tag{6.21}$$

The process noise, which is related to the gyroscope errors (including drift), should be kept small, because of the low noise profile of the gyroscope. But the measurement noise comes from the accelerometer is very high, because the accelerometer is prone to noise and its measurement is erroneous when vibrations present. Q and R matrices can be tuned to get better results.

A test data is collected doing the following steps.

- Put the quadrotor onto a flat.
- Pick it up and do some basic rotations by hand.
- Put the quadrotor to exactly the same (initial) place.

Thus, the real values of the initial and the final angles are the same. The results (Figure 6.3) show the output of the designed Kalman fusion versus the raw data.

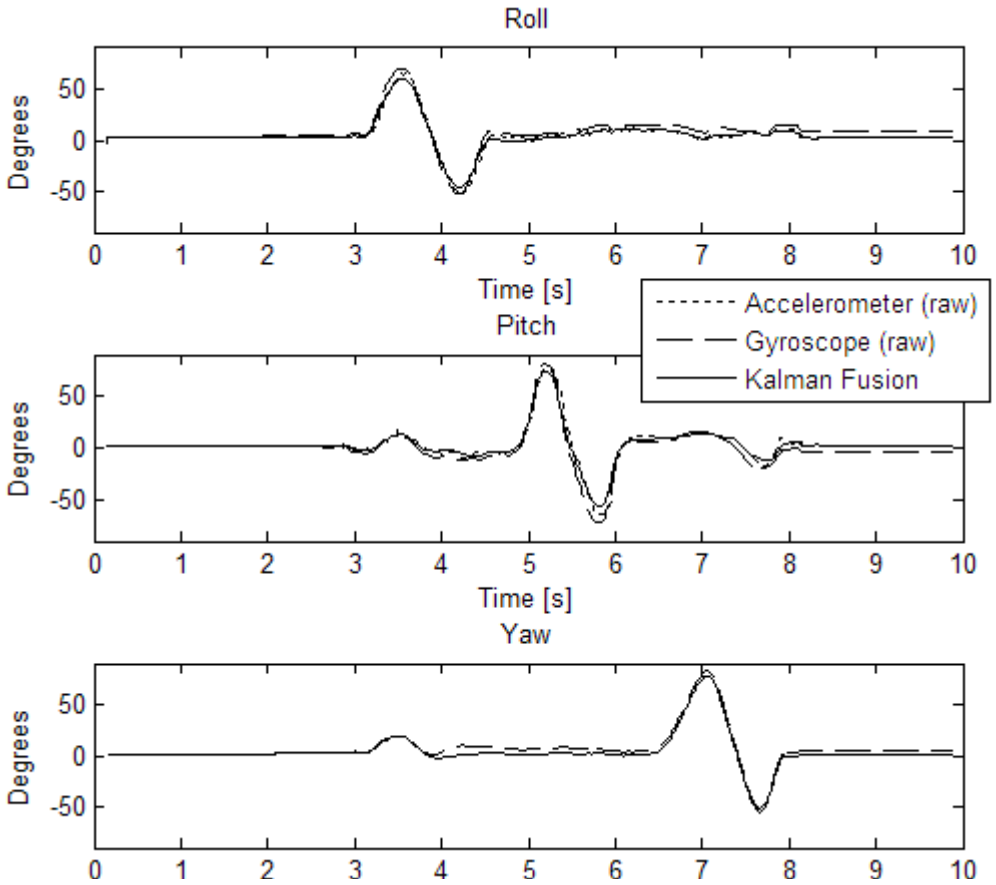


Figure 6.3 : Kalman fusion output.

It is seen that Kalman fusion algorithm cancels out the gyroscope bias drift and accelerometer noise with success. The initial and the final angles match perfectly.

6.3 Verification of the Quadrotor Axes

A Crazyflie quadrotor is used in this work. To verify its axes and rotation directions, some tests are done by hand (not flying) with respect to the following methodology.

- A positive (left is up, right is down) then negative rotation about the roll axis,
- A positive (front is down, back is up) then negative rotation about the pitch,
- A positive (counter-clockwise) then negative rotation about the yaw axis,

The results of the default Crazyflie firmware is given below (Figure 6.4).

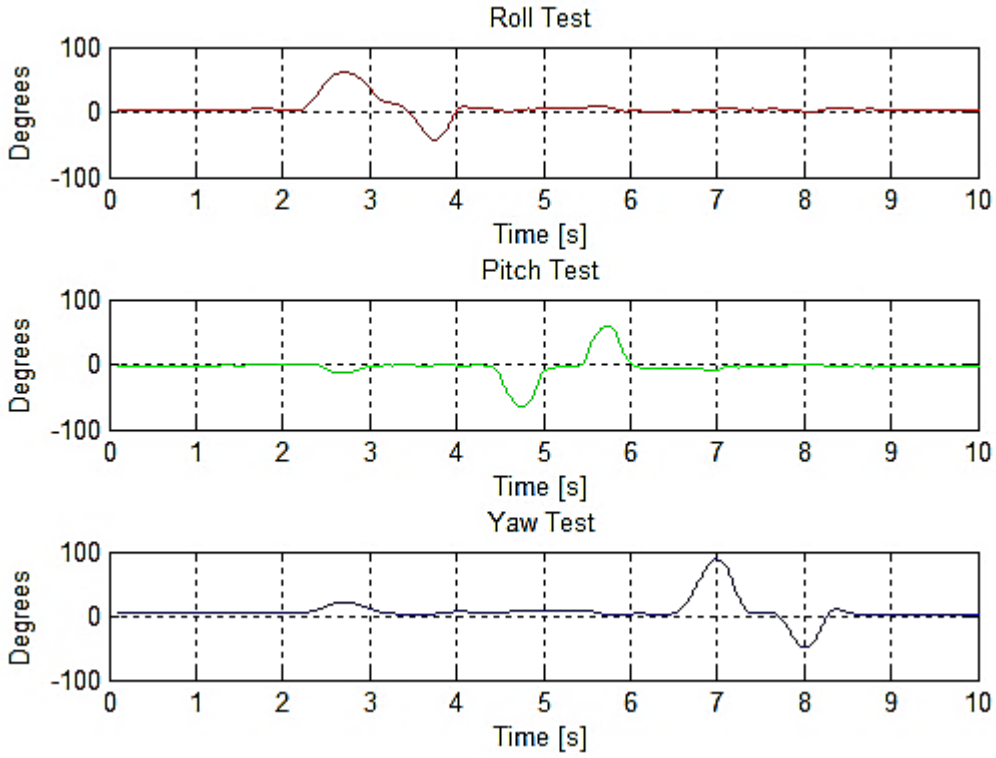


Figure 6.4 : Crazyflie axis test no.1.

It can be seen that the pitch axis is inverted in the default firmware and it should be negated again. Results of the test with corrected axes are given below (Figure 6.5).

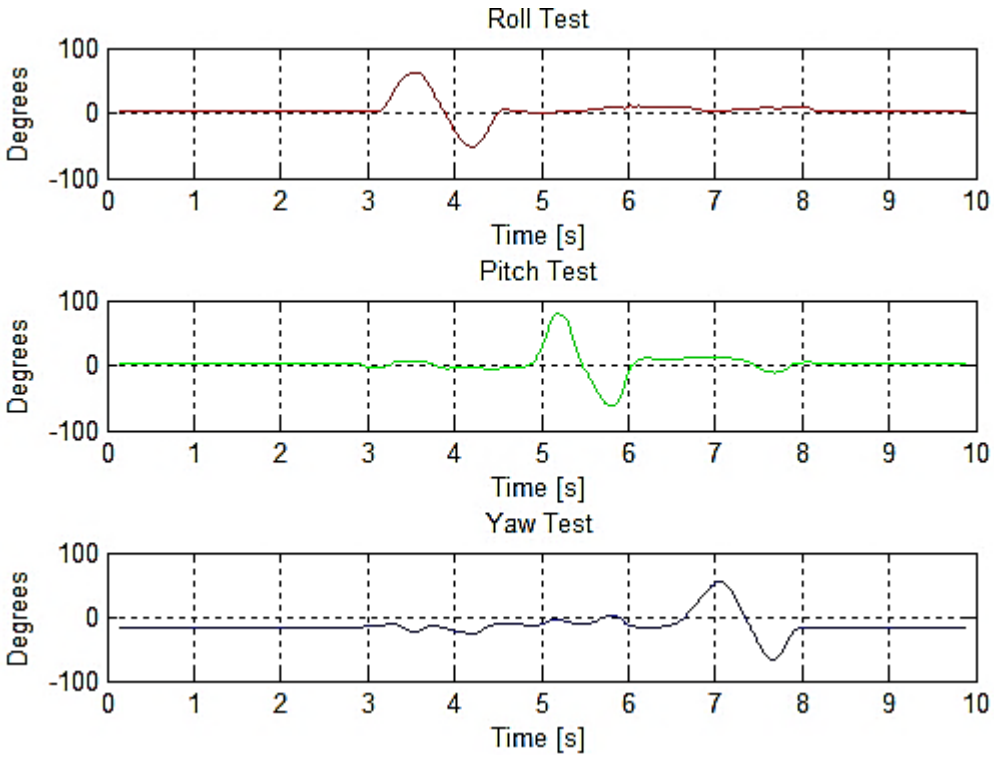


Figure 6.5 : Crazyflie axis test no.2.

7. SYSTEM IDENTIFICATION

One of the goals of this work is identify the exact dynamical model of the quadrotor platform using real-time flight data collected via the telemetry system described in chapter 5.1.

In this chapter, some of the identification techniques used on the quadrotor will be explained in details.

7.1 Identification Approaches

There are many identification techniques to find the undergoing dynamical structure of an unknown or semi-known system. Some of these techniques are given below.

- Linear grey-box models,
- Nonlinear grey-box models,
- Linear auto-regressive (AR, ARX, ARMA, ARMAX) models,
- Nonlinear adaptive neuro-fuzzy (ANFIS) models,

All of the quadrotor subsystems given in Figure 3.1 are nonlinear and the best choice will be nonlinear identification methods. However, some assumptions that give flexibility to use linear methods instead are possible.

Now, some of these methods will be explained.

7.1.1 Grey-Box identification

When the structure of the model of a dynamical system is known but the parameters (or coefficients) are unknown, the model is said to be a grey-box model. Grey-box models can be both linear and nonlinear.

The unknown parameters of the grey-box models can be obtained from collected input-output data of the real system, via identification methods. One of these methods is prediction error method (PEM). It is known that PEM gives satisfactory results in closed-loop systems [20].

7.1.2 Prediction error method (PEM)

The prediction error method (a.k.a. prediction error minimization) is a well-known method that is used for obtaining unknown parameters of a model. A brief description will be given.

The assumed real system and the candidate model are given below.

$$y(t) = P_0(q)u(t) + v_0(t) \tag{7.1}$$

$$y(t) = P(q,Q)u(t) + H(q,Q)w(t) \tag{7.2}$$

Where Q is parameter vector, $v_0(t)$ is disturbance, $w(t)$ is white noise, $P(q,Q)$ and $P_0(q)$ are output transfer functions, $H(q,Q)$ is noise transfer function.

The 1-step-ahead linear predictor based on input-output data up to time $(t-1)$ can be expressed as follows.

$$\hat{y}(t,Q) = H^{-1}(q,Q)P(q,Q)u(t) + [1 - H^{-1}(q,Q)]y(t) \tag{7.3}$$

The prediction error can be expressed as follows.

$$e(t,Q) \triangleq y(t) - \hat{y}(t,Q) \tag{7.4}$$

$$e(t,Q) = H^{-1}(q,Q)[P_0(q) - P(q,Q)]u(t) + H^{-1}(q,Q)v_0(t) \tag{7.5}$$

Finally, the prediction error for N data samples is given by the following cost function that can be minimized using iterative descent algorithms [40].

$$V_N(Q) = \frac{1}{N}\sum_{t=1}^{N} e^2(t,Q) \tag{7.6}$$

7.1.3 Auto-regressive exogenous identification

ARX, the abbreviation for an auto-regression model with exogenous input, is a black-box model that is used to identify unknown systems based on collected input-output data.

A linear dynamical system can be expressed by the following polynomial model where $w(t)$ is white noise, $G(q,Q)$ is output transfer function and $H(q,Q)$ is noise transfer function. [41]

$$G \triangleq \frac{B}{F}, \qquad H \triangleq \frac{C}{A} \tag{7.7}$$

$$y(t) = q^{-k}G(q,Q)u(t) + H(q,Q)w(t) \tag{7.8}$$

Setting F and C to 1, the following ARX model is obtained (Figure 7.1).

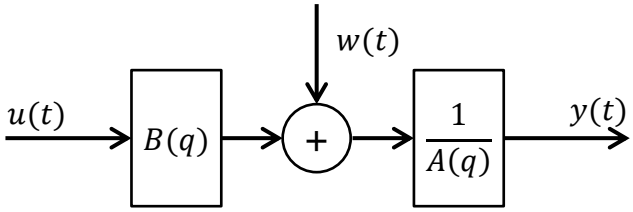


Figure 7.1 : ARX model structure.

The ARX model can be expressed by the following polynomial where na is the number of poles, $nb-1$ is the number of zeros and nk is pure time delay.

$$\begin{aligned} y(t) + a_1 y(t-1) + \cdots + a_{na} y(t-na) \\ = b_1 y(t-nk) + \cdots + b_{nb} y(t-nk-nb+1) + w(t) \end{aligned} \tag{7.9}$$

The model parameters a_i and b_i are found using least-squares method to minimize the variation between the ARX model and the measured input-output data.

7.2 Identification of Motor Parameters

The motor model (Figure 7.2) is obtained in equation (3.82) as a nonlinear, first order dynamical model. The nonlinearity term comes from the propeller dynamics, which cannot be neglected.

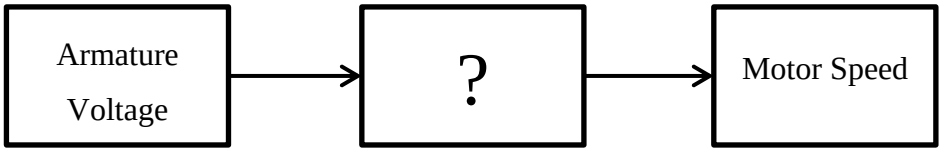


Figure 7.2 : Motor model identification scheme.

7.2.1 Methodology for collecting data

A quadrotor has four motors, which are assumed identical. It will be reasonable to identify one of them and use the identified model for all.

Crazyflie has no hardware support to measure motor speed. For this reason, an additional circuit is implemented as described in chapter 5.2.

The motor speed sensor causes unbalanced payload and Crazyflie is not able to fly with the sensor mounted. For this reason, motor speed versus armature voltage data is collected as Crazyflie is on the ground.

The following methodology is used to collect open-loop identification data.

- Speed measurement sensor is connected to one of the motors.
- Crazyflie is fixed to on top of a test bench to allow free airflow.
- Data logging software is started.
- Constantly varying arbitrary thrust input is applied through the joystick.
- The CSV file is read through Matlab and data is parsed into variables.

7.2.2 Pre-processing of the collected data

The collected data is shown below (Figure 7.3).

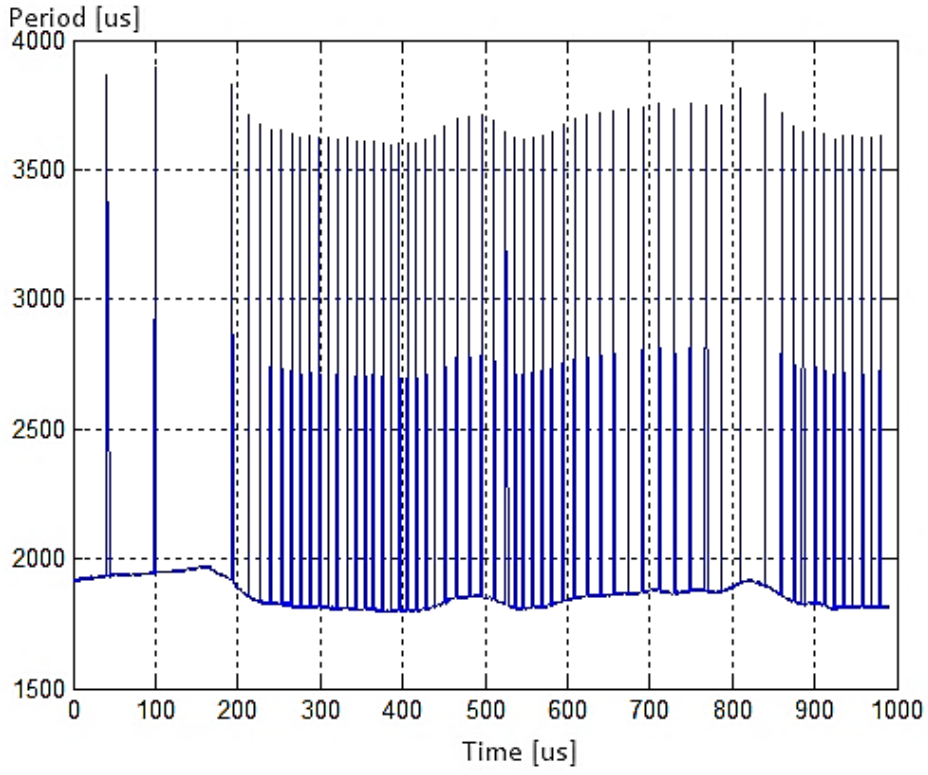


Figure 7.3 : Logged motor revolution period data.

The logged data (given in Table 5.1) does not include motor speed but only the times when a rotor blade pass from inside of the optocoupler. This data yields the time that corresponds to half of a revolution period, shown in the graph above.

As clearly seen in the graph, there are big glitches because of the insufficient sampling period of the speed sensor. While there is no way to prevent this phenomena, to get rid of the glitches a smart filter is implemented. The filtered period and motor speed graphs are given below (Figure 7.4 and Figure 7.5).

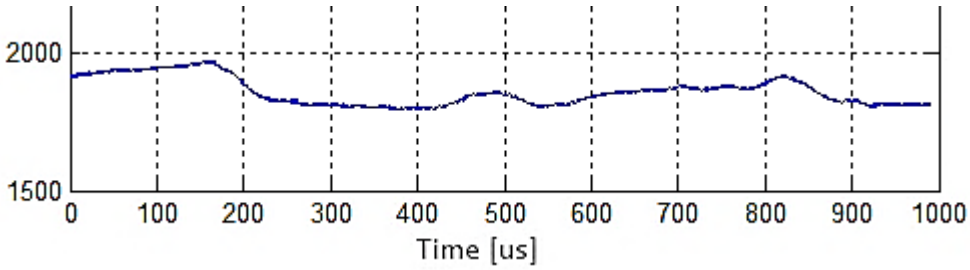


Figure 7.4 : Filtered motor revolution period data.

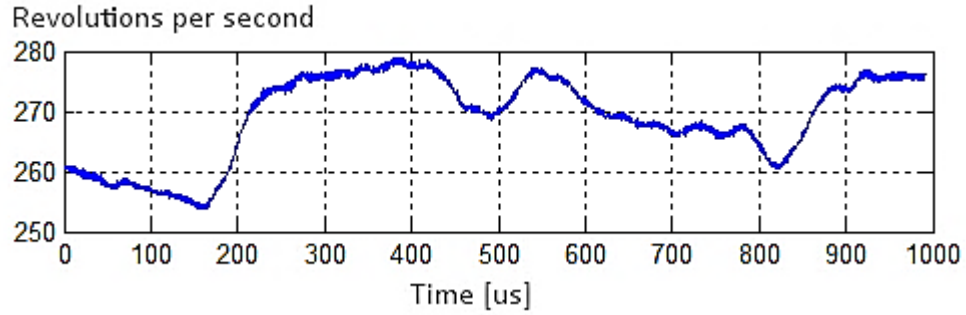


Figure 7.5 : Measured motor speed.

The input data is consists of PWM duty and battery voltage. To obtain armature voltage of the motor, equation (3.84) is used. The result is given below (Figure 7.6).

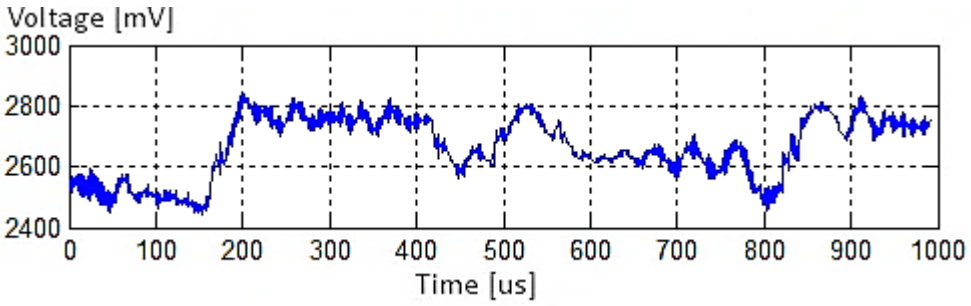


Figure 7.6 : Armature voltage.

All of the collected motor data is processed according to this procedure and saved as *iddata* objects in Matlab.

7.2.3 Identification process

Identification of nonlinear motor model is done via nonlinear grey-box identification tools available in Matlab. To use these tools a Matlab function (Figure 7.7) is implemented according to the continuous time nonlinear motor model given in equation (3.82).

Figure 7.7 : Block diagram of motor model.

In Matlab, an *idnlgrey* object is created using the prepared *iddata* object and *QuadrotorMotor_m* function.

Name of the Matlab function that is used to identify grey-box model parameters using prediction error minimization (PEM) is *pem*. It tries to minimize the mean square error using one of the nonlinear least squares, Gauss-Newton, Levenberg-Marquardt or gradient descent algorithms.

4000 samples are provided for the identification of 3 parameters. This process is shown below (Figure 7.8).

