POLITECNICO DI TORINO

SCUOLA DI DOTTORATO
Dottorato in Ingegneria Aerospaziale - XXVI ciclo

Tesi di Dottorato

Automatic control of a multirotor

Filippo Rinaldi

Tutori
Prof. Sergio Chiesa
Prof.ssa Fulvia Quagliotti
Prof. Giorgio Guglieri

Coordinatore del corso di Dottorato
Prof.ssa Fulvia Quagliotti

Marzo 2014

Allegato n. 4

COMMISSIONE GIUDICATRICE
DI INGEGNERIA AEROSPAZIALE

Il dott. Filippo Rinaldi ha discusso in data 21-03-2014 presso il DIMEAS del Politecnico di Torino la tesi di Dottorato avente il seguente titolo:

Automatic control of multirotor

Le ricerche oggetto della tesi sono significative e ben sviluppate.

Le metodologie appaiono adeguate e correttamente applicate.

I risultati sono interessanti ed analizzati con adeguato senso critico.

Nel colloquio il candidato dimostra un'approfondita conoscenza delle problematiche trattate.

La Commissione unanime giudica più che soddisfacente il lavoro svolto e propone che al dott. Filippo Rinaldi venga conferito il titolo di **Dottore di Ricerca**.

Data 21-03-2014

Prof. Franco Bernelli Zazzera (Presidente)

Prof. Lorenzo Trainelli (Componente)

Prof. Cecilia Surace (Segretario)

Summary

Objective of this thesis is to describe the design and realisation phases of a multirotor to be used for low risk and cost aerial observation. Starting point of this activity was a wide literature study related to the technological evolution of multirotors design and to the state of the art. Firstly the most common multirotor configurations were defined and, according to a size and performance based evaluation, the most suitable one was chosen. A detailed computer aided design model was drawn as basis for the realisation of two prototypes. The realised multirotors were "X-shaped" octorotor with eight coaxially coupled motors. The mathematical model of the multirotor dynamics was studied. "Proportional Integral Derivative" and "Linear Quadratic" algorithms were chosen as techniques to regulate the attitude dynamics of the multirotor. These methods were tested with a nonlinear model simulation developed in the Matlab Simulink® environment. In the meanwhile the Arduino board was selected as the best compromise between costs and performance and the above mentioned algorithms were implemented using this platform thanks to its main characteristic of being completely "open source". Indeed the multirotor was conceived to be a serviceable tool for the public utility and, at the same time, to be an accessible device for research and studies. The behaviour of the physical multirotor was evaluated with a test bench designed to isolate the rotation about one single body axis at a time. The data of the experimental tests were gathered in real time using a custom Matlab code and several indoor tests allowed the "fine tuning" of the controllers gains.
Afterwards a portable "ground station" was conceived and realised in adherence with the real scenarios users needs. Several outdoor experimental flights were executed with successful results and the data gathered during the outdoor tests were used to evaluate some key performance indicators as

the endurance and the maximum allowable payload mass. Then the fault tolerance of the control system was evaluated simulating and experimenting the loss of one motor; even in this critical condition the system exhibited an acceptable behaviour.
The reached project readiness allowed to meet some potential users as the "Turin Fire Department" and to cooperate with them in a simulated emergency. During this event the multirotor was used to gather and transmit real time aerial images for an improved "situation awareness".
Finally the study was extended to more innovative control techniques like the neural networks based ones. Simulations results demonstrated their effectiveness; nevertheless the inherent complexity and the unreliability outside the training ranges could have a catastrophic impact on the airworthiness. This is a factor that cannot be neglected especially in the applications related to flying platforms.
Summarising, this research work was addressed mainly to the operating procedures for implementing automatic control algorithms to real platforms. All the design aspects, from the preliminary multirotor configuration choice to the tests in possible real scenarios, were covered obtaining performances comparable with other commercial off-the-shelf platforms.

Sommario

Obiettivo di questa tesi è la descrizione delle fasi di progettazione e realizzazione di una piattaforma multirotorica per l'osservazione aerea a basso rischio e costo. Punto di partenza del lavoro è stata una vasta ricerca storico-bibliografica effettuata allo scopo di conoscere l'evoluzione tecnologica nel design dei multirotori ed il relativo stato dell'arte. Sono state dunque individuate le principali configurazioni di multirotori e sulla base della valutazione delle dimensioni minime e delle potenziali prestazioni ottenibili è stata scelta la configurazione ottima tra quelle possibili. Successivamente è stato realizzato un modello CAD di dettaglio sulla cui base sono stati costruiti due prototipi di multirotore con otto rotori coassiali a coppie. Constestualmente è stata effettuato uno studio di dettaglio sulla modellazione matematica del comportamento dinamico del prototipo. Sono stati inoltre progettati due regolatori automatici, uno di tipo "Proporzionale Integrale Derivato" e l'altro di tipo "Lineare Quadratico" per il controllo della dinamica di assetto del velivolo costruito. Questi controllori sono stati validati in ambiente Matlab Simulink® verificandone l'interazione con il modello non lineare del velivolo. Nel contempo è stata eseguita un' indagine di mercato sui principali produttori di hardware programmabili per il controllo automatico ed è stata selezionata la scheda Arduino come miglior compromesso tra costo ed adeguatezza al raggiungimento degli obiettivi prefissati. Lo studio delle caratteristiche di tale scheda elettronica e del relativo linguaggio di programmazione ha permesso di implementare su tale piattaforma gli algoritmi di controllo delle tecniche precedentemente menzionate. E' d'uopo menzionare che una caratteristica peculiare del sistema di controllo utilizzato è la logica *open source*, valida sia per quanto attiene all'hardware che per quanto concerne il software. Il velivolo è stato difatti concepito con l'obiettivo di essere immediatamente utilizzabile per