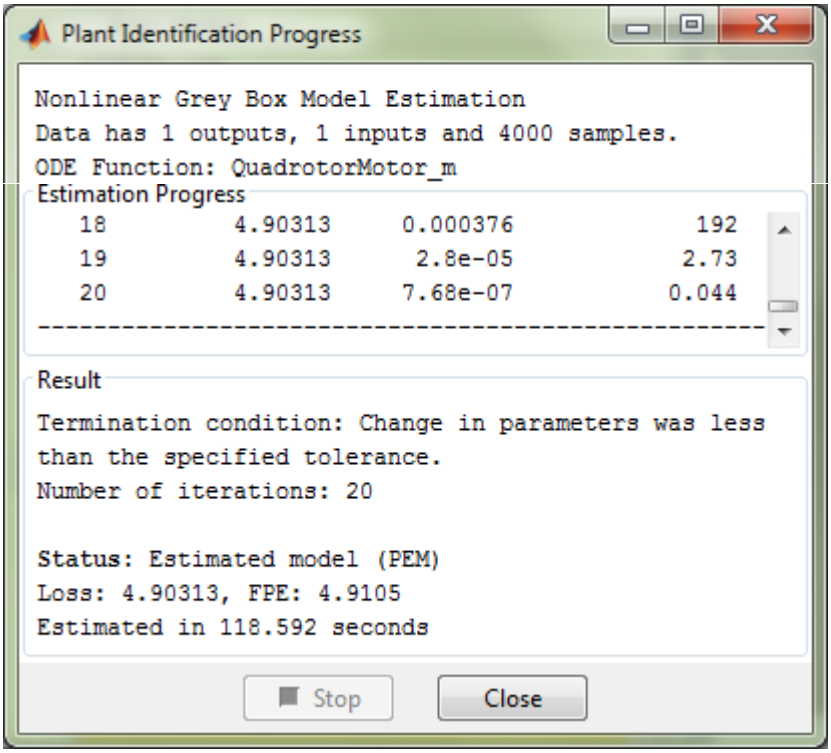


Figure 7.8 : Prediction error method (PEM) tool.

7.2.4 Identification results

The results of nonlinear grey-box identification are given below (Table 7.1).

Table 7.1 : Motor model identification results.

Data	Fit %
Identification	86.80
Validation-1	79.45
Validation-2	91.73

The identified model is given below.

$$\dot{x} = 0.862446u - 6.814507x - 0.006149x^2 \qquad \textbf{(7.10)}$$

Identification and validation data fits are shown below (Figure 7.9 - Figure 7.11).

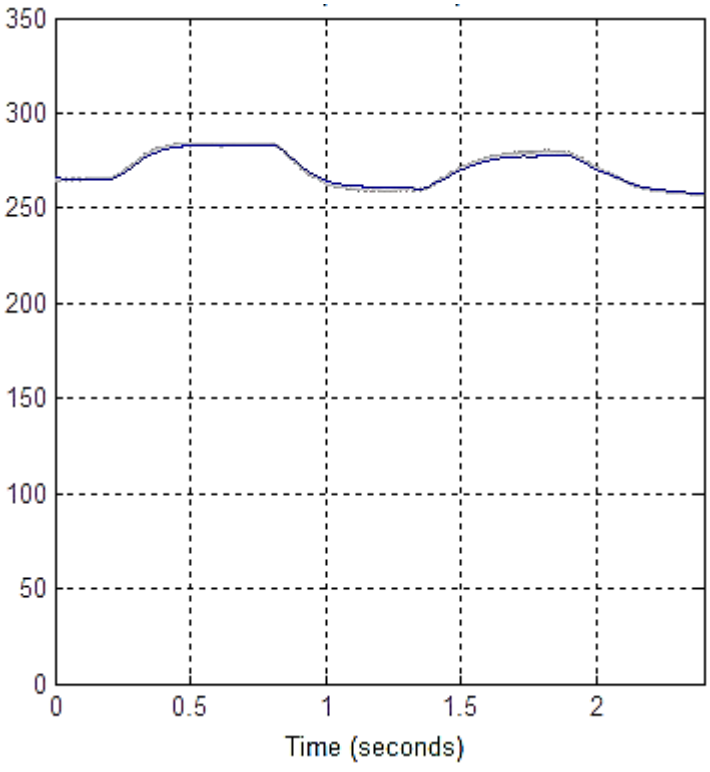


Figure 7.9 : Identification data fit.

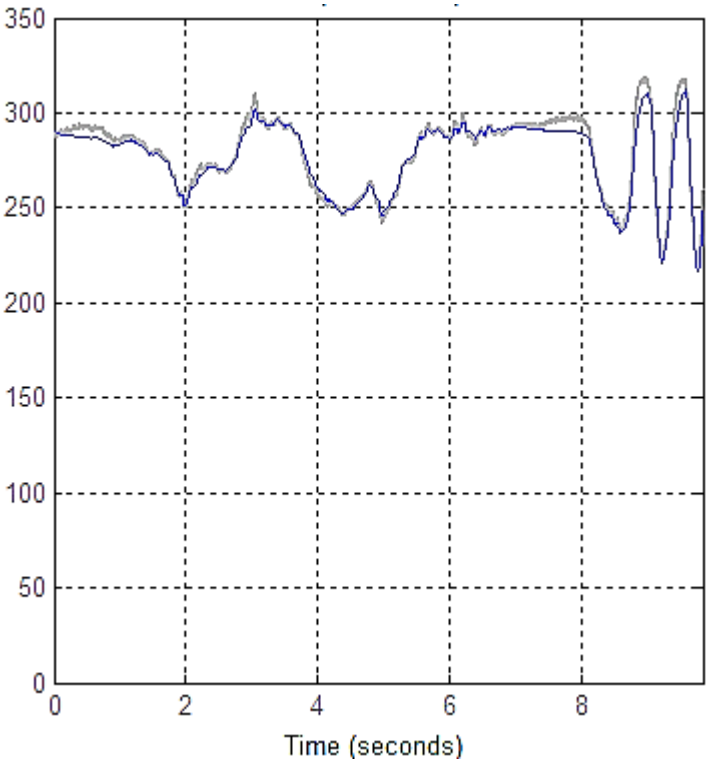


Figure 7.10 : Validation-1 data fit.

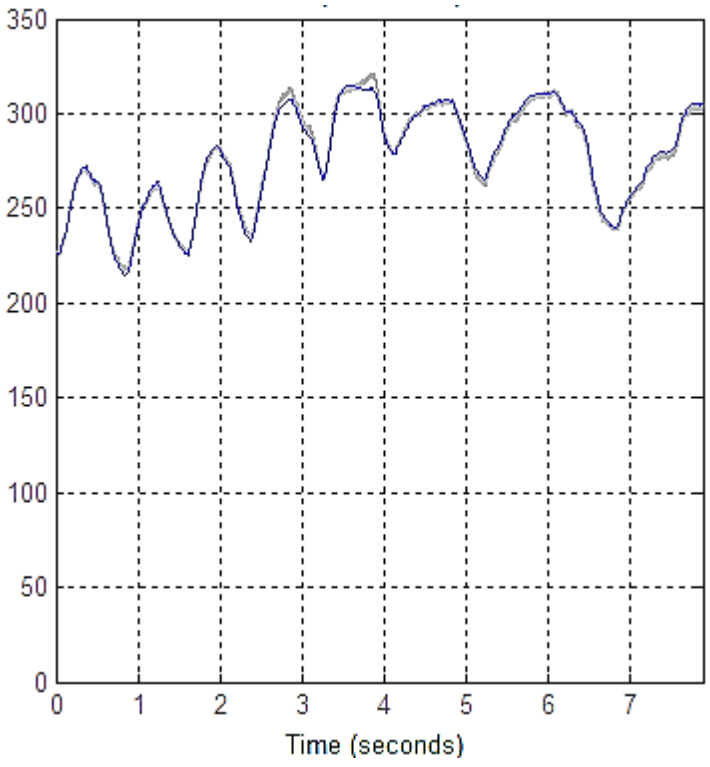


Figure 7.11 : Validation-2 data fit.

7.3 Identification of Quadrotor Body Model

Quadrotor body model will be identified using real-time flight data in this chapter.

7.3.1 Requirements for identification

Quadrotor is an unstable system that requires a closed-loop control to fly. In addition, the identification of a closed-loop system requires high excitation. The identification data must be collected considering this.

As described before, there is an accelerometer as well as a gyroscope on Crazyflie.

The accelerometer sensor is prone to high measurement noise coming from vibrations caused by blade flapping, electromagnetic inferences caused by motor drivers and offset drifts caused by temperature changes. Additionally, double integration of acceleration data causes integral drift. All of these effects prevent accurate translational measurements.

As system identification requires accurate and low-noise measurements of input-output data, it is possible to say that, identification of the translational quadrotor sub system is not possible without additional sensors.

Only the rotational model will be evaluated (Figure 7.12) within this work because of this hardware limitation.

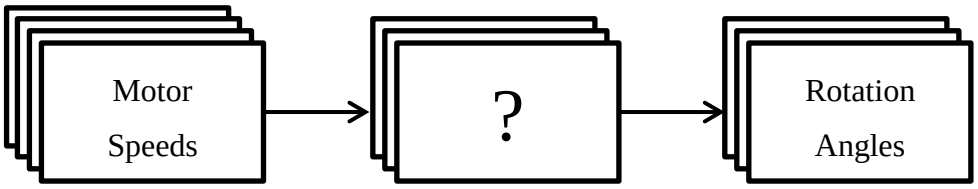


Figure 7.12 : Quadrotor body identification scheme.

7.3.2 Model selection

Several dynamical models of quadrotor body are obtained in chapter 3. As system identification is an iterative process, a computationally simple and less complex model gives better results.

The inertial frame model, which is obtained using Euler-Lagrange method, is extremely complex for system identification. However, it will be reasonable to use

the much simpler body frame model for identification. Furthermore, cross-axis effects prevent decoupled identification.

Thus, equations (3.58), (3.59) and (3.60) are suitable for identification of quadrotor body model.

7.3.3 Methodology for collecting flight data

The following methodology is used to collect identification data.

- Put Crazyflie on a flat surface.
- Wait at least 30 seconds for the sensors to stabilize.
- Start the data logging software.
- Apply thrust and take-off.
- Keep Crazyflie in hover position.
- Constantly change axes references; thus, excite Crazyflie axes well.
- Land safely and read the CSV file through Matlab.

7.3.4 Pre-processing of the flight data

As seen in Figure 7.12, quadrotor body identification requires rotational speeds of the motor. The identified motor speed model will be used to obtain this data from the armature voltages.

Motor armature voltages are shown below (Figure 7.13, Figure 7.14).

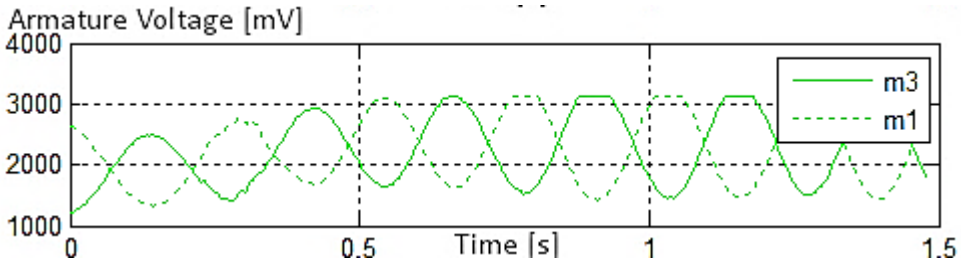


Figure 7.13 : Motor1 and Motor3 armature voltages.

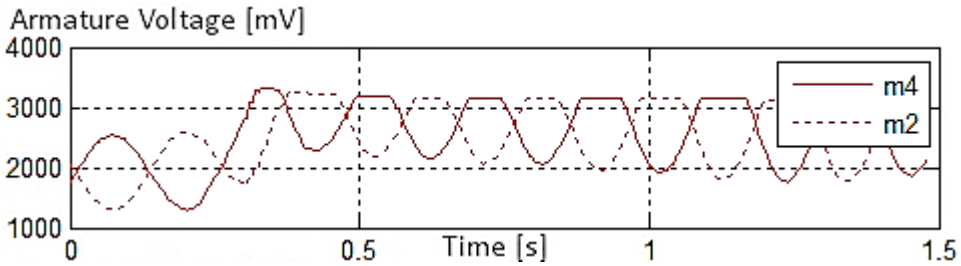


Figure 7.14 : Motor2 and Motor4 armature voltages.

In some regions, the tops of the curves are flat. This is caused by saturated PWM, i.e. 100% duty is applied. Thus the armature voltages are limited to battery voltage.

Separately, *iddata* structures are created using the armature voltages. Then nonlinear grey-box motor model is used to simulate them. Thus the input voltages are converted to motor speeds.

The resulting graphs are shown below (Figure 7.15, Figure 7.16).

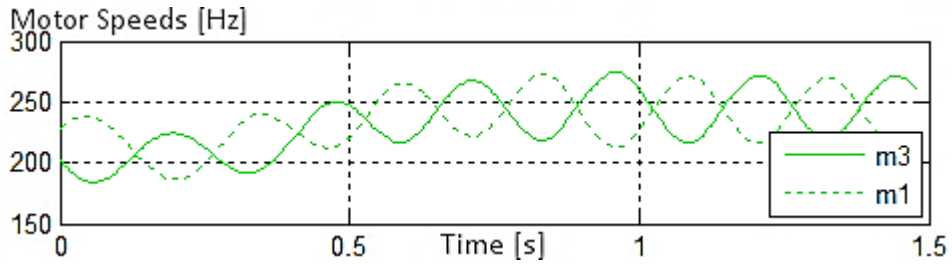


Figure 7.15 : Rotational speeds of Motor1 and Motor3.

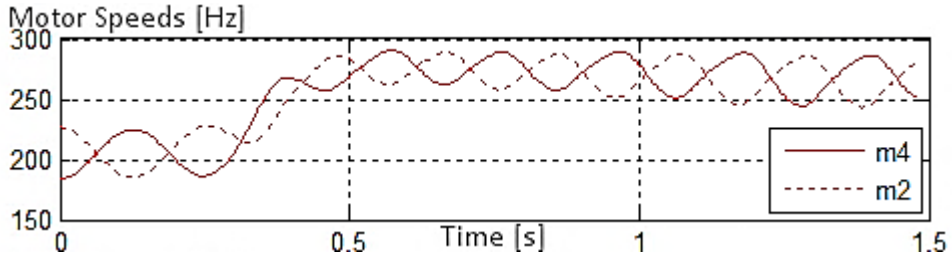


Figure 7.16 : Rotational speeds of Motor2 and Motor4.

To see the applied torques relative to the motor speeds, equation (3.21) can be used. This equation is called input mixer. It is possible to neglect *k* and *b* constants taking unity gains.

Roll torque is a function of the squares of Motor2 and Motor4 rotational speeds. The torque applied about roll axis is shown below (Figure 7.17).

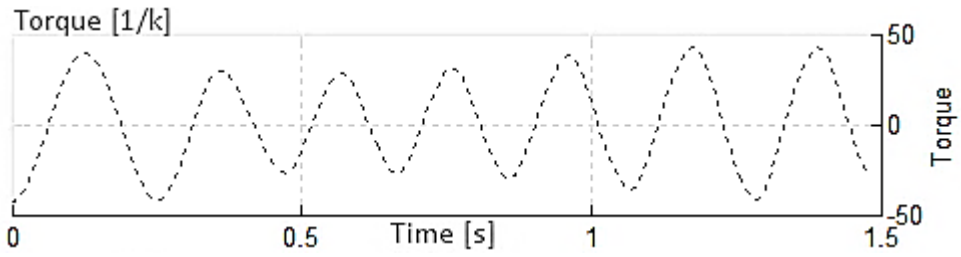


Figure 7.17 : Roll torque respect to the ***k*** constant.

Pitch torque is a function of the squares of Motor1 and Motor3 rotational speeds. The torque applied about pitch axis is shown below (Figure 7.18).

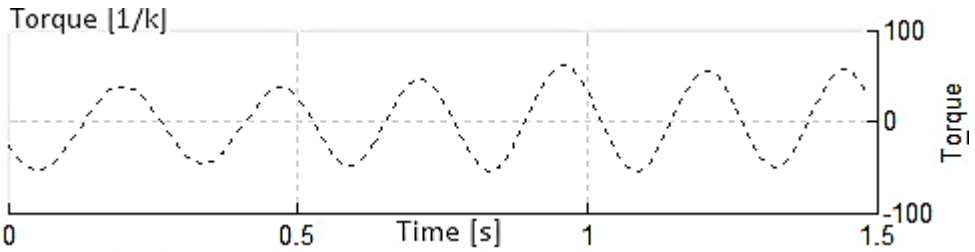


Figure 7.18 : Pitch torque respect to the ***k*** constant.

Yaw torque is a function of the squares of Motor1, Motor2, Motor3 and Motor4 rotational speeds. The torque applied about yaw axis is shown below (Figure 7.19).

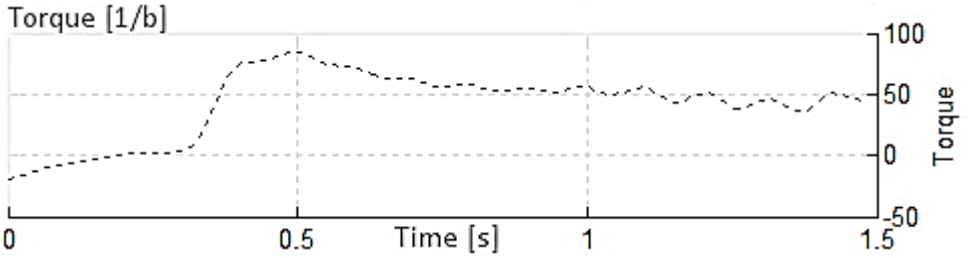


Figure 7.19 : Yaw torque respect to the ***b*** constant.

Finally, the Euler angles of the quadrotor will be used in identification process directly, without modification.

The tilt angle about roll axis is shown below (Figure 7.20).

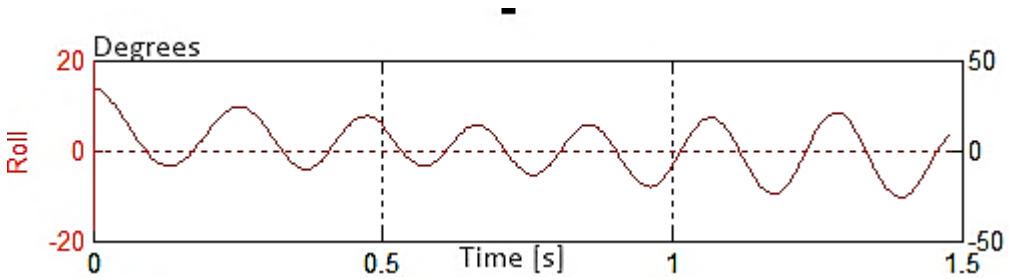


Figure 7.20 : Roll angle of the flight data.

The tilt angle about pitch axis is shown below (Figure 7.21).

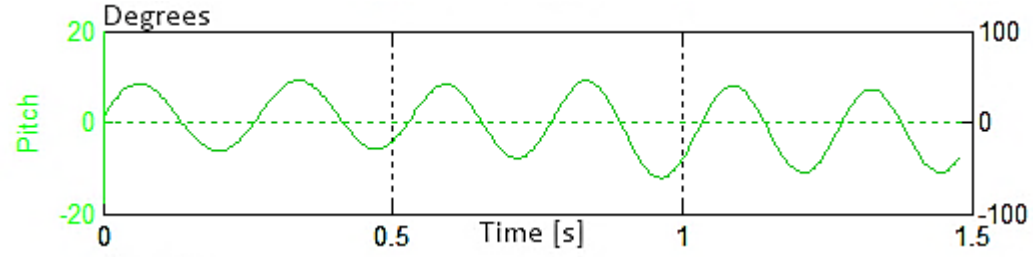


Figure 7.21 : Pitch angle of the flight data.

The position about yaw axis is shown below (Figure 7.22).

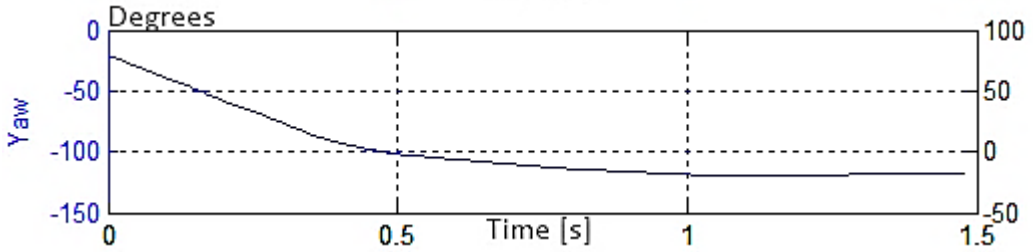


Figure 7.22 : Yaw angle of the flight data.

7.3.5 Real-Time flight datasets

Crazyflie is flown many times and rich flight datasets are collected. Sampling time (T_S) of the collected data is 2ms.

Some aerodynamic effects i.e. unregulated airflows, may cause significant changes in flight dynamics and make the identification impossible. Thus, the flight datasets are grouped in different lengths of data respect to some criteria, i.e. constant or changing altitude, stationary or moving, low or high altitude.

Some of the flight data examples that show Euler angles versus torques (dashed lines) are given below (Figure 7.23 - Figure 7.25).

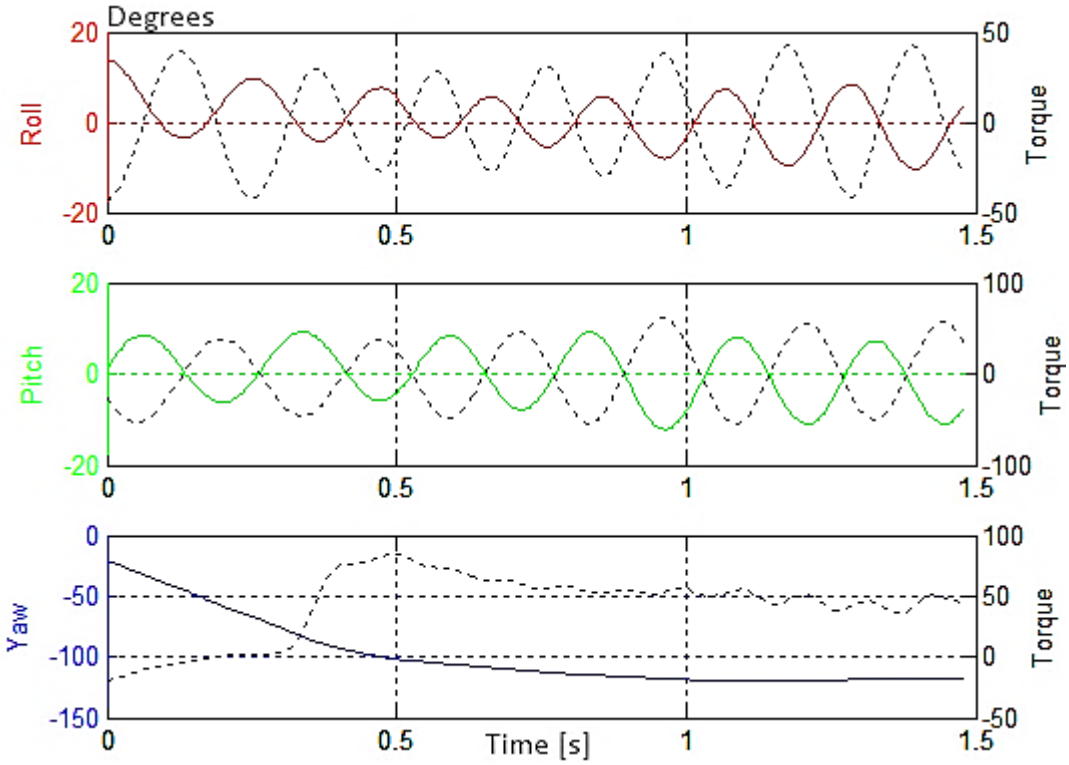


Figure 7.23 : Real-time flight data example-1.

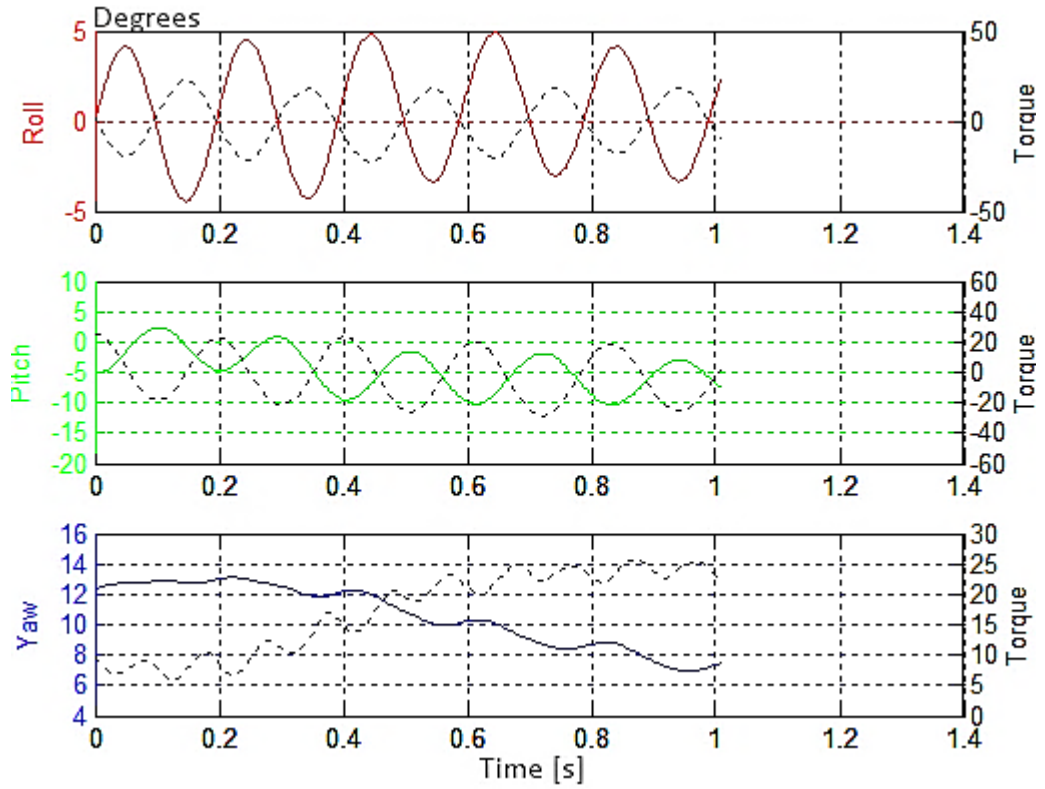


Figure 7.24 : Real-time flight data example-2.

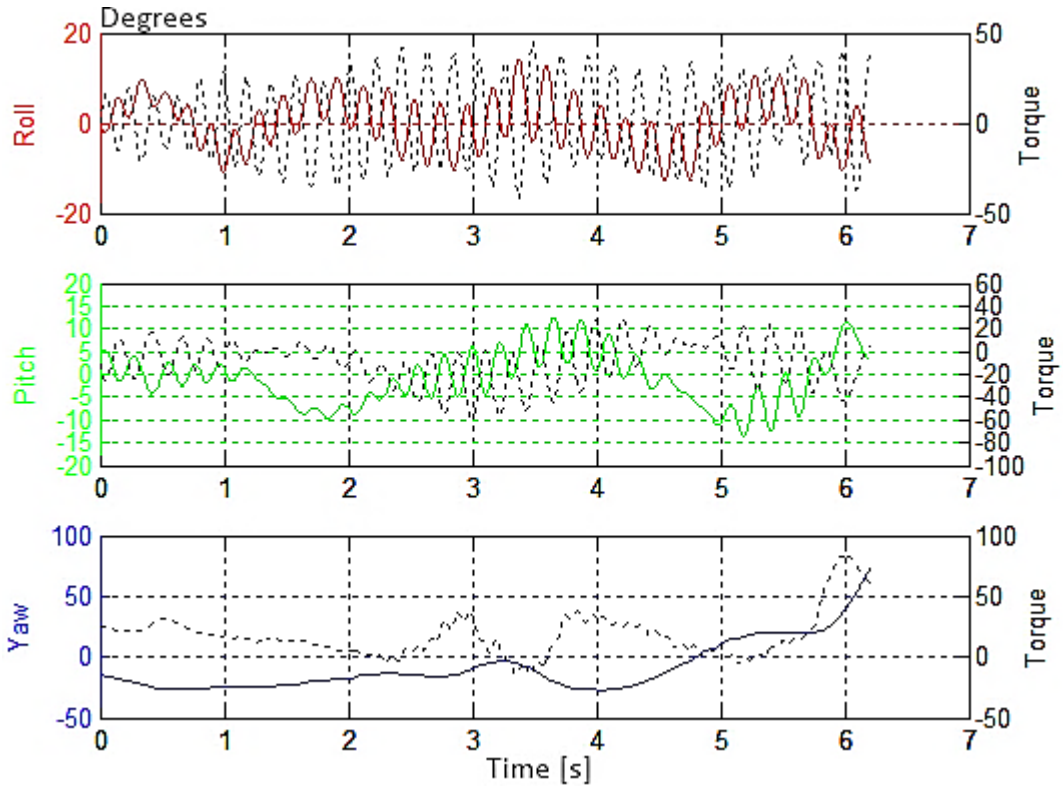


Figure 7.25 : Real-time flight data example-3.

All of the collected flight data is processed according to this procedure and saved as *iddata* objects in Matlab.

7.3.6 Nonlinear grey-box identification

Identification of quadrotor body model will be done via nonlinear grey-box tools.

7.3.6.1 Grey-box identification approach

Grey-box model is obtained using the explicit equations below. A grey-box model is implemented with respect to the equations (7.12) to (7.17), where the explicit input vector is (7.11) and the state vector is (3.61).

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \triangleq \begin{bmatrix} \omega_{M1} \\ \omega_{M2} \\ \omega_{M3} \\ \omega_{M4} \end{bmatrix} \tag{7.11}$$

$$\dot{x}_1 = c_0(u_4^2 - u_2^2) \tag{7.12}$$

$$\dot{x}_2 = c_1(u_3^2 - u_1^2) \tag{7.13}$$

$$\dot{x}_3 = c_2(-u_1^2 + u_2^2 - u_3^2 + u_4^2) \tag{7.14}$$

$$\dot{x}_4 = x_1 \tag{7.15}$$

$$\dot{x}_5 = x_2 \tag{7.16}$$

$$\dot{x}_6 = x_3 \tag{7.17}$$

Using the grey-box model, a nonlinear grey-box function (Figure 7.26) is implemented in Matlab.

Figure 7.26 : Block diagram of quadrotor body model.

In Matlab, an *idnlgrey* object is created using the prepared *iddata* object and *QuadrotorModel_m* function. Then prediction error minimization (PEM) method is used to train the grey-box model.

7.3.6.2 Identification results

A continuous-time grey-box model is obtained using the collected flight data.

As the PEM identification runs very slow with complex models, identification of roll, pitch and yaw axes done separately. Results of the PEM identification of nonlinear grey-box model will be given.

The identification and validation data fits of the roll model are given below (Figure 7.27, Figure 7.28).

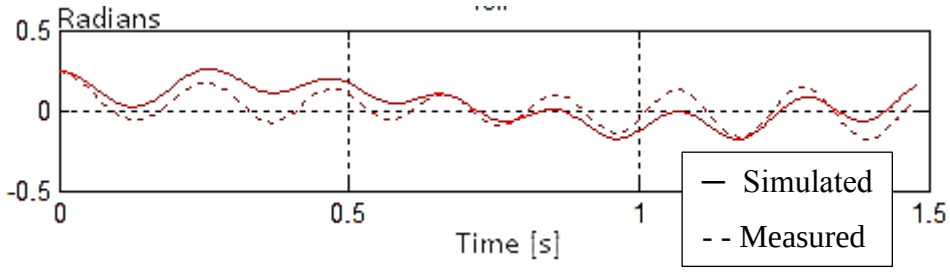


Figure 7.27 : Roll axis identification data fit.

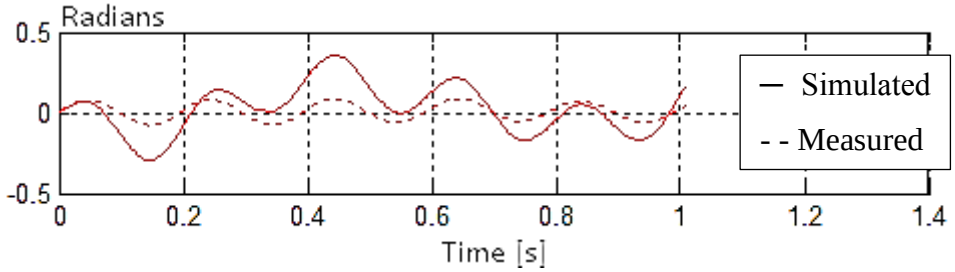


Figure 7.28 : Roll axis validation data fit.

Because of the structural symmetry of the quadrotor body, it is reasonable to use the roll axis model for also the pitch axis. There is no need to re-identify the pitch axis.

The identified model for the roll axis is simulated using the pitch axis inputs. The result is shown below (Figure 7.29).

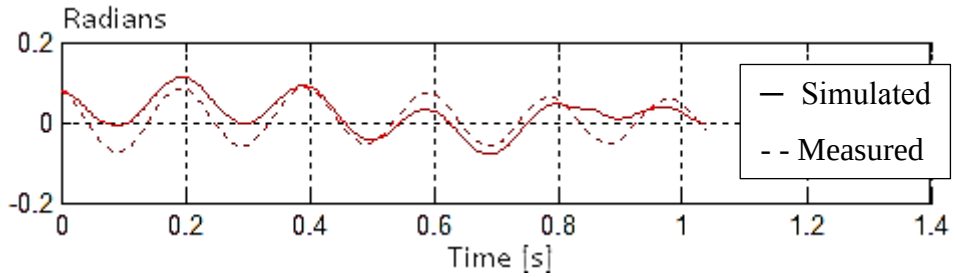


Figure 7.29 : Pitch axis validation data fit.

Yaw axis model has different dynamics. So, it must be identified separately. The results of the yaw axis identification are given below (Figure 7.30).

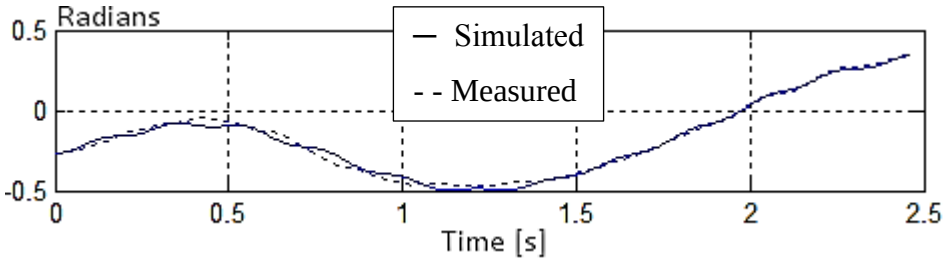


Figure 7.30 : Yaw axis identification data fit.

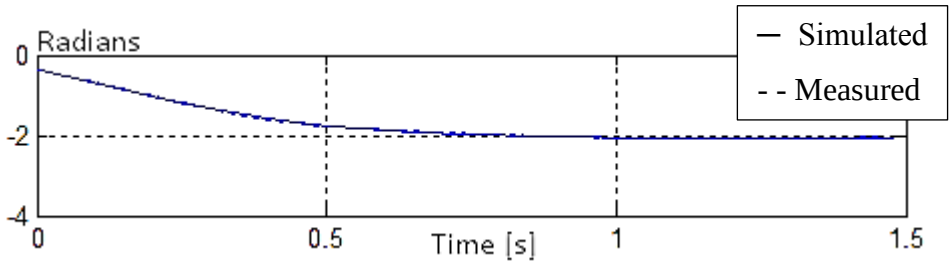


Figure 7.31 : Yaw axis validation data fit.

The table below (Table 7.2) gives the root mean square (RMS) error values of the identification and verification steps.

Table 7.2 : Results of nonlinear grey-box identification of the quadrotor.

Data Fit	RMS Error
Identification (roll)	0.0847
Validation (roll)	0.1209
Validation (pitch)	0.0345
Identification (yaw)	0.0230
Validation (yaw)	0.0181

Surprisingly, the roll axis model gave better results with the pitch axis model. With this result, it is proven that the pitch and roll axes models are said to be identical, and can be used for each other.

Finally, the identified constants of the nonlinear grey-box quadrotor body model are given below.

$$c_0 = 0.0136, \qquad c_1 = 0.0136, \qquad c_2 = 0.0005 \tag{7.18}$$

7.3.7 Polynomial ARX model identification

Identification of quadrotor body model will be done using an ARX black-box model.

7.3.7.1 ARX model identification approach

It is known that the system is highly nonlinear. The nonlinearity at the input vector given in (3.21) cannot be neglected. To use an ARX model for the highly nonlinear quadrotor body dynamics, it is required to present a nonlinear input mixer. The equation for the input mixer is given below.

$$u' = \begin{bmatrix} u'_1 \\ u'_2 \\ u'_3 \end{bmatrix} \triangleq \begin{bmatrix} u_4^2 - u_2^2 \\ u_3^2 - u_1^2 \\ -u_1^2 + u_2^2 - u_3^2 + u_4^2 \end{bmatrix} \tag{7.19}$$

Inputs of equations (7.12) to (7.17) replaced with the outputs of the mixer such that a quasi-linear model is obtained.

$$\dot{x}_1 = c_0 u'_1 \tag{7.20}$$

$$\dot{x}_2 = c_1 u'_2 \tag{7.21}$$

$$\dot{x}_3 = c_2 u'_3 \tag{7.22}$$

$$\dot{x}_4 = x_1 \tag{7.23}$$

$$\dot{x}_5 = x_2 \tag{7.24}$$

$$\dot{x}_6 = x_3 \tag{7.25}$$

Finally using this mixer block, Figure 7.12 becomes Figure 7.32.

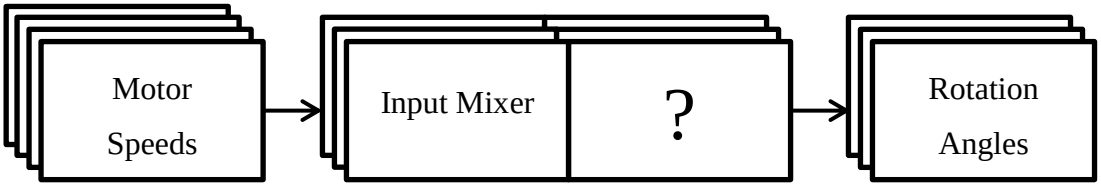


Figure 7.32 : Block diagram for ARX model.

7.3.7.2 Identification method

Matlab's System Identification Toolbox (Figure 7.33) provides many methods and tools for identification of ARX, ARMAX and OE etc. models, based on input-output data collected from the system. It is possible to use iddata objects that are prepared before with identification toolbox as identification data.

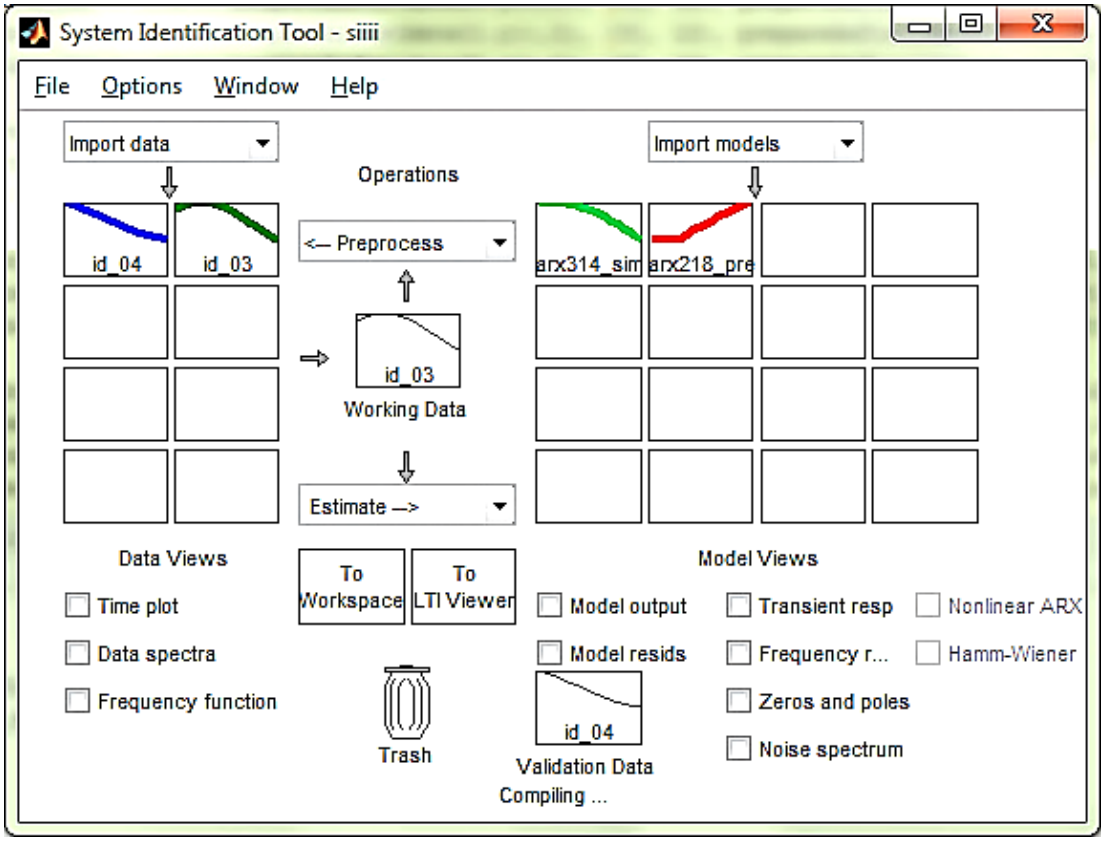


Figure 7.33 : System Identification Toolbox GUI

An iterative Matlab function to find the best ARX configuration is implemented. The function tries several ARX models via the toolbox function *arx* with all reasonable na, nb and nk parameters. It also simulates and compares the models with identification and verification datasets to calculate the root mean square errors. The errors are recorded to a table. The table is sorted to find the best configuration.

The ARX identification process is done in a similar way that of grey-box identification with two basic differences.

- The identified ARX models are discrete-time models.
- As an addition to grey-box, ARX models may have input delay.

A priori-knowledge about the model is its order and causality. It is known that the nonlinear model is of second order (for each axis). As a second thought, the model is not related to the past inputs.

From this starting point, it can be said that the order parameter (na) might be 2 and the input dependency parameter (nb) might be 1. Finally, the input delay parameter (nk) is guessed to be not zero but smaller than 10.

7.3.7.3 Identification results (roll)

Model orders is given in parenthesis such that ARX (na,nb,nk). Some of the model configurations for the roll axis are given below (Figure 7.34 - Figure 7.37).