l'osservazione aerea ed essere, al tempo stesso, un dispositivo di studio e ricerca con il quale sperimentare, senza vincoli imposti dal produttore, nuove leggi di controllo automatico ed ulteriori funzionalità. Il comportamento del velivolo è stato dunque testato mediante un banco prova sviluppato *ad hoc* con l'obiettivo di isolare la variazione di uno solo dei tre angoli di Eulero per ogni test di dinamica eseguito. Al fine di valutare rapidamente e con precisione l'efficacia del sistema di controllo, è stato sviluppato, in ambiente Matlab, un software di acquisizione ed elaborazione dei parametri di assetto e dei comandi acquisiti in tempo reale. L'esecuzione di numerosi test al banco prova ha permesso di effettuare il *fine tuning* dei guadagni dei controllori. L'attività descritta è stata seguita dall'assemblaggio di una *ground station* portatile realizzata in modo da poter rispondere alle esigenze degli utenti in uno scenario reale. E' stata dunque eseguita una campagna di test sperimentali nella quale è stato valutato e confrontato l'effetto dei controllori automatici sulla dinamica di assetto del velivolo. I dati raccolti sono stati utilizzati per verificare le stime dei parametri prestazionali più limitanti per questa tipologia di velivoli quali l'autonomia di durata ed il massimo carico utile. Successivamente è stato sperimentato, prima in simulazione e poi realmente, il comportamento del velivolo in caso di perdita totale di spinta da parte di uno degli otto rotori disponibili. I test, che hanno avuto esito positivo, hanno permesso di verificare la robustezza dei controllori di assetto implementati, nei limiti di accettabilità prevedibili per un possibile impiego reale. La maturità raggiunta dal progetto ha reso fattibile l'incontro con potenziali utenti. Particolare interesse è stato manifestato dai Vigili del Fuoco del Comune di Torino con i quali è stata avviata una stretta collaborazione. Questo ha permesso di partecipare attivamente a simulazioni di calamità naturali fornendo, ai gestori del piano di emergenza, la trasmissione, in tempo reale, di immagini aeree dell'area interessata e garantendo dunque la necessaria *situation awareness*.
Parallelamente è stata data enfasi allo studio dei fondamenti teorici alla base di più innovative tecniche di controllo automatico. In particolare sono state utilizzate le tecniche basate sull'uso di reti neurali per costruire, al calcolatore, delle simulazioni di sistema *closed loop*. I risultati ottenuti hanno

permesso di dimostrare la potenziale efficacia di questo strumento quando utilizzato nei range di addestramento della rete neurale. Tuttavia l'attività di ricerca eseguita in tale ambito ha permesso di evidenziare che l'intrinseca complessità e l'inattendibilità del controllore al di fuori dei range di addestramento potrebbero avere effetti catastrofici sulla condotta di un volo e pertanto esistono ancora necessari margini di perfezionamento prima che queste tecniche possano diffondersi, con adeguata garanzia di successo, nel settore aeronautico.

In conclusione l'attività di ricerca è stata principalmente indirizzata all'apprendimento delle modalità operative di implementazione di leggi di controllo su piattaforme aeree reali. Gli aspetti progettuali inerenti lo sviluppo di tali sistemi sono stati curati nella loro interezza, dalla fase preliminare di scelta della configurazione alla realizzazione e sperimentazione in scenari operativi, evidenziando *performance* comparabili a quelle di velivoli commerciali della medesima categoria disponibili sul mercato.

To my darling wife

The master in the art of living makes little distinction
between his work and his play, his labor and his leisure,
his mind and his body, his education and his recreation,
his love and his religion. He hardly knows which is which.
He simply pursues his vision of excellence at whatever he does,
leaving others to decide whether he is working or playing.
To him he is always doing both.
From the Zen Buddhist Text

Ringraziamenti

Essere un Ufficiale dell'Aeronautica Militare e contemporaneamente uno studente di Dottorato é un impegno indiscutibilmente arduo. Al fine di portarlo a termine é necessario tenere alta la concentrazione per oltre tre anni nei quali le responsabilità personali e lavorative crescono velocemente. Nella pratica questo non é possibile senza essere guidati da una forte passione e curiosità per il proprio lavoro e per la materia oggetto di ricerca. Sebbene condizione necessaria questo non é sufficiente per il conseguimento dell'obiettivo. É fondamentale, difatti, essere in un ambiente stimolante come quello che ho avuto il privilegio di trovare al Politecnico di Torino dove ho incontrato i miei tutori, Prof. Sergio Chiesa, Prof.ssa Fulvia Quagliotti and Prof. Giorgio Guglieri. A loro va il mio primo ringraziamento per la dedizione con la quale si dedicano alla crescita morale e tecnica dei loro studenti costruendo le basi di un futuro migliore per tutti. Durante questi tre anni ho anche avuto la fortuna di conoscere un giovane ingegnere di talento, Alessio Gargioli, che ha redatto la sua Tesi di Laurea sotto la mia supervisione; credo di aver imparato da lui più di quanto gli abbia insegnato. Sono stato inoltre supportato dai miei amici e colleghi del 1° Reparto Manutenzione Velivoli di Cameri. In particolare ringrazio Roberto Farris, Massimo Mavilio ed Antonio Ciotola perché con la loro curiosità, saggezza e simpatia mi hanno incoraggiato nel cominciare e percorrere il cammino di Dottorato. Il mio pensiero va ora alla mia famiglia che ha sempre avuto fiducia in me permettendomi di scegliere in libertà e consapevolezza il mio futuro. Voglio infine ringraziare la persona che é sempre stata al mio fianco, mia moglie Maria Antonietta; abbiamo dovuto spesso sacrificare le nostre vite private ma l'abbiamo fatto con la felicità di essere insieme a condividere gli stessi sogni, costruendo il futuro della nostra splendida famiglia, la più grande conquista della mia vita.

Contents

List of Figures

LIST OF FIGURES

List of Tables

Acronyms

A.D.C. Analog to Digital Converter

A.P.M. Ardupilot Mega Board

AFV Autonomous Flying Vehicle

BFGS Broyden, Fletcher, Goldfarb and Shanno algorithm

C.O.T.S. Commercial off-the-shelf

CAD Computer Aided Design

CEA Centre d'Energie Atomique

COG Centre Of Gravity

DC Direct Current

DSP Discrete Signal Processor

e.g. Exempli Gratia

EEPROM Electrically Erasable Programmable Read-Only Memory

EKF Extended Kalman Filter

EPFL École Polytechnique Fédérale de Lausanne

G.C.S. Ground Control Station

G.P.S. Global Positioning System

i.e. Id Est

I.M.U. Inertial Measurement Unit

INS Inertial Navigation System

IR Infrared

L.G.P.L. Lesser General Public License

LiPo Lithium Polymer

LPF Low Pass Filter

LQR Linear Quadratic Regulator

M.I.P.S. Million Instructions Per Second

MAV Micro Aerial Vehicle

MB Megabyte

MEMS Micro Electro-Mechanical System

MIT Massachusetts Institute of Technology

NARMA Nonlinear Autoregressive-Moving Average

P.C.B. Printed Circuit Board

PC Personal Computer

PID Proportional Integral Derivative

PWM Pulse Width Modulation

QRT Quad-Rotor Type

R.C. Radio Control

RPV Remotely Piloted Vehicle

SRAM Static Random Access Memory

STARMAC Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

UTC Université de Technologie de Compiègne

vs. Versus

VTOL Vertical Take-Off and Landing

1

Introduction

1.1 UAVs

Unmanned Aerial Vehicles (UAVs) are autonomous or remotely piloted aircraft. They range in size from full-scale craft, similar to those flown by humans, to miniature aircraft centimetres in size. UAVs are driven by a variety of power plants, including petrol engines, gas turbines and electric motors. The utility of UAVs in military applications is readily apparent, UAVs can potentially carry out the range of tasks normally executed by piloted aircraft without placing human pilots in jeopardy. However, these benefits also carry over to civilian aircraft that operate in hazardous conditions or require tedious or onerous piloting during lengthy operations. For example, unmanned aircraft could carry out power-line inspection in close proximity to live electrical cables, a task currently performed by manned aircraft as reported in [4]. Autonomous rotorcraft also have the potential to revolutionise commercial practice in a variety of fields such as mining, infrastructure and agriculture, which do not presently employ aircraft due to the size and expense of full-scale vehicles as detailed in [5]. Small-scale UAVs, or "Micro Air Vehicles" (MAVs), expand the range of possible aero-robot duties further with their high portability and ability to operate in small spaces as reported in [6]. Recent advances in miniaturisation, battery and control technology have made very small rotorcraft possible [7].

1.2 Multirotors

Multirotors are a special form of rotorcraft UAV that use pairs of counterrotating rotors to provide lift and directional control. Unlike conventional helicopters, multirotors typically have fixed-pitch blades and vary their thrust by changing rotors speed. Flight attitude is regulated entirely by rotors speed. When the vehicle tilts, a component of the thrust is directed sideways and the aircraft translates horizontally. Two major motivators for multirotors are reliability and compactness, both are essential for a system that will be portable and useful in close proximity to people and structures in commercial applications. Conventional helicopters are mechanically very complex. They rely on a complex, adjustable mechanism that causes each blade to go through a complete pitch cycle each revolution of the rotor, providing attitude control of the rotor plane that, in turn, is used to control airframe attitude. The most common system used is a "swashplate" structure that consists of two parallel moving bearings fixed on the rotor mast to transmit angular displacement to the pitch horns of the rotor blades (see Fig. 1.1). Small helicopters may further require a Bell-Hillier stabilizer linkage to

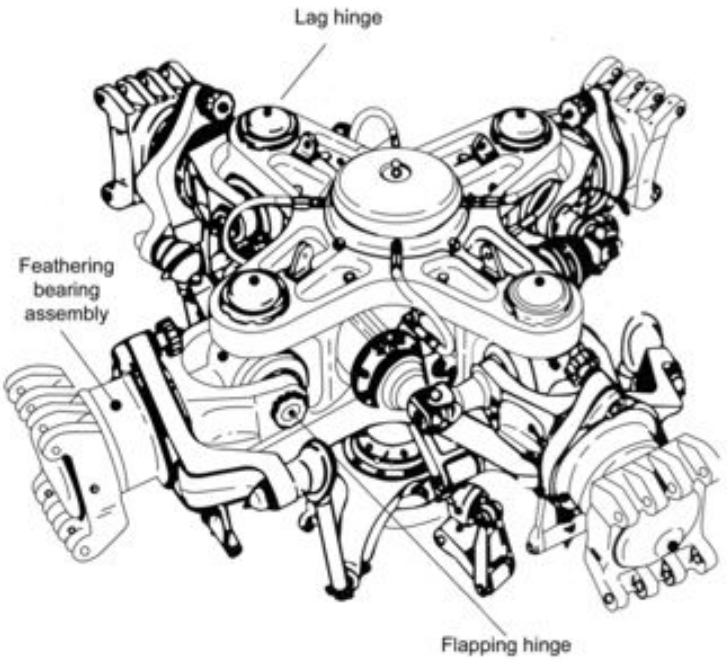

Figure 1.1: Full-scale helicopter swashplate [1]