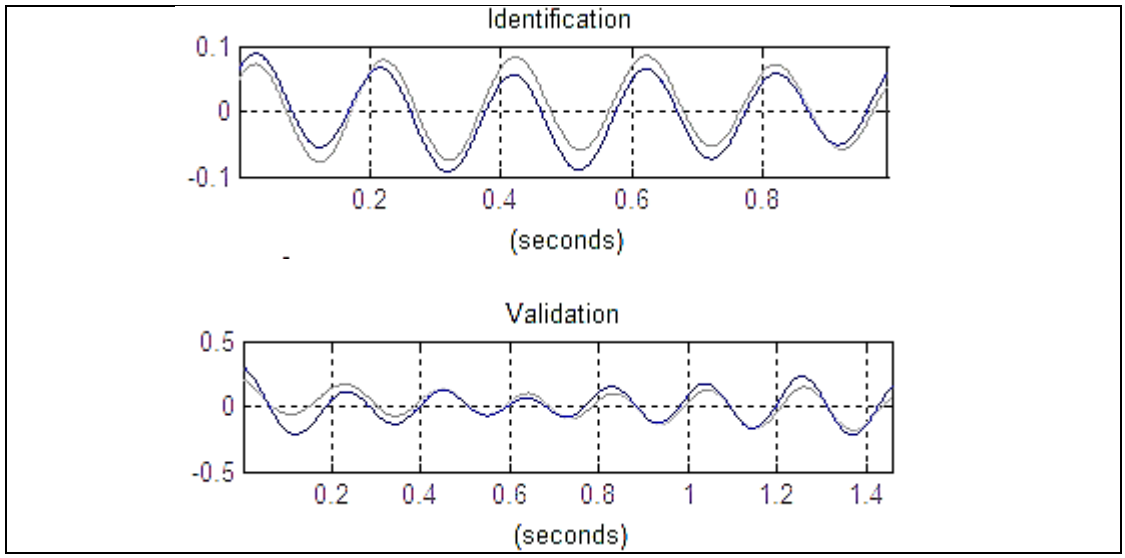


Figure 7.34 : ARX (2,1,0) roll model.

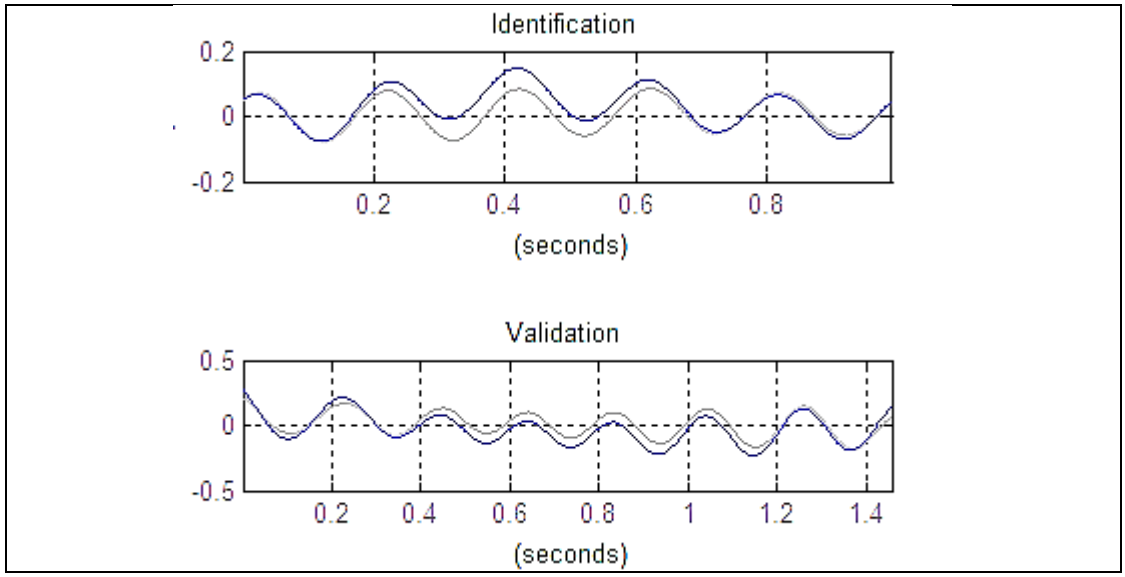


Figure 7.35 : ARX (2,1,4) roll model.

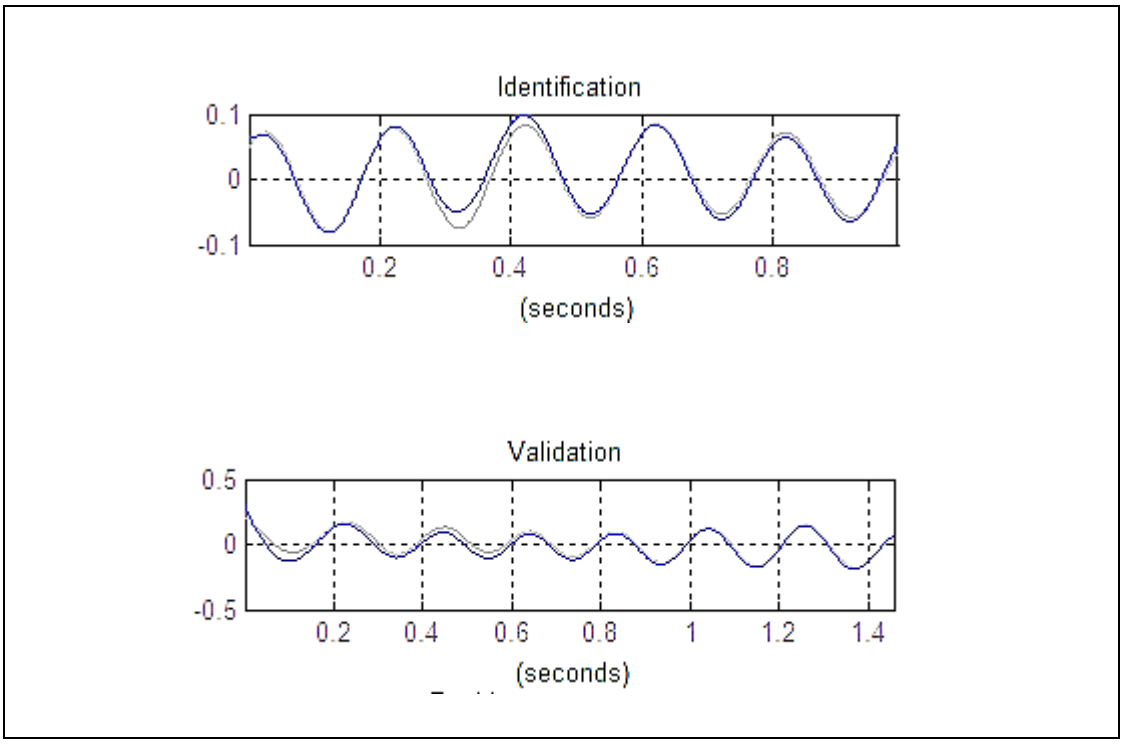


Figure 7.36 : ARX (2,1,8) roll model.

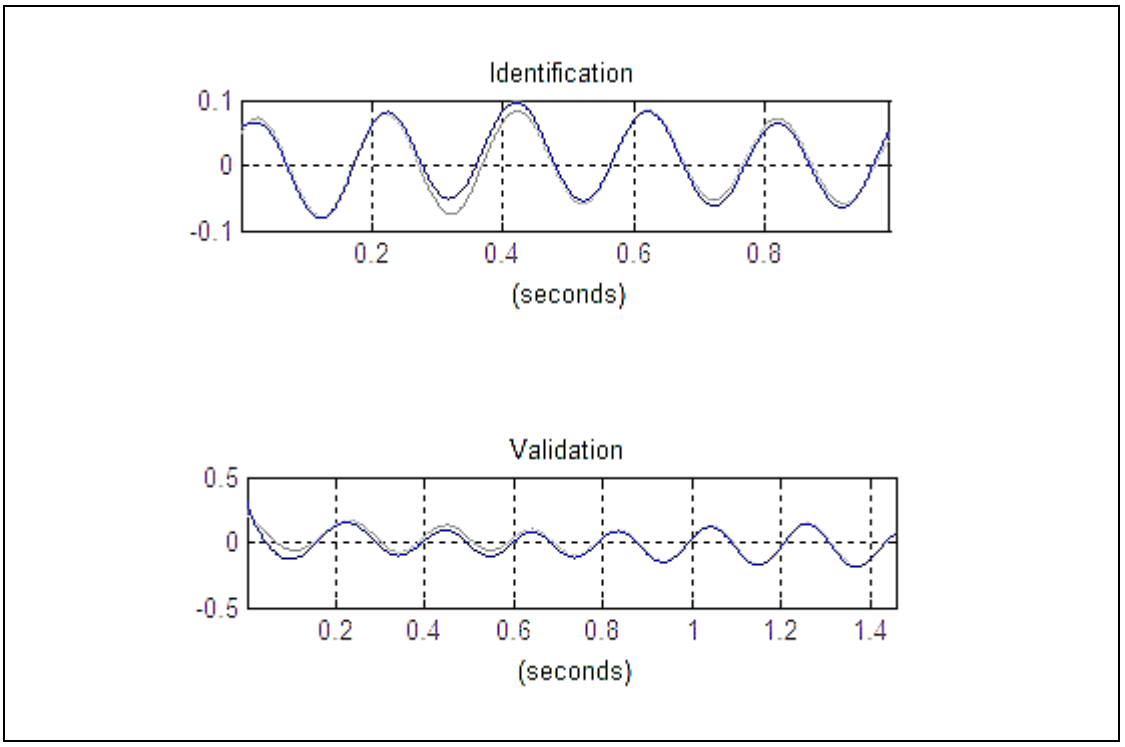


Figure 7.37 : ARX (2,1,10) roll model.

Although the model is assumed second order, it will be reasonable to try third and fourth order models because of the previously neglected dynamics. Some third and fourth order ARX modeling results are given below (Figure 7.38 - Figure 7.40).

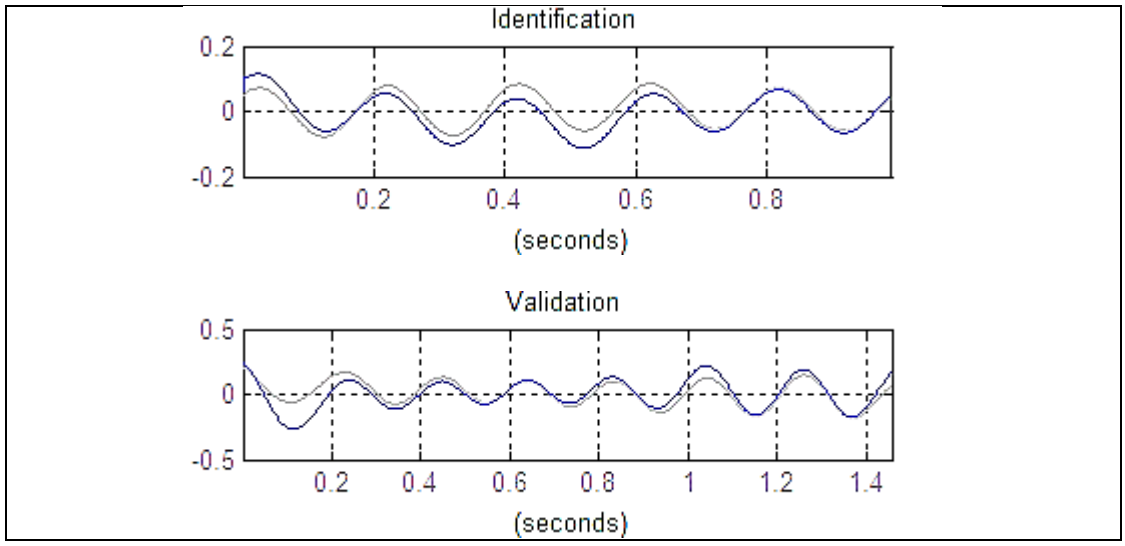


Figure 7.38 : ARX (3,1,0) roll model.

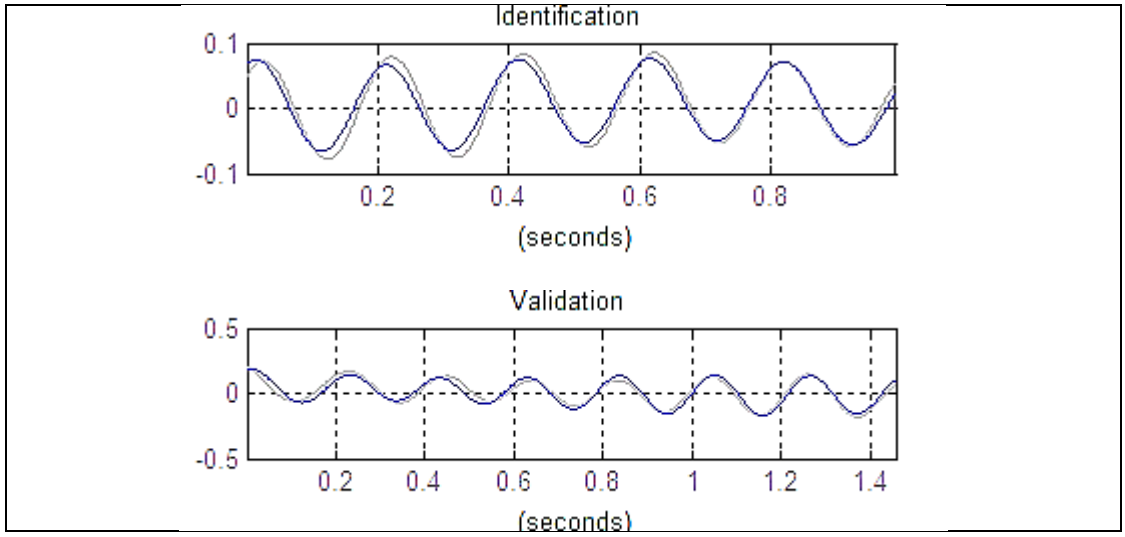


Figure 7.39 : ARX (3,1,4) roll model.

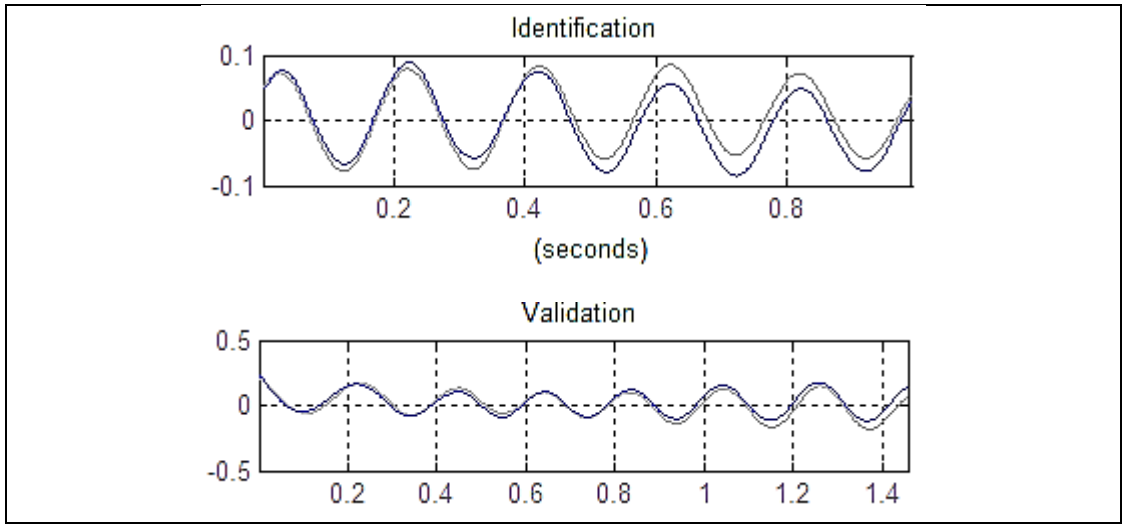


Figure 7.40 : ARX (4,2,0) roll model.

Additionally, just to be sure about the input dependency parameter (nb), $na = 2$ configurations are tried. Some of them are given below (Figure 7.41 - Figure 7.44).

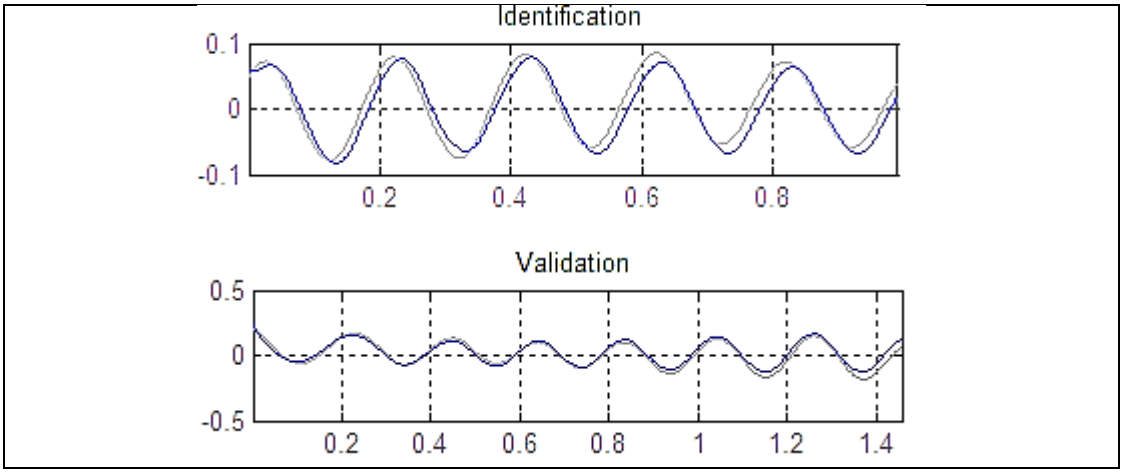


Figure 7.41 : ARX (2,2,0) roll model.

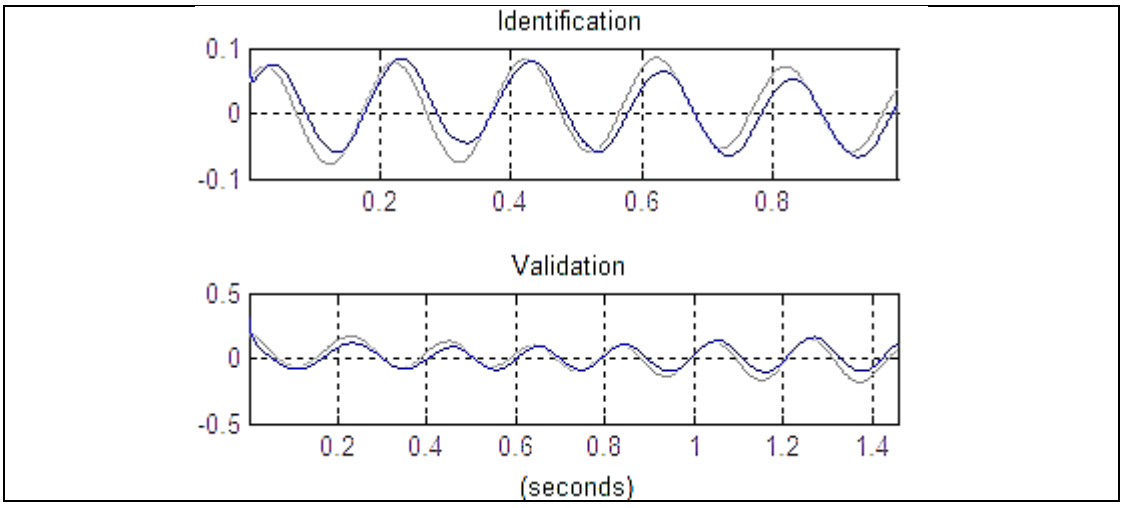


Figure 7.42 : ARX (2,2,4) roll model.

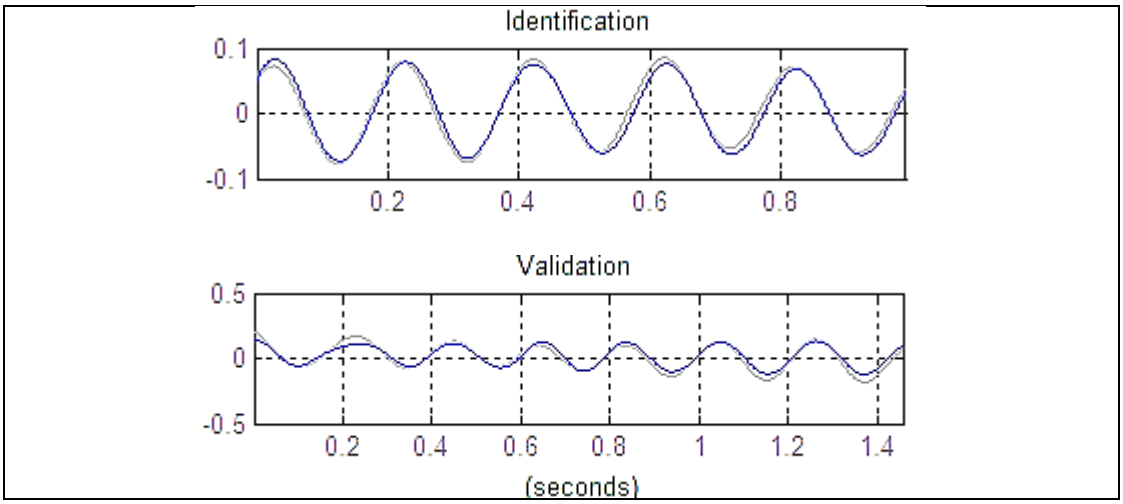


Figure 7.43 : ARX (3,2,0) roll model.