slow the natural dynamic response of the rotor. Swashplates are sophisticated pieces of high-speed machinery operating in a vibrating environment and are highly prone to failure without constant maintenance. Failure of the swashplate causes catastrophic loss of cyclic control and, typically, destruction of the vehicle. The inherent mechanical robustness of electric multirotors stems from the simplicity of the rotor head. The easy and inexpensive maintenance required by multirotors is a key consideration for civilian craft that must operate reliably in proximity to humans, without regular skilled maintenance. The compactness of multirotors is due to reduced rotor diameters and closely spaced layout. They do not have a single large rotor or long tail boom that can readily collide with nearby obstacles and, instead, use small rotors that are easily shrouded for protection. This makes them ideal for tasks indoors or in enclosed spaces, such as inspecting ceilings of a factory, flying down mine shafts or scanning close to civil infrastructure such as bridges or dam walls.

1.3 Historial context

The utility of unmanned aerial vehicles has always been dictated by the technology available to control and direct the craft. As early as 1917 (only 14 years after the invention of the aeroplane itself) Elmer Sperry constructed a self-stabilising aircraft using gyroscopes, barometers and servo-motor control [8]. After take-off controlled by a human, the Hewitt-Sperry Automatic Aeroplane was capable of flying up to 48 km and dropping a bag of sand within 3.2 km of a predefined target. The first fully-unmanned flight was the 1918 Curtis-Sperry Flying Bomb, which was launched from a moving car and flew a preset distance of 900 m [9]. In the 1930s, development continued on both sides of the Atlantic, but the emphasis was on radio-controlled drones for target practice rather than on autonomous vehicles. The outbreak of the Second World War in 1939 prompted renewed interest in flying bombs. Advances in radio, gyroscopic control technology and television produced more sophisticated weapons, but with mixed results. The Allies focused on radio-control of modified bombers, using telemetry taken from cameras in the nose looking forward and in the cockpit pointed at the instruments. These attempts had only limited success. The Axis flying bombs, specifically the V-1, enjoyed great notoriety for their part in the London blitz. The V-1 used a weighted

pendulum for attitude control, a gas-powered gyroscope compass for bearing and a barometer for altitude control [8]. A free-wheeling propeller at the front of the craft estimated distance and caused the bomb to dive when a preset number of rotations was reached. In practice the V-1 was as inaccurate as other flying bombs of the era, but the sheer number of launches accounted for more than 6,000 casualties. Captured V-1s catalysed the Allies to continue developing cruise-missile, Remote-Piloted Vehicle (RPV) and radio-controlled drone technology, which formed the basis of modern UAVs. Notable among the early post-war RPVs was the QH-50 Gyrodyne (see Fig. 1.2), the first unmanned helicopter. Developed for anti-submarine warfare in 1950, the Gyrodyne was remotely piloted from ships and used gyroscope feedback control stability in the air [10]. Post-war cruise-missiles such as Navaho and Matador advanced the capa-

Figure 1.2: QH-50 gyrodyne ASW UAV

bilities of fixed-wing drones. The N-69 Snark and X-10 Navaho introduced an Inertial Navigation System (INS) to manoeuvre through a trajectory on approach to its target [8]. The TM-61C Matador had a microwave-based positioning system that allowed it to map its location using signals received from known transmitters. The TM-76A had INS and down-looking terrain-following radar. Drones such as the MQM-57 Falconer and Model 147J Lightning Bug added cameras and automated flight capability to remotely-piloted aircraft; they were used for reconnaissance missions over China and the Soviet Union after the loss of several U-2 spy planes in the 1960s. This technology culminated in the SLCM Tomahawk missile, which features INS, Global Positioning System (GPS), terrain-following radar and terminal guidance based on video feature

recognition. The Tomahawk was used to good effect during the 1992 Gulf War, demonstrating a 94% strike rate in its first combat deployment [11]. Today, robot aircraft combine modern computer power with technology originally developed for drones and cruise missiles to perform a variety of roles including reconnaissance, surveillance, air-to-ground and air-to-air attack missions. Progress in computers, light-weight cameras and Micro Electro-Mechanical System (MEMS) inertial sensors [12] has now made UAV technology affordable for non-military use.

1.4 Manned Quadrotors

The first manned quadrotor was the Bréguet-Richet "Gyroplane No. 1" constructed in 1907 (see Fig. 1.3). The gyroplane consisted of a cross-beam fuselage with four

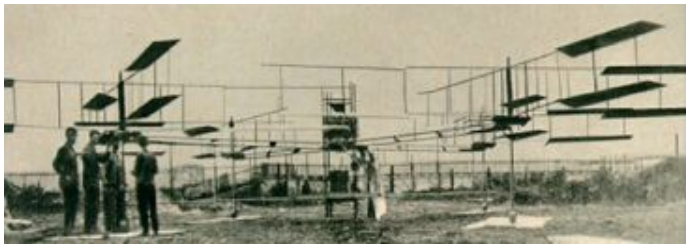

Figure 1.3: Bréguet-Richet Gyroplane No.1 - The First Quadrotor

bi-plane rotors (for a total of 32 blades) at each end. The machine could carry a small person but it never flew outside of ground effect. As can be read in [13], its handling was reported to be poor and it required a team of men to stabilise it during hovering flight. Other early quadrotors that achieved flight were the 1921 Æhmichen quadrotor and 1922 Jerome-de Bothezat quadrotor "Flying Octopus" [2]. Two notable manned quadrotor craft were built during the 1960s as part of the United States "X-Plane" research vehicle series. The Curtiss-Wright X-19 (see Fig. 1.4) was a quad tilt-rotor the size of a business jet that used a special type of radial propeller. The propellers used high-angle high-twist rotors to induce vertical thrust even when the rotors were aligned horizontally. The X-19 was destroyed on its first test flight and the radial lift rotor technology was not developed further [14]. The Bell X-22 was a quad ducted-fan craft that saw long service as a research vehicle (see Fig. 1.5). The X-22 could be configured to emulate the flight behaviour of theoretical aircraft and was used as

Figure 1.4: Curtiss-Wright X-19 Radial Propeller Craft

Figure 1.5: Bell X-22 Ducted Fan Research Vehicle

a test-bed for the Hawker Siddeley GR.1 Harrier [14]. Both the X-19 and the X-22 used variable pitch rotors for attitude control and the X-22 had additional vanes in its outflow to allow for low-speed yaw control. Following the success of the V-22 Osprey tilt-rotor, Boeing produced conceptual designs for a quad tilt-rotor based on the same technology. Although no aircraft has yet been built, quad tilt-rotor models have been tested in wind tunnels for aeroelastic loading of its wings and surfaces (see Fig. 1.6) as well as in simulation with complex Computational Fluid Dynamics (CFD) programs for analysis of inflow behaviour and vortex-ring states that plagued the V-22.

Figure 1.6: Boeing Quad Tilt-rotor Half-model in the Langley Wind Tunnel

1.5 Micro Quadrotors

In the last 15 years the number and variety of micro quadrotor vehicles has increased substantially. Early efforts to build small quadrotors were based upon radio-controlled toys. The Hoverbot, built in 1992, was constructed from four radio-controlled helicopters joined at the tail [15] (see Fig. 1.7). The aircraft could lift off in a test frame and stabilise itself in orientation using potentiometers built into its test gimbal. It used variable pitch on all four rotors to change thrust. The mid-90s "Roswell Flyer" and "HMX-4", later to become the "Draganflyer", consisted of cheap motors and rotors, a foam frame and early MEMS gyros in feedback for pilot-assist. The craft were very light and small, limited to carrying tens of grams of payload. Flying the craft required continuous pilot attention. This craft has formed the basis of numerous research vehicles. The "Mesicopter" was a late-90s Stanford University project aimed at creating

Figure 1.7: Borenstein Hoverbot

centimetre-scale quadrotors. The total aircraft weight was of the order of a gram and special wafer-cut moulds were required to fabricate its rotors. The first "Mesicopter" prototypes had fixed-pitch rotors in a conventional quadrotor configuration, but later models used shrouded rotors with inverted mass and a passive aerodynamic system with rotor cowls and fixed vanes for control [16] (see Fig. 1.8). Post-2000, quadrotors have

Figure 1.8: Stanford "Mesicopter" Micro UAV

proliferated as toys and research tools. The Draganfly Innovations produced several multirotor versions aimed at professional applications as shown in [17] and illustrated in Fig. 1.9. The basic Draganflyer quadrotor lifts approximately 250 g of payload for about 10 minutes. The pilot must stabilise the craft with the assistance of damping from rate gyros, although more advanced models can self-stabilise using ultrasonic sensors. Draganflyer parts are used by many control and robotics researchers around the world. The number of purpose-built quadrotors is low, compared with derivative craft, due to the high overheads involved in constructing aircraft from scratch. Typically, a research quadrotor will consist of Draganflyer chassis, rotors and motors complemented

Figure 1.9: Draganflyer "X4-P"

by custom avionics and control. Numerous universities have used quadrotors for research into attitude control, visual servoing, swarm control and aerodynamics. The following is only a brief overview of selected quadrotor research projects. CEA's "X4-Flyer" project seeks to develop quadrotor technology for intuitive pilot operation and operation in hazardous environments [18]. This quadrotor is a novel departure from other modified Draganflyers in that it doubles the number of blades on each motor as illustrated in Fig. 1.10.

It also has custom drive electronics consisting of 1 GHz Discrete Signal Processor

Figure 1.10: CEA's "X4-Flyer"

(DSP) card that provides excellent flight stability. In 2008 Guenard added four ducts around the rotors [19]. Bourquardez used visual feedback in an outer control loop for position and altitude [20]; the system can guide the CEA's quadrotor through waypoints using a single down-facing camera. The École Polytechnique Fédérale de Lausanne (EPFL) "OS4" project is aimed at developing autonomous indoor Vertical Take-Off and Landing (VTOL) vehicles [21], capable of using different navigation schemes. The

"OS4" quadrotor began as a Draganflyer test-bed on a gimbal but has evolved into an entirely original vehicle, including custom avionics, airframe and rotors (see Fig. 1.11). The craft has been successfully used for testing a variety of control schemes as

Figure 1.11: EPFL "OS4" Quadrotor

detailed in [22, 23]. Tayebei and McGilvray have investigated quadrotors deeply, focusing on quaternion and nonlinear control [24]. Their experimental apparatus consists of a non-flying modified Draganflyer with original airframe and drive systems, but with custom avionics. The quadrotor is fixed to a ball-joint test rig with off-board power that allows limited rotation in all three axes [25]. The Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC) project uses multiple vehicles flying in formation for collision and obstacle avoidance [26]. Quadrotors were chosen for this project because they are not as cumbersome as other rotorcraft and can operate in small environments. The STARMAC quadrotors use Draganflyer rotors and motors (see Fig. 1.12), but incorporate a sliding-mode controller for attitude stability [27]. MIT's Aerospace Controls Laboratory uses quadrotors for UAV swarm experiments. As many as five quadrotors may fly simultaneously and cooperate with Unmanned Ground Vehicles (UGVs). In one experiment, a quadrotor was landed successfully on a moving UGV. MIT uses unmodified Draganflyers with onboard video, which are controlled by off-the-shelf hobby radios via a PC interface connected to the handset's "trainer port" [28]. KITECH's Division of Applied Robotic Technology Quad-Rotor Type (QRT) is designed to investigate quadrotor technology for use in indoor emergency observation applications [29]. The QRT consists of a 1.5 kg custom-made flyer built around a Draganflyer chassis. It uses rigid hobby propellers, driven by geared