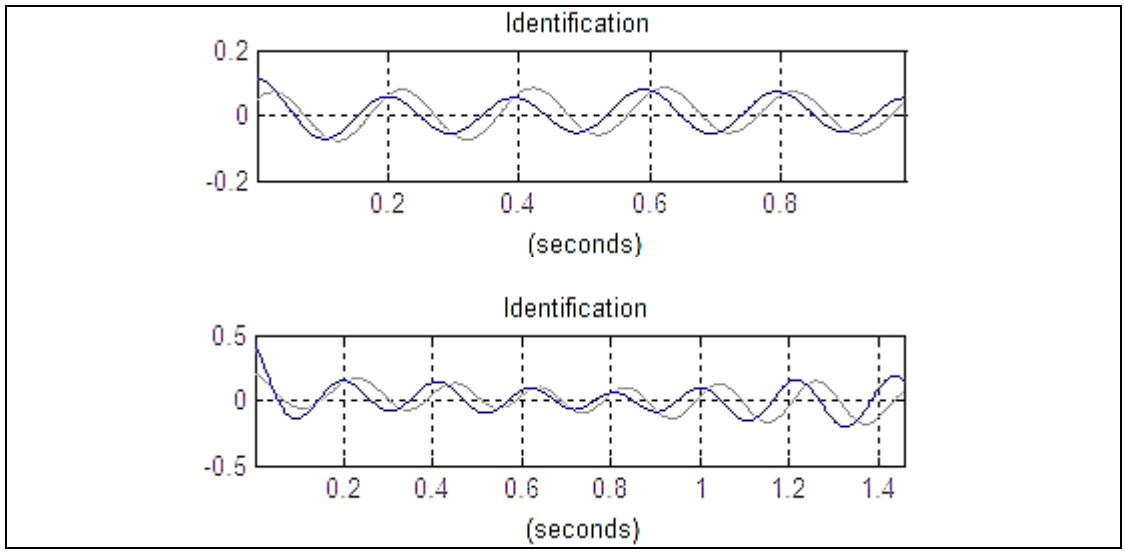


Figure 7.44 : ARX (3,2,4) roll model.

Model performances is comparison is done in respect to root mean square error values of verification data given in the table below (Table 7.3).

Table 7.3 : Top results of the iterative ARX identification (roll).

Model Configuration	RMS Error Identification	RMS Error Validation
ARX (2,2,0)	0,0134	0,0277
ARX (2,1,8)	0,0103	0,0296
ARX (2,1,9)	0,0099	0,0301
ARX (2,1,7)	0,0127	0,0304
ARX (2,1,6)	0,0140	0,0309
ARX (3,1,4)	0,0184	0,0332
ARX (4,2,0)	0,0192	0,0340
ARX (2,2,9)	0,0122	0,0345
ARX (2,2,6)	0,0157	0,0367
ARX (3,1,9)	0,0205	0,0388
ARX (3,2,9)	0,0207	0,0392
ARX (2,1,5)	0,0232	0,0396
ARX (2,2,8)	0,0163	0,0413
ARX (3,1,8)	0,0238	0,0420
ARX (4,2,9)	0,0256	0,0446
ARX (4,1,9)	0,0264	0,0460
ARX (2,2,2)	0,0223	0,0463
ARX (2,2,7)	0,0199	0,0486
ARX (4,1,8)	0,0316	0,0523
ARX (2,1,4)	0,0355	0,0550

It is clearly seen that ARX (2,2,0), ARX (2,1,8) and ARX (2,1,9) models gave better results with verification data.

7.3.7.4 Model verification (roll)

In order to verify the identified ARX models, positive and negative step responses should be observed. The input and output are expected to have same signs.

The model that have best validation performance is ARX (2,2,0). But the amplitude of its step response is very small and cannot be true (Figure 7.45).

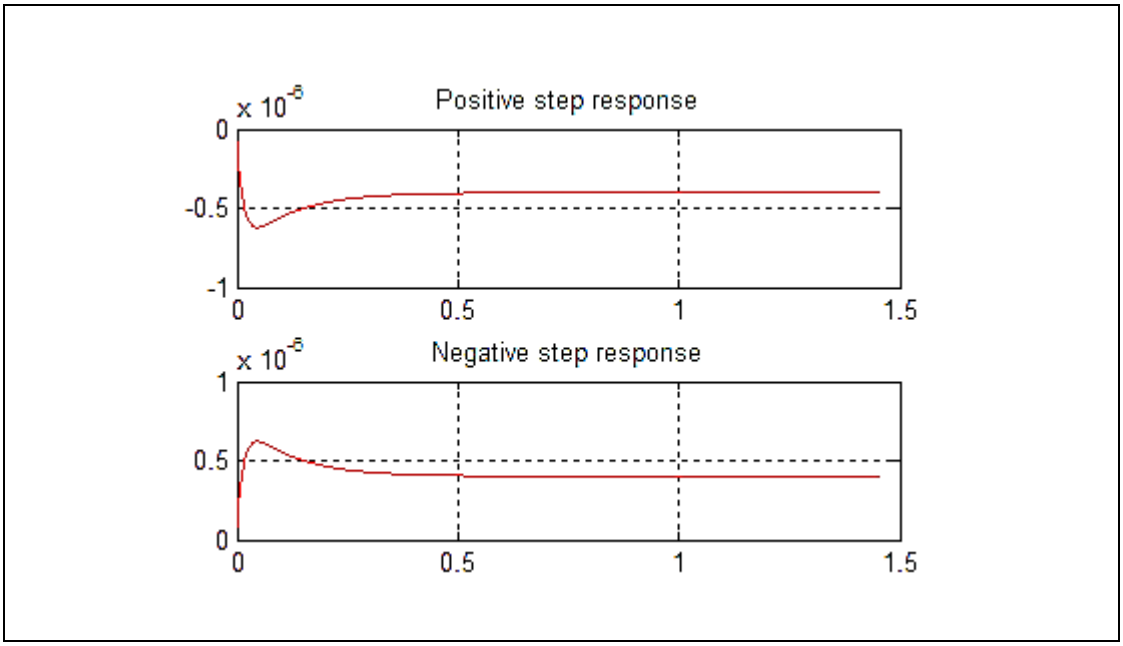


Figure 7.45 : ARX (2,2,0) roll model step responses.

Another model is ARX(2,1,8) and its step responses are given below (Figure 7.46).

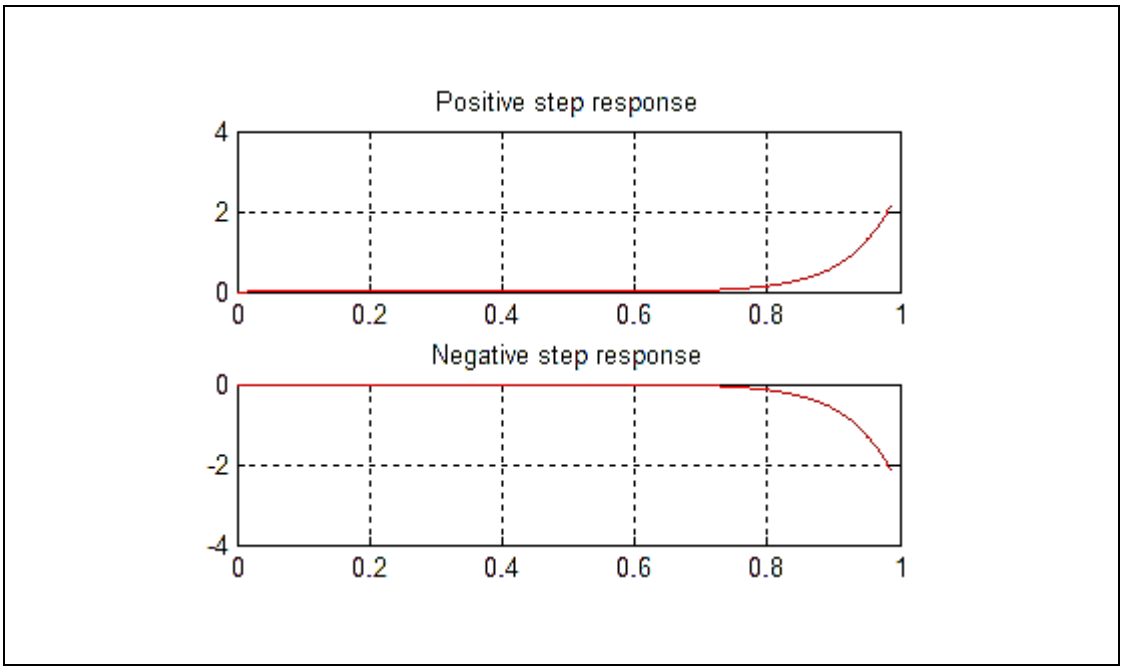


Figure 7.46 : ARX (2,1,8) roll model step responses.

It can be clearly seen that the positive step response makes the output of ARX (2,1,8) positive while the negative step response makes it negative. The amplitudes are growing exponentially as expected.

In addition, a third order model, ARX (3,1,4) and a fourth order model, ARX (4,2,0) are simulated with step inputs and the responses are given (Figure 7.47, Figure 7.48).

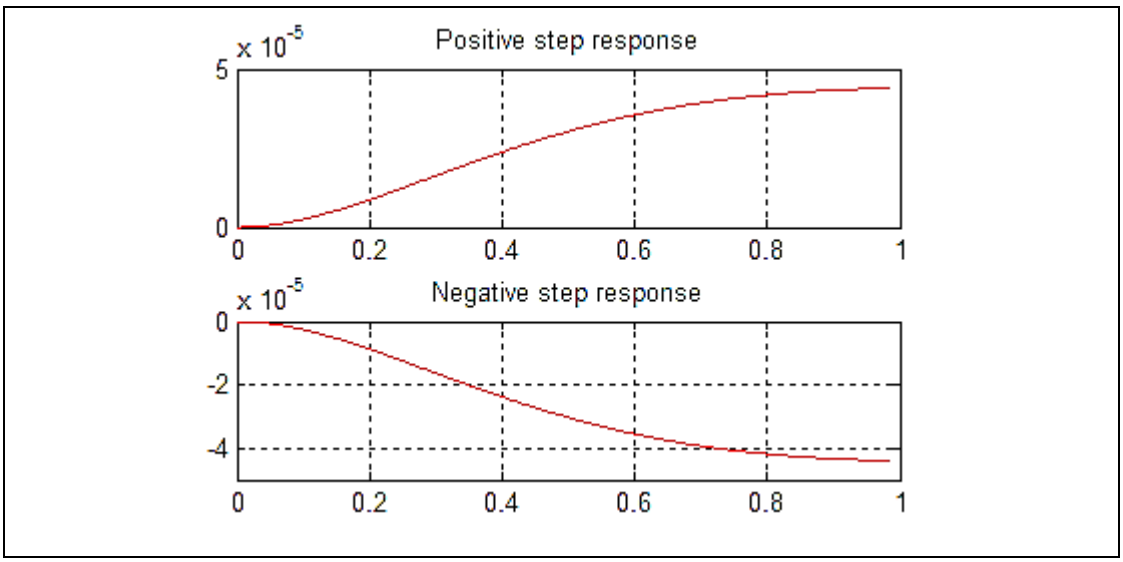


Figure 7.47 : ARX (3,1,4) roll model step responses.

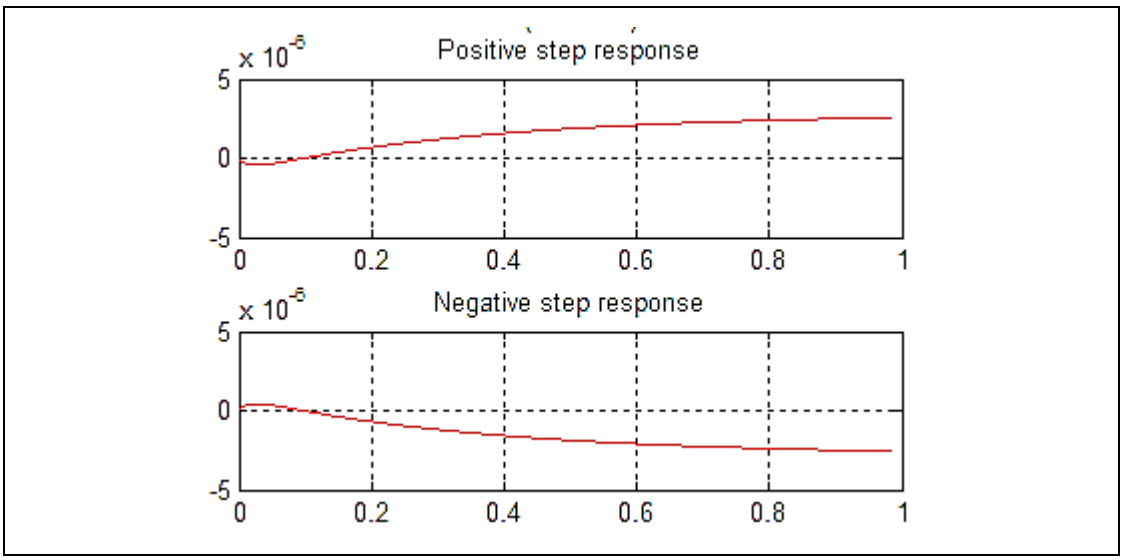


Figure 7.48 : ARX (4,2,0) roll model step responses.

Responses of the third and fourth order models are not the responses that are expected when a continuous torque is applied to a free body.

Only the step responses of ARX (2,1,8) model are found reasonable while the other models failed.

7.3.7.5 Cross-correlation test (roll)

Another test that can give information about the goodness of the model is cross-correlation test.

The tested model is divided into residues with respect to a given lag value. Then the cross-correlation between the model residues and the input are calculated. The 99% confidence intervals are also calculated and compared to these values. The values are expected to lie in this confidence region [42].

The test results for the models (with 25 lag) are given below (Figure 7.49 - Figure 7.52).

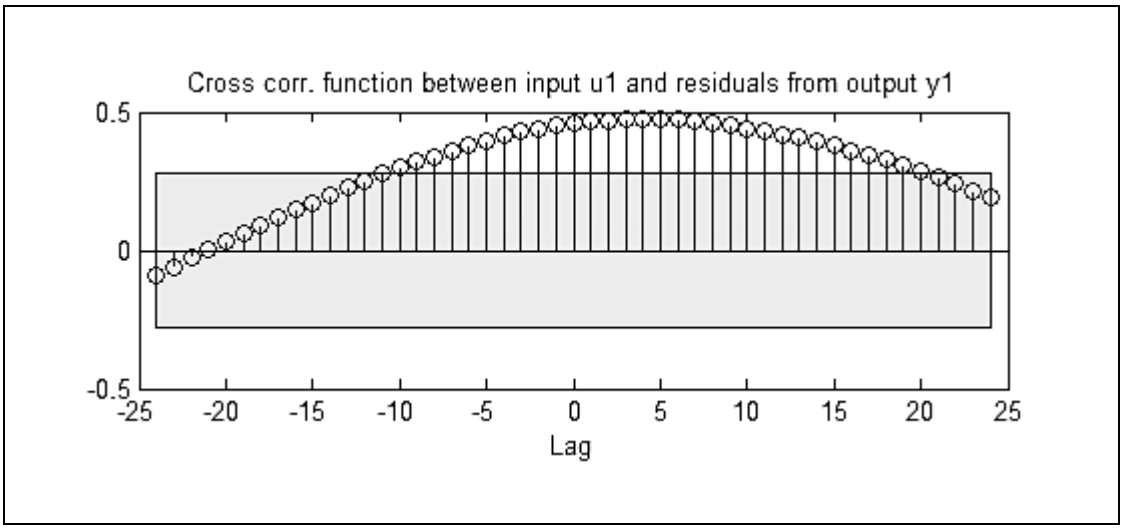


Figure 7.49 : ARX (2,2,0) roll model cross-correlation test.

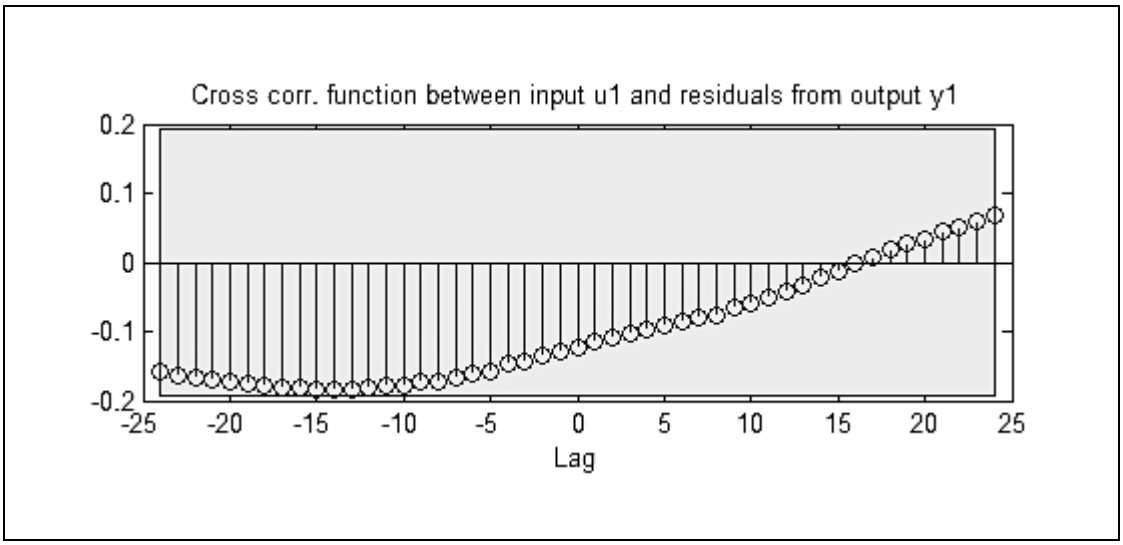


Figure 7.50 : ARX (2,1,8) roll model cross-correlation test.

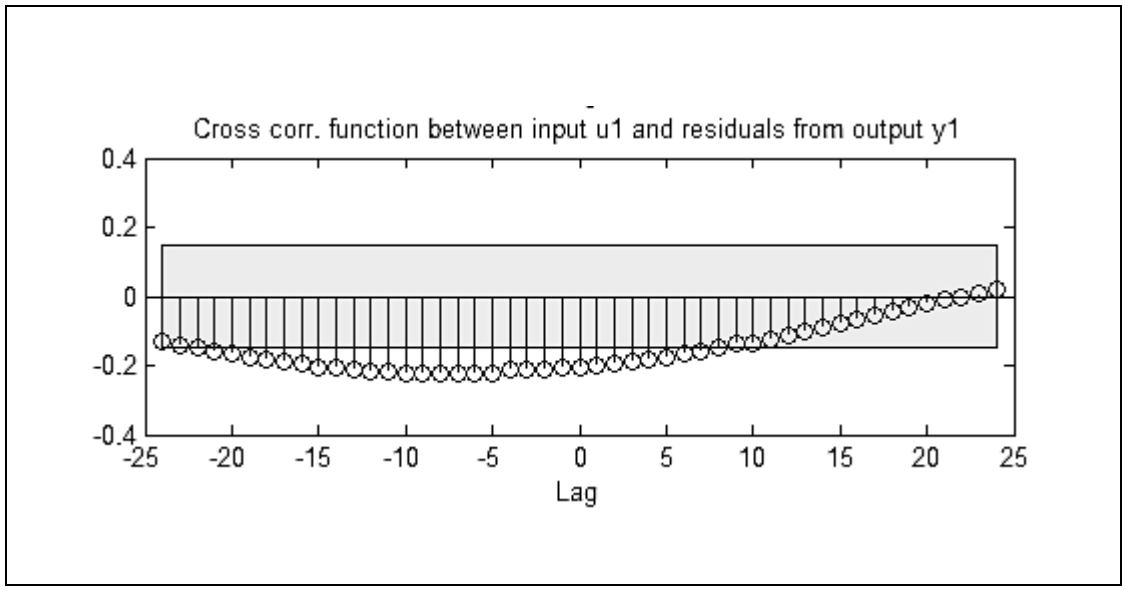


Figure 7.51 : ARX (3,1,4) roll model cross-correlation test.

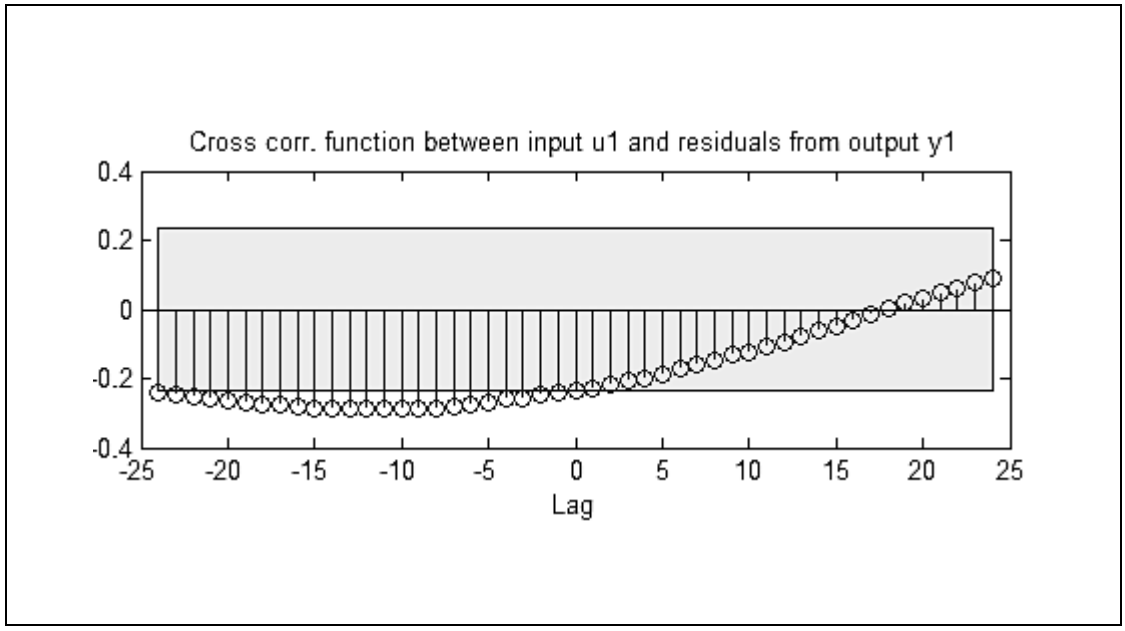


Figure 7.52 : ARX (4,2,0) roll model cross-correlation test.