Figure 1.12: Stanford STARMAC Quadrotor

motors with encoders, that produce a total maximum "mass thrust" of 1.8 kg for 20% headroom; no information is given on flight time or non-battery payload. The QRT uses custom avionics, INS and has an onboard camera, IR and ultrasonic sensors. UTC's Centre de Recherche de Royallieu quadrotor project aims to develop simple control strategies for four-rotor helicopters [30, 31]. The early hardware setup was similar to that used by MIT, the quadrotor was an unmodified Draganflyer with mounted inertial sensors transmitting wirelessly to a PC interfaced to a hobby radio handset. In this case, the PC interface card is connected to the handset potentiometers, rather than to a "trainer port" [32]. More recently, the onboard system was replaced with custom electronics built around the Rabbit Microprocessor RCM3400 core, which reads inertial sensors, controls the motors and communicates over a wireless modem [33]. It includes onboard video transmitting to an offboard PC that sends command signals via the radio handset. The University of Pennsylvania quadrotor project focuses on vision control for autonomous UAV rotorcraft. The quadrotor consists of an HMX-4 connected to a tether that allows it to fly vertically and pitch, roll and yaw without lateral translation [34]. A pair of cameras connected to a PC detect flyer position and pose, using coloured blobs attached to the craft. A second PC receives the pose information and controls the orientation of the flyer via a parallel port "remote control device" [35]. Next to the Hoverbot, the largest quadrotor found was the 6.2 kg Cornell Autonomous Flying Vehicle (AFV). The craft was custom-built and consisted of hobby rotors, motors, speed controllers and early lithium polymer batteries. A try-and-see method was used to find the best mix of rotors, motors and gearing. The craft used rotor speed control loops

via shaft encoders and performed bias estimation for its inertial sensors using Kalman filters. Although the vehicle achieved hover stability on a test platform with tethered power, damage during testing prevented free flight experiments.

1.6 Multirotor most common configurations

This section presents an overview of the most common multirotor *flight* and *design* configurations. They are shown from Fig. 1.13 to Fig. 1.19.
Fig. 1.13 shows two possible tricopter flight configurations; in the first one, to the left, all the rotors are co-rotating; in the second one, to the right, one of the three rotors (coloured in green) counterrotates. In both configurations the rotation axis of one motor can be modified by means of a servo motor.

Figure 1.13: Trirotor flight configurations

Fig. 1.14 shows the, probably, most widely diffused configuration: the quadrotor. Quadrotors, as well as hexarotors (see Fig. 1.15) and octorotors (see Fig. 1.16), can be controlled to fly either in "+ flight configuration", i.e. pointing an arm in the navigation direction, or in "X flight configuration", i.e. with the navigation direction in the middle plane between two arms.

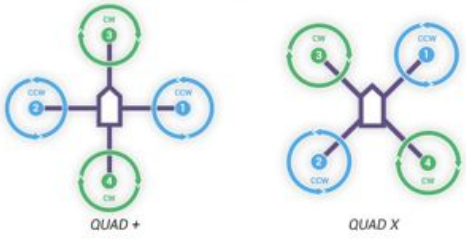

Figure 1.14: Quadrotor + and X flight configurations

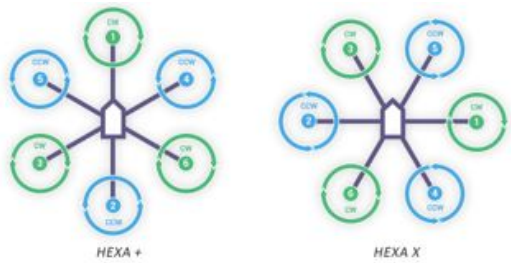

Figure 1.15: Hexarotor + and X flight configurations

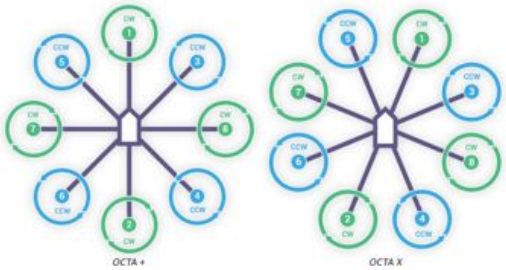

Figure 1.16: Octorotor + and X flight configurations

Some other, less common, design configurations, more suitable for heavier payload and major safety demanding flight missions, are possible. Fig. 1.17 shows the octorotor V design configuration in which the 8 motors are equally split on two divergent bars.

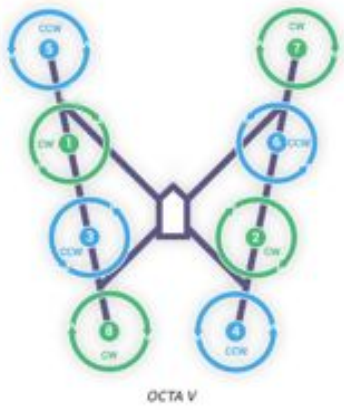

Figure 1.17: Octorotor V design configuration

Fig. 1.18 presents the hexarotor Y design configuration in which the multirotor is lifted by 6 coaxially coupled rotors.