Cross-correlation test supports the results of the step-response test, which is previously done in chapter 7.3.7.4.

It is obvious from Figure 7.49 that the input and the residuals of ARX (2,2,0) are correlated because of being outside of the confidence region (shown as a box). The same sentence is also true for ARX (3,1,4) and ARX (4,2,0) models.

ARX (2,1,8) passes the cross-correlation test as it lies inside the confidence region. It can be said that ARX (2,1,8) is a very good model to continue with.

The discrete-time ARX roll model is explicitly given below.

$$A_{ROLL}(z)\phi(t) = B_{ROLL}(z)u_1'(t) + e(t) \quad \textbf{(7.26)}$$

$$A_{ROLL}(z) = 1 - 1.959\,z^{-1} + 0.9567\,z^{-2}$$

$$B_{ROLL}(z) = 4.222 \times 10^{-8}\,z^{-8}$$

Finally, the discrete-time transfer function for the roll axis is given below.

$$G_{ROLL}(z) = \frac{4.222 \times 10^{-8}\,z^{-8}}{1 - 1.959\,z^{-1} + 0.9567\,z^{-2}} \quad \textbf{(7.27)}$$

7.3.7.6 Identification results (pitch)

The pitch model is assumed identical to roll as described before, but verification is required. Performance of the ARX (2,1,8) model when applied to pitch axis data is given below (Figure 7.53).

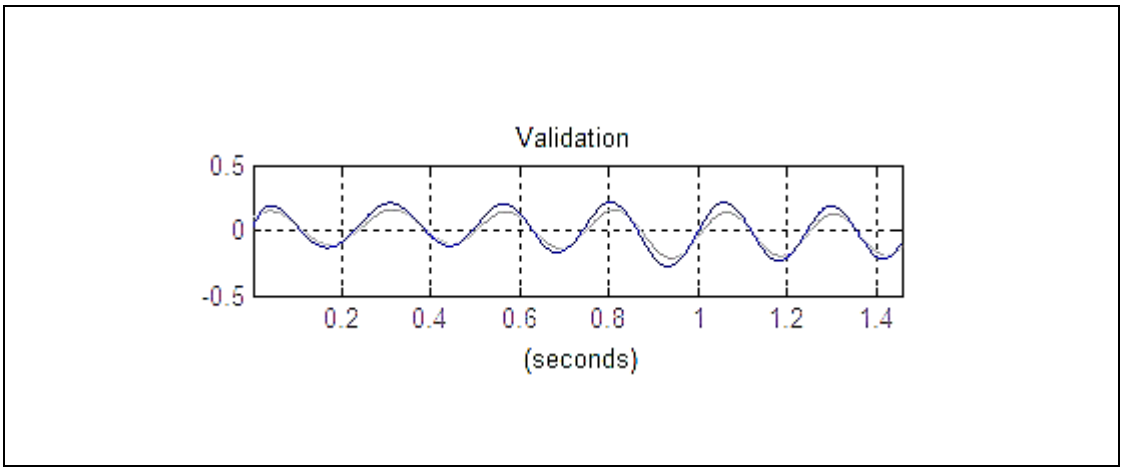


Figure 7.53 : Roll model with pitch axis data.

The result is quite satisfactory, given in the table below (Table 7.4).

Table 7.4 : Performance of the selected ARX model (roll and pitch).

Data Axis	RMS Error Identification	RMS Error Validation
Roll	0.0103	0.0296
Pitch	-	0.0411

The discrete-time ARX pitch model is explicitly given below.

$$A_{PITCH}(z)\theta(t) = B_{PITCH}(z)u_2'(t) + e(t) \quad \textbf{(7.28)}$$

$$A_{PITCH}(z) = 1 - 1.959\,z^{-1} + 0.9567\,z^{-2}$$

$$B_{PITCH}(z) = 4.222 \times 10^{-8}\,z^{-8}$$

Finally, the discrete-time transfer function for the pitch axis is given below.

$$G_{PITCH}(z) = \frac{4.222 \times 10^{-8}\,z^{-8}}{1 - 1.959\,z^{-1} + 0.9567\,z^{-2}} \quad \textbf{(7.29)}$$

7.3.7.7 Identification results (yaw)

The same process is applied to the yaw axis. The model performances of various models are given below (Table 7.5).

Table 7.5 : Top results of ARX identification (yaw).

Model Configuration	RMS Error Identification	RMS Error Validation
ARX (2,1,8)	0,0129	0,0140
ARX (3,2,8)	0,0124	0,0172
ARX (2,1,7)	0,0142	0,0277
ARX (2,2,8)	0,0096	0,0419
ARX (2,2,7)	0,0115	0,0461
ARX (3,2,7)	0,0113	0,0573
ARX (2,1,9)	0,0091	0,0694
ARX (2,2,6)	0,0107	0,0746
ARX (3,1,9)	0,0089	0,0795
ARX (4,2,9)	0,0103	0,0849
ARX (3,2,9)	0,0101	0,0896
ARX (2,2,9)	0,0086	0,0957
ARX (2,2,5)	0,0094	0,1056
ARX (3,1,8)	0,0085	0,1077
ARX (3,2,6)	0,0096	0,1149
ARX (2,1,6)	0,0083	0,1188
ARX (2,2,4)	0,0090	0,1229
ARX (4,2,8)	0,0095	0,1251
ARX (3,1,7)	0,0081	0,1309
ARX (2,1,5)	0,0081	0,1342

Performance of ARX (2,1,8) is the best fit for validation data (Figure 7.54). This result also supports that the system has a delay, which is also appeared in the roll axis identification results given in section 7.3.7.3.

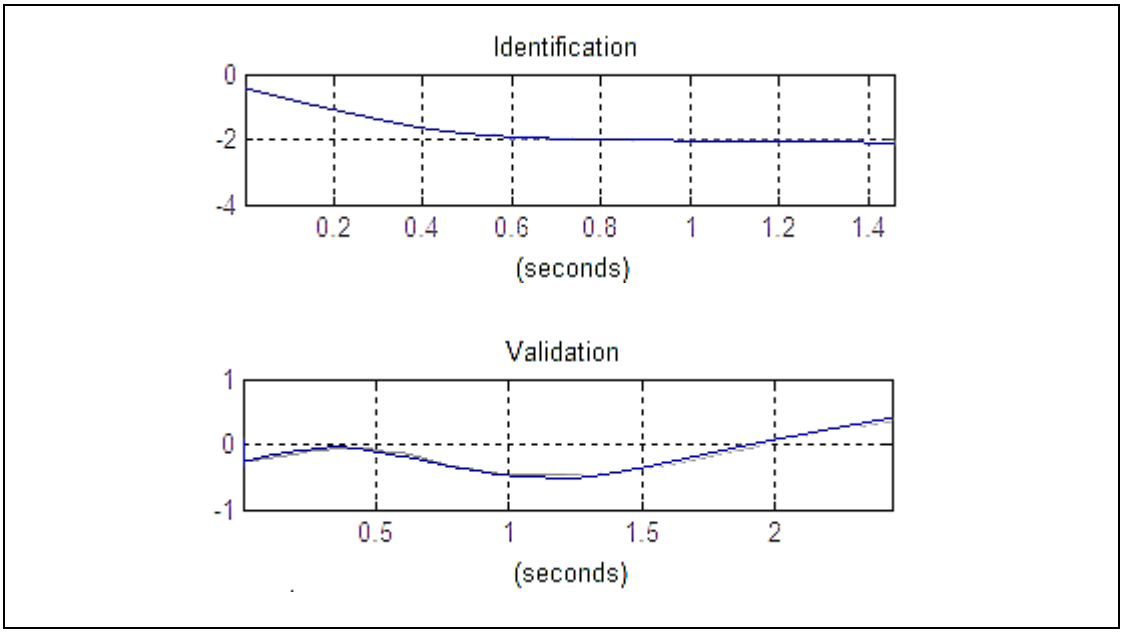


Figure 7.54 : ARX (2,1,8) yaw model.

7.3.7.8 Model verification (yaw)

To test the identified model, one positive and one negative step inputs applied. The result (Figure 7.55) has the same sign as the input and no oscillations occurred.

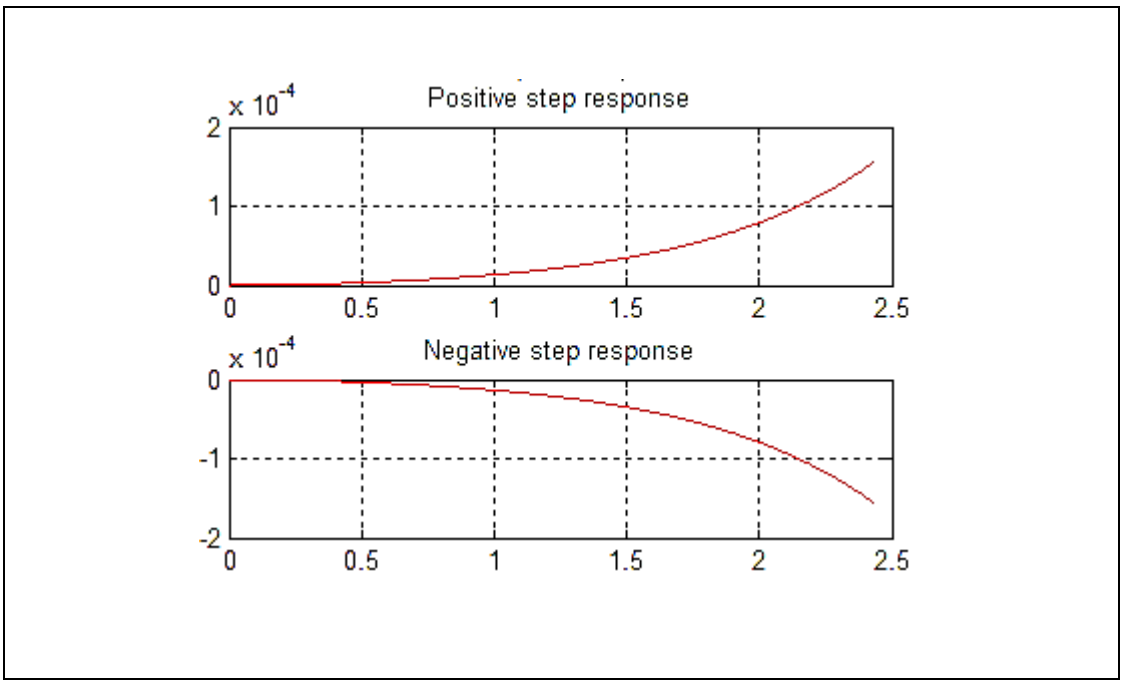


Figure 7.55 : ARX (2,1,8) yaw model step responses.

7.3.7.9 Cross-correlation test (yaw)

Cross-correlation test is applied to ARX (2,1,8) yaw axis model and the result is given below (Figure 7.56).

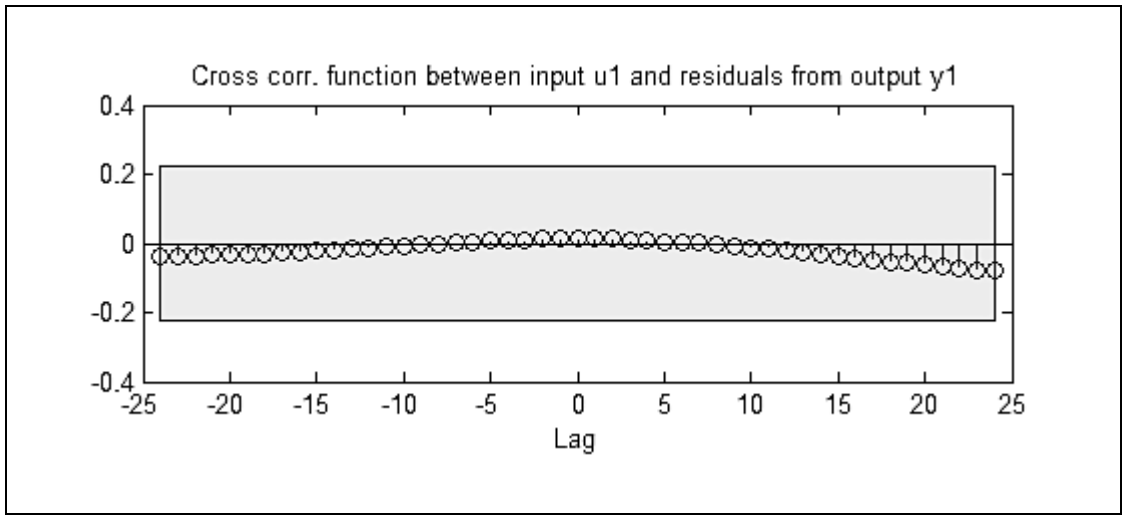


Figure 7.56 : ARX (2,1,8) yaw model cross-correlation test.

All of the cross-correlation values lies inside the 99% confidence region. This indicates that the model works well. This model can be used for simulation of the yaw axis of the quadrotor.

The discrete-time ARX model for the yaw axis is explicitly given below.

$$A_{YAW}(z)\psi(t) = B_{YAW}(z)u_3'(t) + e(t) \quad \textbf{(7.30)}$$

$$A_{YAW}(z) = 1 - 1.995\,z^{-1} + 0.9949\,z^{-2}$$

$$B_{YAW}(z) = 1.382 \times 10^{-9}\,z^{-8}$$

Finally, the discrete-time transfer function for the yaw axis is given below.

$$G_{YAW}(z) = \frac{1.382 \times 10^{-9}\,z^{-8}}{1 - 1.995\,z^{-1} + 0.9949\,z^{-2}} \quad \textbf{(7.31)}$$

7.3.8 Model decision

It is a truth that the ARX approach gave better results for all of the axes. The underlying reason for that is the input delay, which is persistently appeared in all ARX models.

The linear ARX models are found in a quasi-linear form that is previously shown in Figure 7.32, thanks to the input mixer block. Now including the input mixer, the complete ARX based nonlinear models are given below.

$$\phi(z) = \frac{4.222 \times 10^{-8}}{1 \; - \; 1.959\, z^{-1} \; + \; 0.9567\, z^{-2}} (\omega_{M4}^2 - \omega_{M2}^2) z^{-8} \tag{7.32}$$

$$\theta(z) = \frac{4.222 \times 10^{-8}}{1 \; - \; 1.959\, z^{-1} \; + \; 0.9567\, z^{-2}} (\omega_{M3}^2 - \omega_{M1}^2) z^{-8} \tag{7.33}$$

$$\psi(z) = \frac{1.382 \times 10^{-9}}{1 \; - \; 1.995\, z^{-1} \; + \; 0.9949\, z^{-2}} (-\omega_{M1}^2 + \omega_{M2}^2 - \omega_{M3}^2 + \omega_{M4}^2) z^{-8} \tag{7.34}$$

8. QUADROTOR SIMULATION

The goal of this work is simulation the dynamics of a quadrotor platform, Crazyflie, on a computer. The advantage of being able to simulate the real platform is the ability to design control systems safely.

In this chapter, a simulation scheme for Crazyflie quadrotor will be presented.

8.1 Simulation Overview

The most basic blocks of the quadrotor simulation diagram are shown (Figure 8.1).

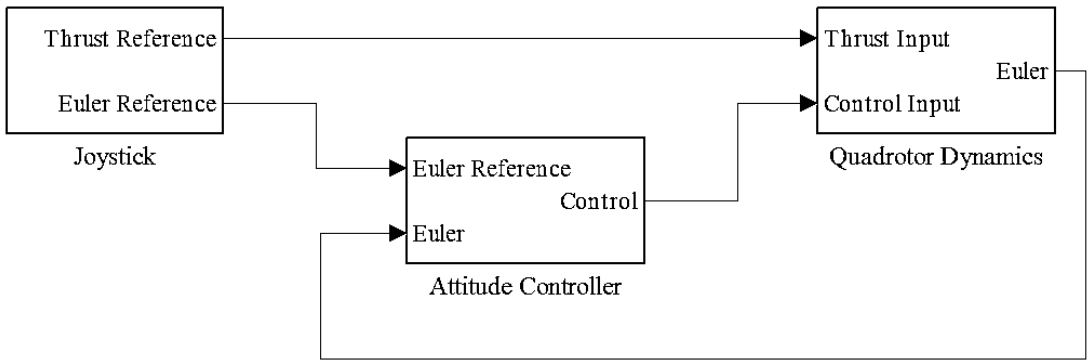


Figure 8.1 : Simulation diagram overview.

User (or pilot) generates thrust and Euler angle references via the joystick controller.

Euler angle reference vector is compared to the current Euler angles of the quadrotor to calculate an error vector.

An attitude controller minimizes the attitude errors generating control input to the quadrotor.

As there is no onboard sensor for altitude feedback, thrust reference is directly applied to the plant.

8.2 Simulation Blocks

Now the simulation blocks presented in section 8.1 will be described in details.

8.2.1 Joystick block

Joystick block is simulated using constants. The detailed scheme is given below (Figure 8.2).

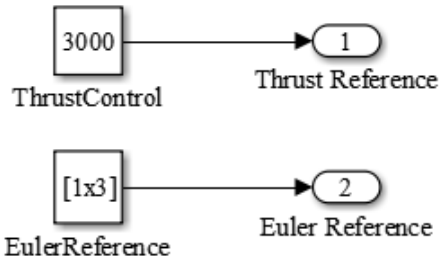


Figure 8.2 : The joystick block.

8.2.2 Attitude controller block

The inner structure of the attitude controller block is shown below (Figure 8.3).

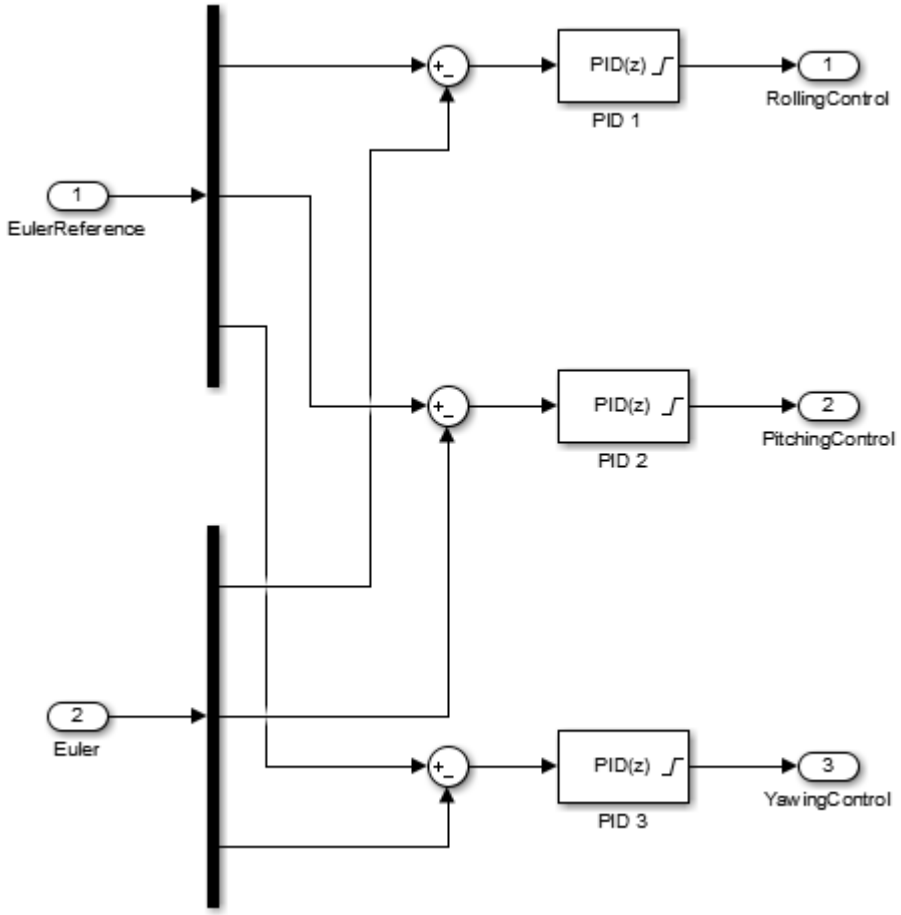


Figure 8.3 : Attitude controller block.

The attitude controller includes proportional-integral-derivative (PID) controllers just to prove that the simulation is up and working.

The outputs of the PID blocks are saturated to the maximum and the minimum values of the PWM signals; in this case $\pm$4096.

The inner structure of one of the PID blocks is shown below.

$$K_P + K_I \frac{T_S}{z^{-1}} + K_D \frac{z^{-1}}{T_S} \tag{8.1}$$

Where K_P is the proportional gain, K_I is the integral gain, K_D is the derivative gain and T_S is the sampling time.

8.2.3 Quadrotor dynamics block

The most complex part of the simulation scheme is the quadrotor dynamics block, which is shown below (Figure 8.4).

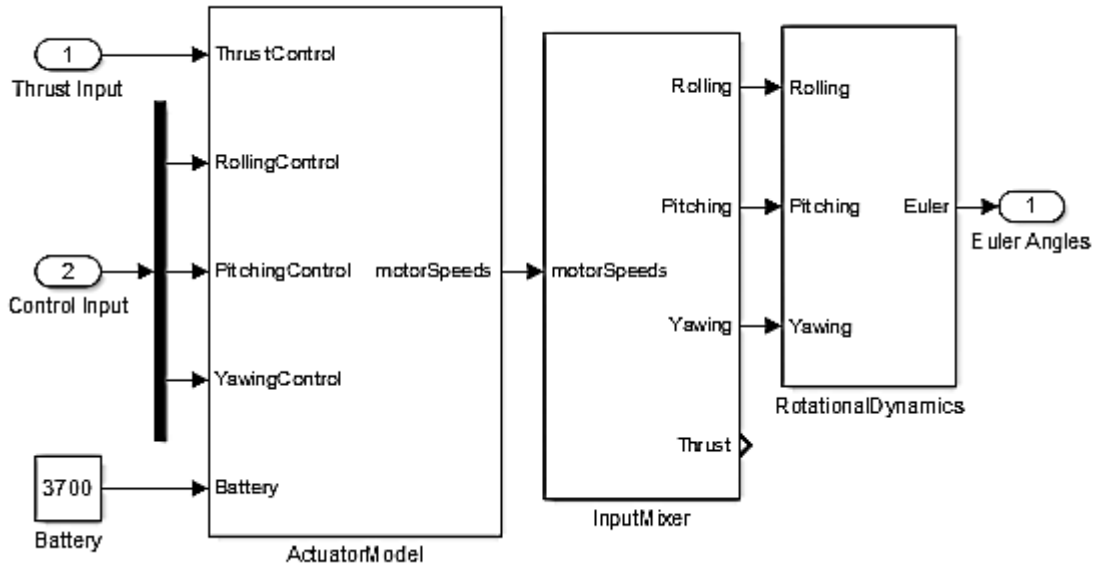


Figure 8.4 : Quadrotor dynamics block.

The control input signal that includes rolling, pitching and yawing control inputs distributed via a demux. Thrust control input comes directly from the joystick. An additional input to this block is battery voltage, which heavily affects the actuators.

The actuator model block outputs motor speeds vector. Input mixer block converts motor speeds vector to rolling, pitching and yawing moments.

Finally, the rotational dynamics block mimics the quadrotor body and outputs the attitude angels of the body.

8.2.4 Actuator model block

The figure below (Figure 8.5) shows the inside of the actuator model block.

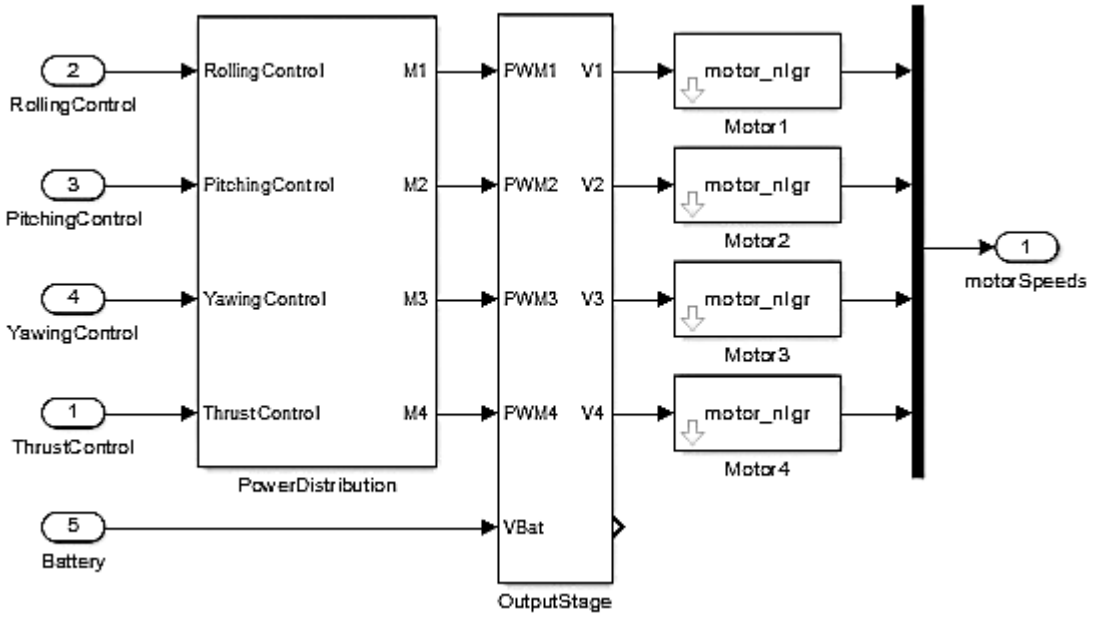


Figure 8.5 : Actuator model block.

The rolling, pitching and yawing control inputs have to be translated into the actuator input, the pulse width modulation (PWM) signals. The power distribution block is responsible for this.

As PWM is a signal related to the battery voltage, the output stage block converts PWM signals into motor armature voltages.

Finally, nonlinear motor models work to convert armature voltages into motor speeds. These models are identical, nonlinear grey-box models. The generated motor speeds fed into a mux and outputted as a vector.

8.2.5 Power distribution block

The power distribution block is responsible for generating PWM signals as a function of rolling, pitching and yawing controls as well as the thrust control input.

The representation of this block is given below.

$$PWM_{4x1} = f(thrust, rolling, pitching, yawing) \tag{8.2}$$

The explicit formulation is given below.

$$\begin{bmatrix} PWM_1 \\ PWM_2 \\ PWM_3 \\ PWM_4 \end{bmatrix} = \begin{bmatrix} thrust & - & pitching & - & yawing \\ thrust & - & rolling & + & yawing \\ thrust & + & pitching & - & yawing \\ thrust & + & rolling & + & yawing \end{bmatrix} \tag{8.3}$$

In addition to the distribution, each PWM signal is limited to a value. The upper limit that corresponds to 100% duty is 4096. The inside of the block is given below (Figure 8.6).

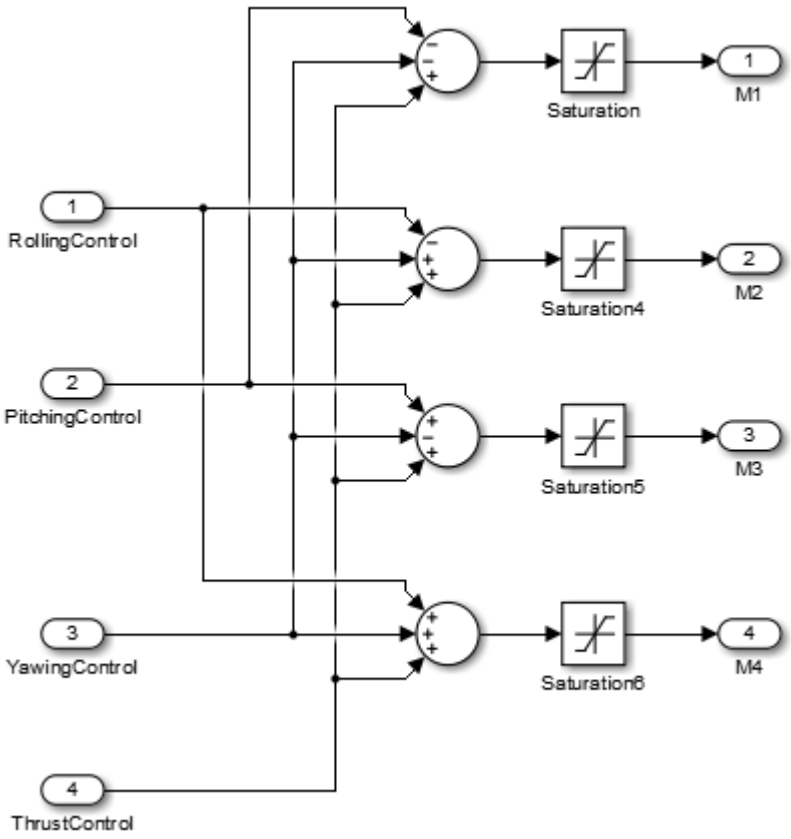


Figure 8.6 : Power distribution block.

8.2.6 Output stage block

This block converts PWM signals into motor armature voltages. First, the PWM signals are normalized using 1/4096 gain block. Then the battery voltage is scaled using these signals.

The inside of the block is given below (Figure 8.7).

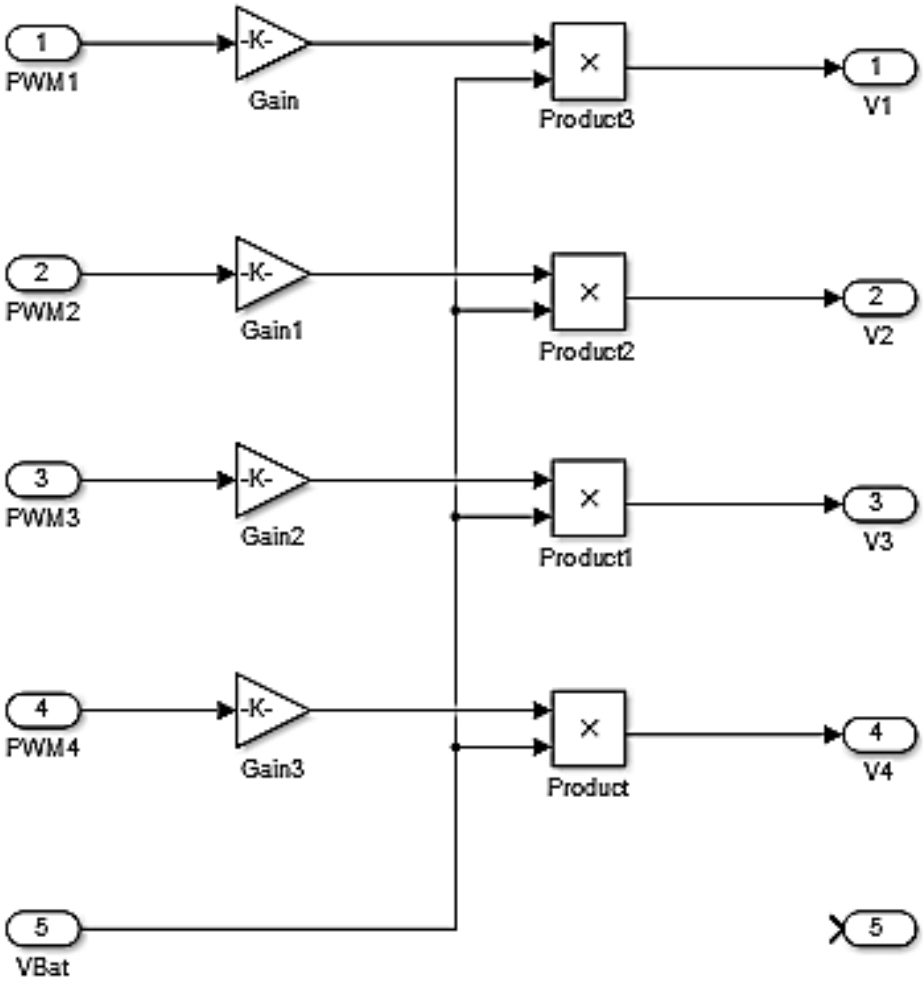


Figure 8.7 : Output stage block.

8.2.7 Input mixer block

The purpose of the input mixer block is to provide inputs to the quadrotor model with respect to the equation (7.19).

Inner view of the block is shown below (Figure 8.8).

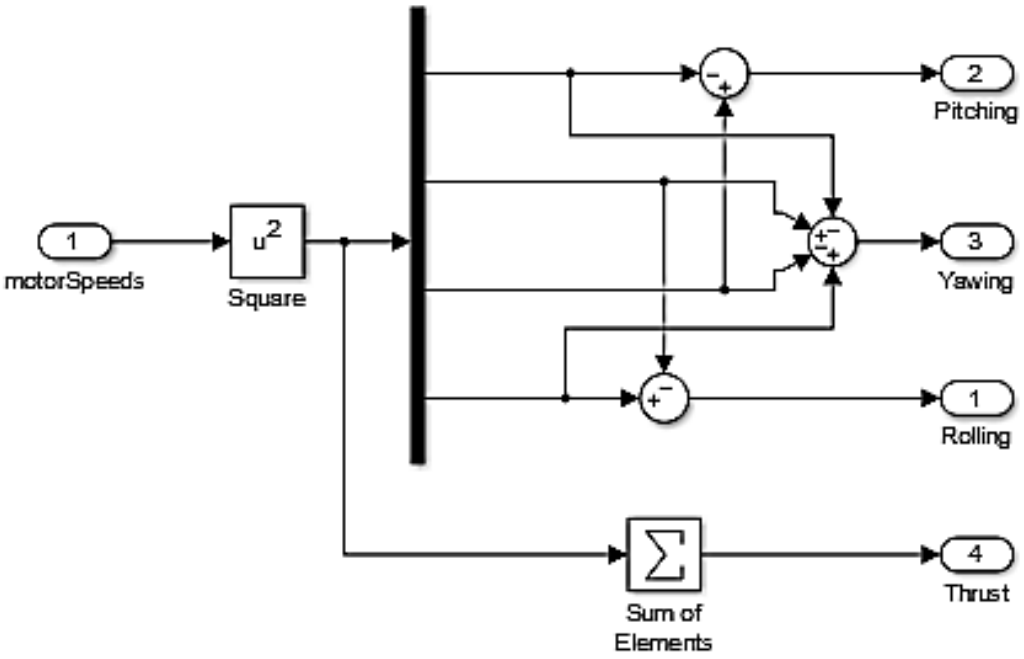


Figure 8.8 : Input mixer block.

8.2.8 Rotational dynamics block

Rotational dynamics block is responsible for translating rolling, pitching and yawing torques into Euler angles. The block diagram is given below (Figure 8.9).

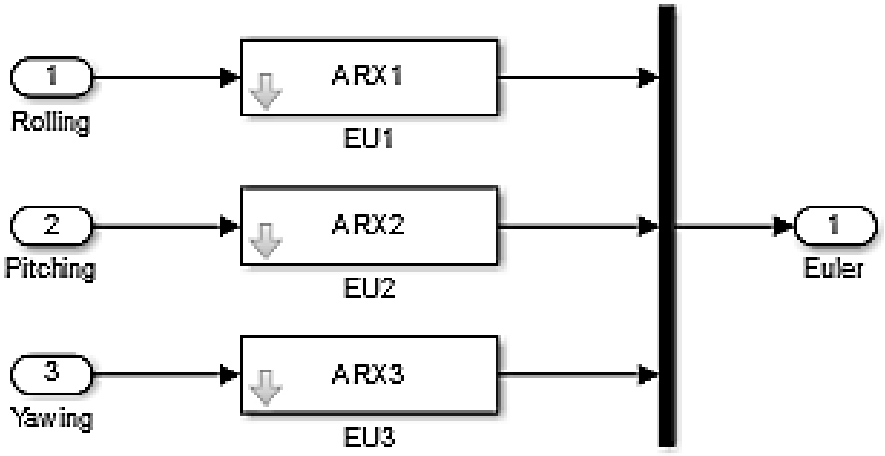


Figure 8.9 : Rotational dynamics block.

The whole simulation is given in Figure 8.10.

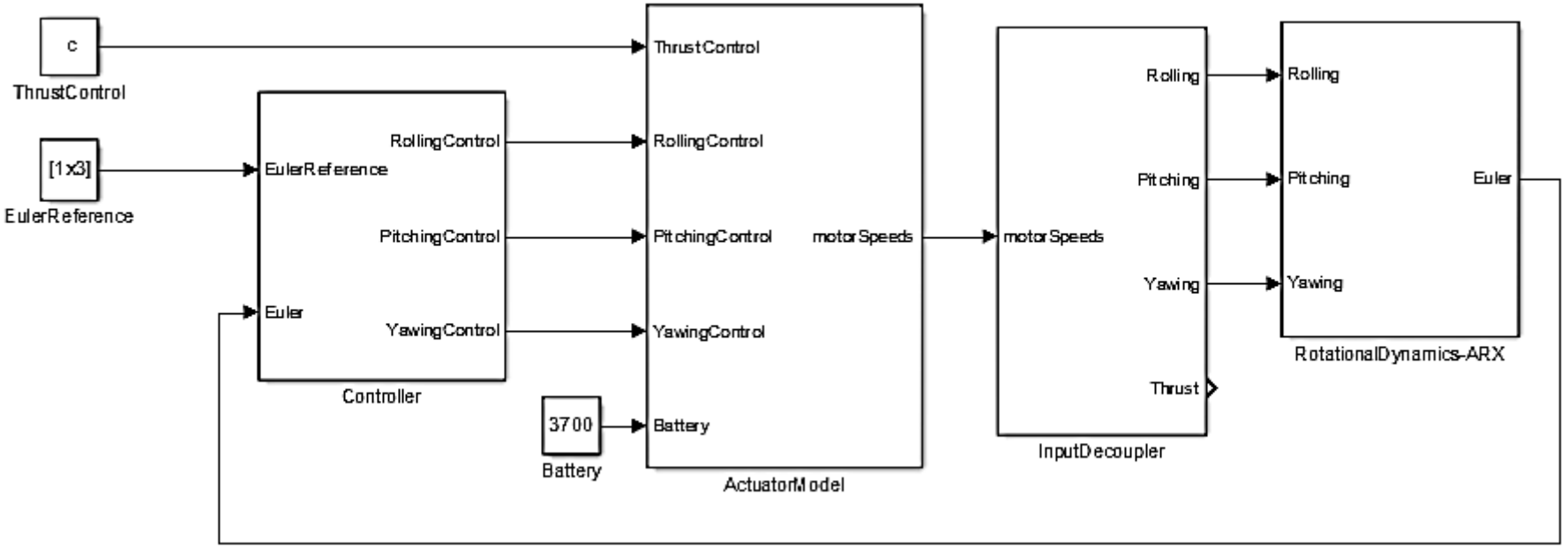


Figure 8.10 : Whole simulation diagram.

8.3 Control of the Simulation Model

In order to show the obtained quadrotor model is good, several proportional integral and derivative based controllers will be given. (Trial and error method is used.)

8.3.1 Proportional (P) controller

Structure of a P controller is given (Figure 8.11) where error e is given by $e = r - y$.

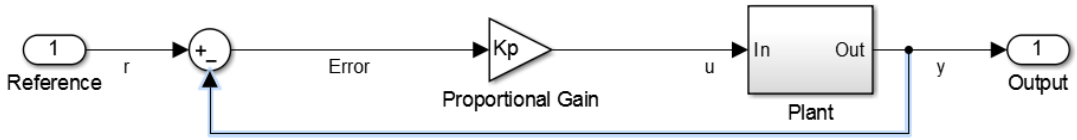


Figure 8.11 : Proportional controller.

$$u = K_P e \tag{8.4}$$

Step responses of the axes using P controller (Table 8.1) are given (Figure 8.12).

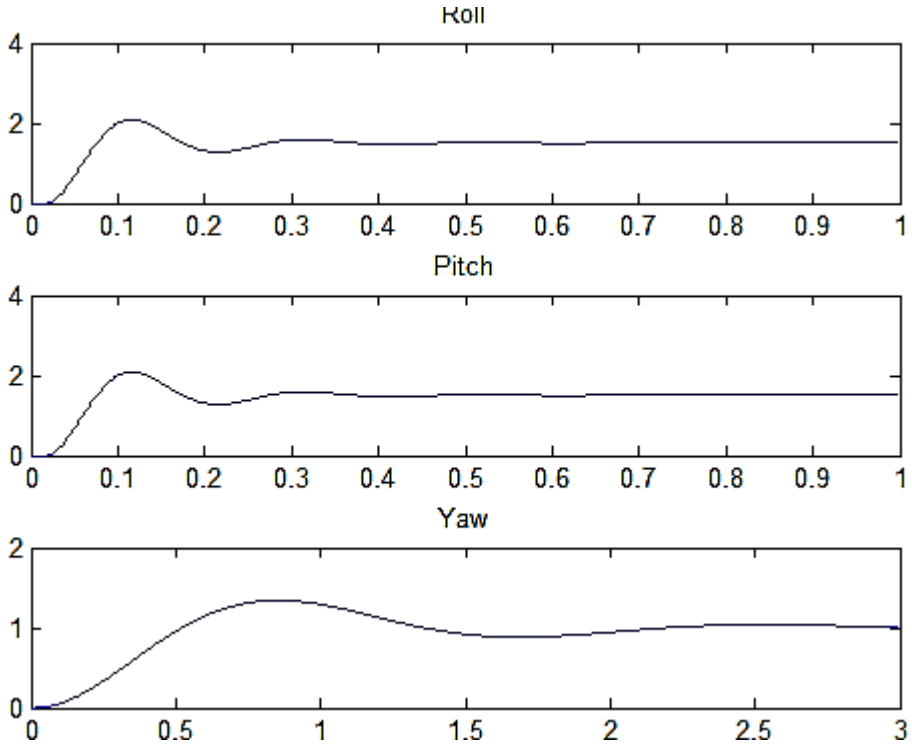


Figure 8.12 : P controller results.

Table 8.1 : P controller gains.

	Roll	Pitch	Yaw
K_P	149000	149000	44100

8.3.2 Proportional-integral (PI) controller

The block diagram of a PI controller is given below (Figure 8.13).

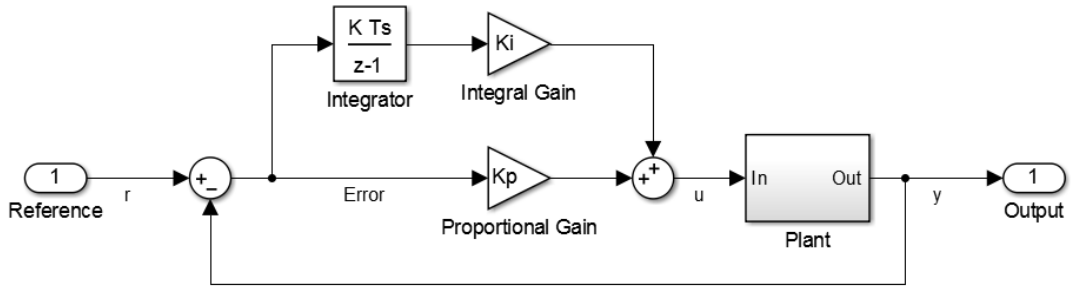


Figure 8.13 : Proportional-integral controller.

$$u = K_P e + K_I \int e\, dt \tag{8.5}$$

Step responses of the axes using PI controller (Table 8.2) are given (Figure 8.14).