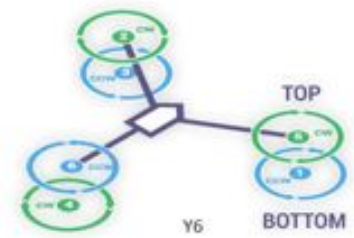

Figure 1.18: Hexarotor Y design configuration

The case in which 8 rotors are coaxially coupled represents the octorotor X design configuration (see Fig. 1.19).

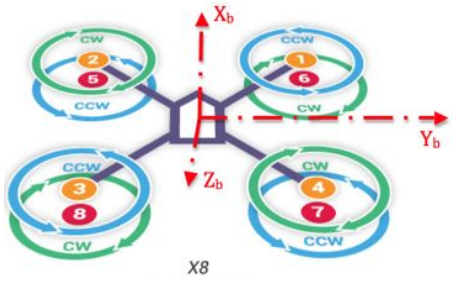

Figure 1.19: Octorotor X flight and design configuration

This thesis presents a thorough analysis of an octorotor, in flight and design X configuration, named *Qx-Rotor*, developed at Politecnico di Torino, Italy. This design configuration has been chosen for its higher thrust to weight ratio among the most common ones. Fig. 1.20 shows the *Qx-Rotor* preliminary computer aided design (CAD), while Fig. 1.21 shows the realized prototype.

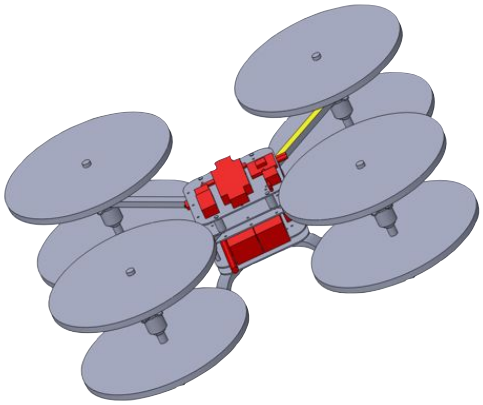

Figure 1.20: Octorotor CAD

Figure 1.21: *Qx-Rotor*: the realized prototype

2

Reference frames

This chapter describes the various reference frames and coordinate systems that are used to define the position and the orientation of an aircraft and the transformation between these coordinate systems. It is necessary to use several different coordinate systems for the following reasons:

- Newton's equations of motion are derived relative to a fixed, inertial reference frame. However, motion is most easily described in a body-fixed frame.
- Aerodynamic forces and torques act on the aircraft body and are most easily described in a body-fixed reference frame.
- On-board sensors like accelerometers and rate gyros measure information with respect to the body frame. Alternatively, GPS measures position, ground speed and course angle with respect to the inertial frame.
- Most mission requirements like loiter points and flight trajectories, are specified in the inertial frame. In addition, map information is also given in an inertial frame.

One coordinate frame is transformed into another through two basic operations: rotations and translations. Section 2.1 describes rotation matrices and their use in transforming between coordinate frames. Section 2.2 describes the specific coordinate frames used for MAV systems. In section 2.3 we derive the Coriolis formula which is the basis for transformations between translating and rotating frames.

2.1 Rotation Matrices

We begin by considering the two coordinate systems shown in Fig. 2.1. The vector $\mathbf{p}$

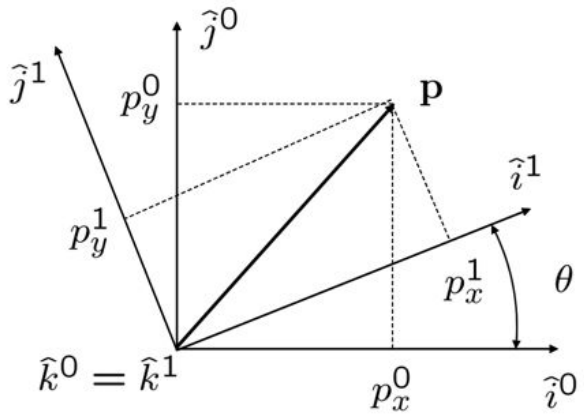

Figure 2.1: Rotation in 2D

can be expressed in both the $\mathcal{F}^0$ frame (specified by $(\hat{i}^0, \hat{j}^0, \hat{k}^0)$) and in the $\mathcal{F}^1$ frame (specified by $(\hat{i}^1, \hat{j}^1, \hat{k}^1)$). In the $\mathcal{F}^0$ frame we have

$$\mathbf{p} = p_x^0 \hat{i}^0 + p_y^0 \hat{j}^0 + p_z^0 \hat{k}^0.$$

Alternatively in the $\mathcal{F}^1$ frame we have

$$\mathbf{p} = p_x^1 \hat{i}^1 + p_y^1 \hat{j}^1 + p_z^1 \hat{k}^1.$$

Setting these two expressions equal to each other gives

$$p_x^1 \hat{i}^1 + p_y^1 \hat{j}^1 + p_z^1 \hat{k}^1 = p_x^0 \hat{i}^0 + p_y^0 \hat{j}^0 + p_z^0 \hat{k}^0.$$