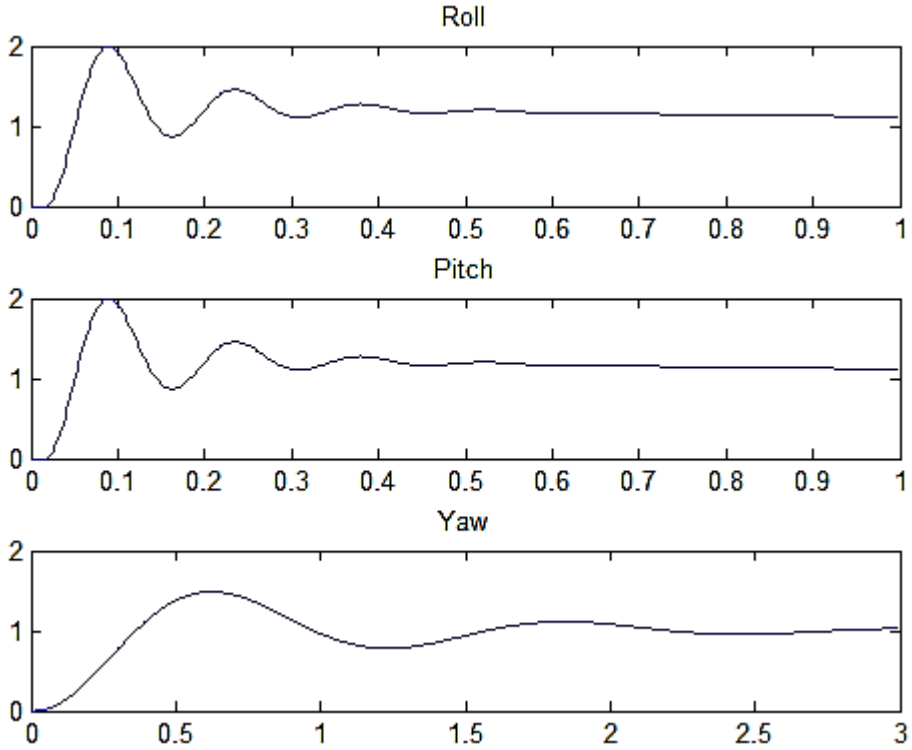


Figure 8.14 : PI controller results.

Table 8.2 : PI controller gains.

	Roll	Pitch	Yaw
K_P	226000	226000	79500
K_I	166000	166000	6980

8.3.3 Proportional-derivative (PD) controller

The block diagram of a PD controller is given below (Figure 8.15).

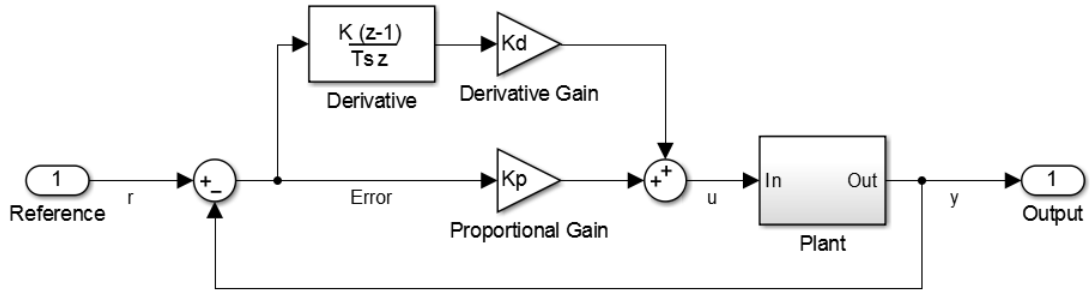


Figure 8.15 : Proportional-derivative controller.

$$u = K_P e + K_D \Delta e \qquad \textbf{(8.6)}$$

Step responses of the axes using PD controller (Table 8.3) are given (Figure 8.16).

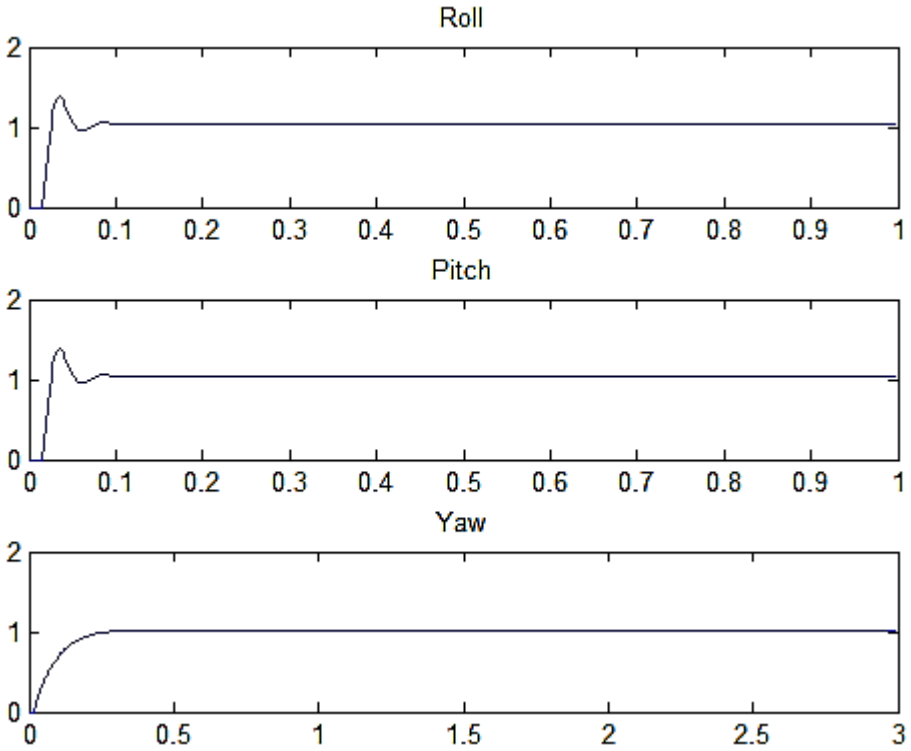


Figure 8.16 : PD controller results.

Table 8.3 : PD controller gains.

	Roll	Pitch	Yaw
K_P	1530000	1530000	121000
K_D	10200	10200	35900

8.3.4 Proportional-integral-derivative (PID) controller

The block diagram of a PID controller is given below (Figure 8.17).

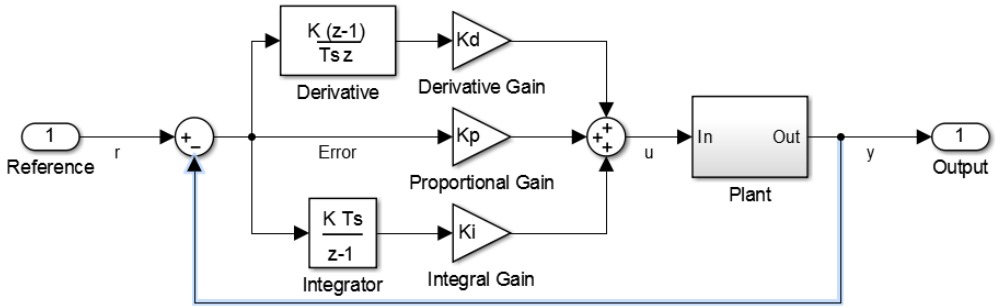


Figure 8.17 : Proportional-integral-derivative controller.

$$u = K_P e + K_I \int e\, dt + K_D \Delta e \tag{8.7}$$

Step responses of the axes using PID controller (Table 8.4) are given (Figure 8.18).

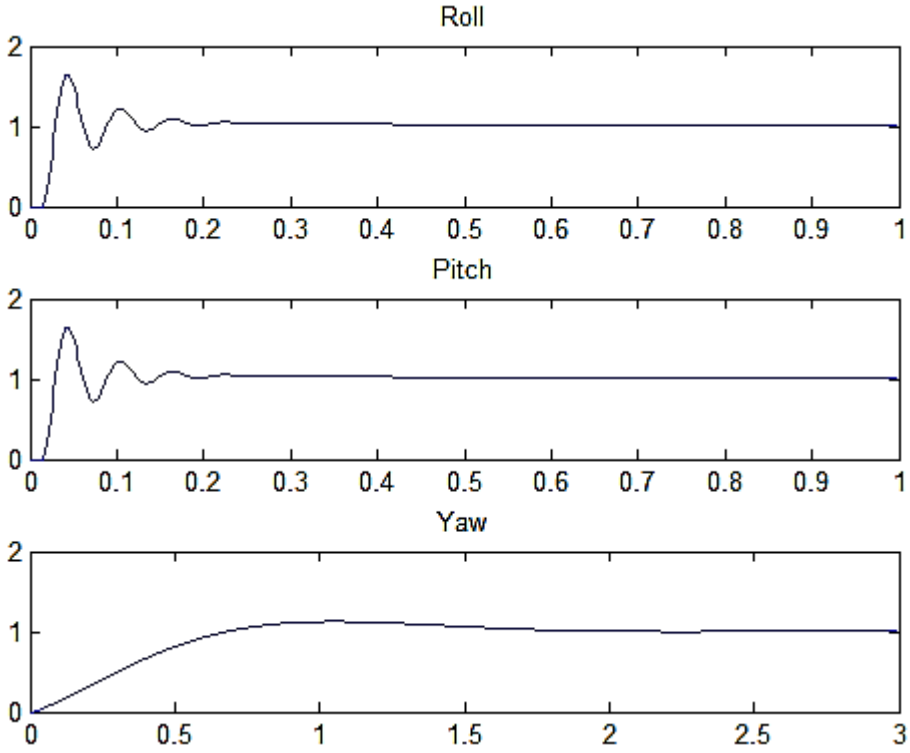


Figure 8.18 : PID controller results.

Table 8.4 : PID controller gains.

	Roll	Pitch	Yaw
K_P	1050000	1050000	30500
K_I	1930000	1930000	1560
K_D	3760	3760	3730

8.3.5 Comparison of controller performances

The presented controllers will be compared in a few aspects, including overshoot percent, settling time and reference tracking errors. The table below (Table 8.5) gives the root-mean-square error values of the controlled system output where error is the difference between the output and the reference input.

Table 8.5 : Controller RMS errors.

	Roll	Pitch	Yaw
P	0.5648	0.5648	0.3131
PI	0.3292	0.3292	0.2987
PD	0.1506	0.1506	0.1340
PID	0.1769	0.1769	0.2736

It can be seen that the PD controller performs better than the others; meaning that it tracks the reference input closer.

The overshoot ratios of the controllers are given in the table below (Table 8.6).

Table 8.6 : Overshoot ratios.

	Roll	Pitch	Yaw
P	50%	50%	40%
PI	100%	100%	60%
PD	40%	40%	1%
PID	70%	70%	5%

With these results, it can be seen that the integral effect causes large overshoots in all of the cases, due to the integration of the error during the dead time, which is caused by the measurement delay. In addition, the P controller has a steady-state error.

The last criterion is the settling time, which is given in the table below (Table 8.7).

Table 8.7 : Settling times.

	Roll	Pitch	Yaw
P	0.3s	0.3s	2.0s
PI	0.4s	0.4s	2.5s
PD	0.1s	0.1s	0.3s
PID	0.2s	0.2s	1.5s

The PD controller settles faster. It is also seen that the settling time of the yaw controller is bigger than the other axes, due to its slower dynamics.

Considering all of these results, the PD controller will be the choice.

9. CONCLUSIONS

This work is on modeling, identification and simulation of quadrotor UAVs using real-time flight data. Within this concept, brief information on unmanned quadrotors is given and a detailed literature summary is presented.

The principles of operation of an unmanned quadrotor are explained. Basic maneuvers as well as the hover are explained.

An unmanned quadrotor is modeled in different ways. The first model is a complete Euler-Lagrange based model while the second one is a Newton-Euler based model. It is seen that the Newton-Euler model is simpler but sufficient for representing most of the system dynamics. This model is even more simplified relying on some reasonable assumptions. Thus, the rotational part of the model is given as three decoupled differential equations such that each equation only depends on a dedicated input.

In addition, electric motor of a quadrotor is modeled and a nonlinear model is presented. Motor driver is also taken into account and its equation is given.

The quadrotor hardware platform that is used in this work is introduced and the details on its hardware and software are given.

A complete telemetry system is implemented on the quadrotor platform to collect real-time flight data. First, the requirements of the telemetry system are listed; then the hardware, software and the data structure of the designed system are described.

The basic principles of an inertial measurement unit (IMU) are given as well as the principles of accelerometer and gyroscope sensors. The well-known sensor fusion methods, which are used to obtain the Euler attitude angles, are described in details. In addition, a Kalman sensor fusion algorithm is implemented and demonstrated.

Using the quadrotor hardware equipped with the telemetry system, real-time flight data is collected. This data is used to identify the quadrotor model and parameters.

The quadrotor used in this work flies with a closed-loop control that takes the desired Euler angles from user, the attitude of the quadrotor from the onboard IMU and sets

pulse width modulation (PWM) signals to the motor. As there is no built-in feedback for motor speeds, the motors are operated in open-loop.

To simulate the system, a two-part model that consists of a motor model and a quadrotor body model is proposed. First, the nonlinear motor model is identified. Then, the flight-data is used to identify the quadrotor body model. Because of a hardware limitation (lack of sensors), only the rotational dynamics of the quadrotor body are identified.

The parameters of the previously presented nonlinear motor model are estimated. The grey-box approach is used thank to the prediction error method (PEM).

Quadrotor body model takes the motor speeds as model input. Because of the lack of motor speed sensors, the identified motor model is used to simulate the motors to obtain motor speeds from the collected flight data. Then quadrotor body model is identified first via grey-box then black-box approaches.

The previously obtained quadrotor body model is used as a nonlinear-grey-box model and its parameters are estimated using PEM. Then the nonlinear model is converted to a quasi-linear form. Several auto-regressive exogenous (ARX) black-box models are estimated and the results are compared. Additionally, cross-correlation tests are applied to the models for model verification.

ARX models gave better results than the grey-box model because of their more flexible structure. The ARX identification also showed that the system has an output delay that comes from the IMU sensors. Accelerometer and gyroscope have internal digital low-pass filters to suppress the noise but they cause a measurement delay.

Finally, a complete simulation scheme for the quadrotor that uses the identified models is designed. All of the hardware properties are applied to that scheme including the joystick, PWM signals, the battery voltage as well as a controller block. Then several proportional, integral and derivative based controllers are designed. It is seen that PI and PID controllers cause large overshoots because of the measurement delay of the model. The PD controller gave the best results.

Thus, it is proven that this work has achieved all of its goals.

REFERENCES

[1] **Kroo, Ilan; Kunz, Peter.** Development of the mesicopter: A miniature autonomous rotorcraft. In: American Helicopter Society (AHS) Vertical Lift Aircraft Design Conference, San Francisco, CA. 2000.

[2] **Kroo, Ilan, Et Al.** The Mesicopter: A miniature rotorcraft concept–phase ii interim report. Stanford university, USA, 2000.

[3] **Hamel, Tarek, et al.** "Dynamic Modelling And Configuration Stabilization for an X4-Flyer." a a 1.2 (2002): 3.

[4] **Altug, Erdinc, James P. Ostrowski, And Camillo J. Taylor.** "Quadrotor control using dual camera visual feedback." In Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, vol. 3, pp. 4294-4299. IEEE, 2003.

[5] **Altug, Erdinc; Ostrowski, James P.; Mahony, Robert.** Control of a quadrotor helicopter using visual feedback. In: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. IEEE, 2002. p. 72-77.

[6] **Hoffmann, Gabe, et al.** "The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)." Digital Avionics Systems Conference, 2004. DASC 04. The 23rd. Vol. 2. IEEE, 2004.

[7] **Tayebi, A., and S. McGilvray.** "Attitude stabilization of a four-rotor aerial robot." Decision and Control, 2004. CDC. 43rd IEEE Conference on. Vol. 2. IEEE, 2004.

[8] **Castillo, Pedro, Alejandro Dzul, and Rogelio Lozano.** "Real-time stabilization and tracking of a four-rotor mini rotorcraft." Control Systems Technology, IEEE Transactions on 12.4 (2004): 510-516.

[9] **Mokhtari, Abdellah, and Abdelaziz Benallegue.** "Dynamic feedback controller of Euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle." Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. Vol. 3. IEEE, 2004.

[10] **Mokhtari, Abdellah, Abdelaziz Benallegue, and Boubaker Daachi.** "Robust feedback linearization and GH∞ controller for a quadrotor unmanned aerial vehicle." Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on. IEEE, 2005.

[11] **Bouabdallah, Samir, Pierpaolo Murrieri, and Roland Siegwart.** "Design and control of an indoor micro quadrotor." In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, vol. 5, pp. 4393-4398. IEEE, 2004.

[12] **Bouabdallah, Samir, and Roland Siegwart.** "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor." Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. IEEE, 2005.

[13] **Mahony, Robert, Tarek Hamel, and Jean-Michel Pflimlin.** "Complementary filter design on the special orthogonal group SO (3)." Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on. IEEE, 2005.

[14] **Pounds, Paul, Robert Mahony, and Peter Corke.** "Modelling and control of a quad-rotor robot." Proceedings Australasian Conference on Robotics and Automation 2006. Australian Robotics and Automation Association Inc., 2006.

[15] **Bouabdallah, Samir, and Roland Siegwart.** "Full control of a quadrotor." Intelligent robots and systems, 2007. IROS 2007. IEEE/RSJ international conference on. IEEE, 2007.

[16] **Bresciani, Tommaso.** Modelling, identification and control of a quadrotor helicopter. Department of Automatic Control, Lund University, 2008.

[17] **Petersen, Christian Fink, et al.** "Autonomous Hovering with a Quadrotor Helicopter." Aalborg: Aalborg Universitet (2008).

[18] **Hussein, W. M., M. M. El-khatib, A. Y. Elruby, and H. M. Haleem.** Quad Rotor Design, Simulation And Implementation. (2009)

[19] **Luo, Ying, et al.** "VTOL UAV altitude flight control using fractional order controllers." 4th IFAC Workshop on Fractional Differentiation and Its Application, Badajoz, Spain. 2010.

[20] **Stanculeanu, Ionel, and Theodor Borangiu.** "Quadrotor black-box system identification." World Academy of Science, Engineering and Technology 78 (2011).

[21] **Falkenberg, Ole, et al.** "Model Identification and H∞ Attitude Control for Quadrotor MAV's." Intelligent Robotics and Applications. Springer Berlin Heidelberg, 2012. 460-471.

[22] **Rich, Matthew.** "Model development, system identification, and control of a quadrotor helicopter." (2012).

[23] **Vanin, Matteo.** "Modeling, identification and navigation of autonomous air vehicles." (2013).

[24] **Li, Qianying.** Grey-Box System Identification of a Quadrotor Unmanned Aerial Vehicle. Diss. TU Delft, Delft University of Technology, 2014.

[25] **Types of Multicopter** <http://blog.oscarliang.net/types-of-multicopter/>, date retrieved 29.05.2015.

[26] **Quadcopter Wiki** <http://en.wikipedia.org/wiki/Quadcopter>, date retrieved 29.05.2015.

[27] **Schilling, Robert J.** Fundamentals of robotics: analysis and control. Vol. 13. New Jersey: Prentice Hall, 1990.

[28] **Maine, Richard E., and Kenneth W. Iliff.** "Application of parameter estimation to aircraft stability and control: The output-error approach." (1986).

[29] **Carrillo, Luis Rodolfo García,** et al. Quad Rotorcraft Control: Vision-Based Hovering and Navigation. Springer Science & Business Media, 2012.

[30] **Luukkonen, Teppo.** Modelling and control of quadcopter. Independent research project in applied mathematics, Espoo, 2011.

[31] **Chovancová, Anežka, Et Al.** Mathematical Modelling and Parameter Identification of Quadrotor (a survey). Procedia Engineering, 2014, 96: 172-181.

[32] **Vendittelli, Marilena**. Elective in Robotics.

[33] **Crazyflie Wiki** <https://wiki.bitcraze.io/projects:crazyflie>, date retrieved 29.05.2015.

[34] **Invensense** <http://www.invensense.com>, date retrieved 29.05.2015.

[35] **Accelerometer** <http://www.sensorwiki.org/doku.php/sensors/accelerometer>, date retrieved 29.05.2015.

[36] **MEMS: A Brief Overview** <http://tr.mouser.com/applications/mems-overview>, date retrieved 29.05.2015.

[37] **Gyroscope** <http://www.sensorwiki.org/doku.php/sensors/gyroscope>, date retrieved 29.05.2015.

[38] **Analog Dialogue** <http://www.analog.com/library/analogDialogue/archives/37-03/gyro.html>, date retrieved 29.05.2015.

[39] **Mau, Sandra.** "What is the Kalman Filter and How can it be used for Data Fusion." *Retrieved July* 15 (2008).

[40] **Katayama, Tohru.** Subspace methods for system identification. Springer Science & Business Media, 2006.

[41] **Selecting a Model Structure in the System Identification Process** <http://www.ni.com/white-paper/4028>, date retrieved 29.05.2015.

[42] **Ljung, Lennart.** "System Identification: Theory for the User, PTR Prentice Hall Information and System Sciences Series." (1999).

CURRICULUM VITAE

Name Surname: Atakan SARIOĞLU

E-Mail **:** atakan.sarioglu@gmail.com, atakan.sarioglu@itu.edu.tr

EDUCATION:

B.Sc. **:** Uludağ University, Electronics Engineering, 2007-2011

M.Sc. **:** Istanbul Technical University, System Dynamics and Control, 2012-2015