Taking the dot product of both sides with $\hat{i}^1, \hat{j}^1, \hat{k}^1$ respectively and stacking the result into matrix form gives

$$\mathbf{p}^1 \triangleq \begin{pmatrix} p_x^1 \\ p_y^1 \\ p_z^1 \end{pmatrix} = \begin{pmatrix} \hat{i}^1 \cdot \hat{i}^0 & \hat{i}^1 \cdot \hat{j}^0 & \hat{i}^1 \cdot \hat{k}^0 \\ \hat{j}^1 \cdot \hat{i}^0 & \hat{j}^1 \cdot \hat{j}^0 & \hat{j}^1 \cdot \hat{k}^0 \\ \hat{k}^1 \cdot \hat{i}^0 & \hat{k}^1 \cdot \hat{j}^0 & \hat{k}^1 \cdot \hat{k}^0 \end{pmatrix} \begin{pmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{pmatrix}.$$

From the geometry of Fig. 2.1 we get

$$\mathbf{p}^1 = R_0^1 \mathbf{p}^0, \tag{2.1}$$

where

$$R_0^1 \triangleq \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The notation R_0^1 is used to denote a rotation matrix from coordinate frame $\mathcal{F}^0$ to coordinate frame $\mathcal{F}^1$. Proceeding in a similar way, a right-handed rotation of the coordinate system about the y-axis gives

$$R_0^1 \triangleq \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix},$$

and a right-handed rotation of the coordinate system about the x-axis results in

$$R_0^1 \triangleq \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix}.$$

As pointed out in [36], the negative sign on the sin term appears above the line with only ones and zeros. The matrix R_0^1 in the above equations are examples of a more general class of rotation matrices that have the following properties:

P.1. $(R_a^b)^{-1} = (R_a^b)^T = R_b^a$.
P.2. $R_b^c R_a^b = R_a^c$.
P.3. $\det\left(R_a^b\right) = 1$.

In the derivation of Eq. (2.1) note that the vector $\mathbf{p}$ remains constant and the new coordinate frame $\mathcal{F}^1$ was obtained by rotating $\mathcal{F}^0$ through a *righted-handed* rotation of angle θ. We will now derive a formula, called the *rotation formula* that performs a *left-handed* rotation of a vector $\mathbf{p}$ about another vector $\hat{n}$ by an angle of μ. Our derivation follows that given in [36]. Consider Fig. 2.2 which is similar to Fig. 1.2-2 in [36]. The vector $\mathbf{p}$ is rotated, in a left-handed sense, about a unit vector $\hat{n}$ by an angle of μ to produce the vector $\mathbf{q}$. The angle between $\mathbf{p}$ and $\hat{n}$ is ϕ. By geometry we have that

$$\mathbf{q} = \vec{ON} + \vec{NW} + \vec{WQ}. \tag{2.2}$$

The vector $\vec{ON}$ can be found by taking the projection of $\mathbf{p}$ on the unit vector $\hat{n}$ in the direction of $\hat{n}$:

$$\vec{ON} = (\mathbf{p} \cdot \hat{n})\hat{n}.$$

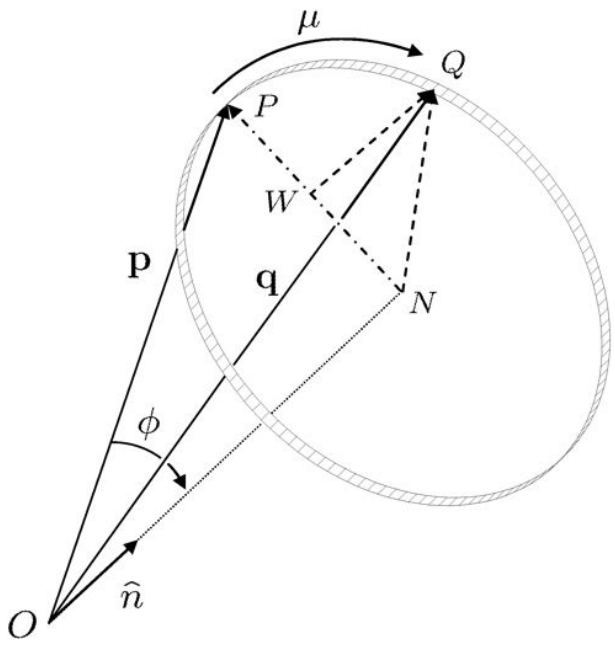

Figure 2.2: Left-handed rotation of a vector $\mathbf{p}$ about the unit vector $\hat{n}$ by an angle of μ to obtain the vector $\mathbf{q}$

The vector $\vec{NW}$ is in the direction of $\mathbf{p} - \vec{ON}$ with a length of $NQ\cos\mu$. Noting that the length NQ equals the length NP which is equal to $\|\mathbf{p} - \vec{ON}\|$ we get that

$$\vec{NW} = \frac{\mathbf{p} - (\mathbf{p}\cdot\hat{n})\hat{n}}{\|\mathbf{p} - (\mathbf{p}\cdot\hat{n})\hat{n}\|} NQ\cos\mu =$$

$$= (\mathbf{p} - (\mathbf{p}\cdot\hat{n})\hat{n})\cos\mu.$$

The vector $\vec{WQ}$ is perpendicular to both $\mathbf{p}$ and $\hat{n}$ and has length $NQ\sin\mu$. Noting that $NQ = \|\mathbf{p}\|\sin\phi$ we get

$$\vec{WQ} = \frac{\mathbf{p}\times\hat{n}}{\|\mathbf{p}\|\sin\phi} NQ\sin\mu =$$

$$= -\hat{n}\times\mathbf{p}\sin\mu.$$

Therefore Eq. (2.2) becomes

$$\mathbf{q} = (1-\cos\mu)(\mathbf{p}\cdot\hat{n})\hat{n} + \cos\mu\mathbf{p} - \sin\mu(\hat{n}\times\mathbf{p}), \tag{2.3}$$

which is called the rotation formula. As an example of the application of Eq. (2.3)

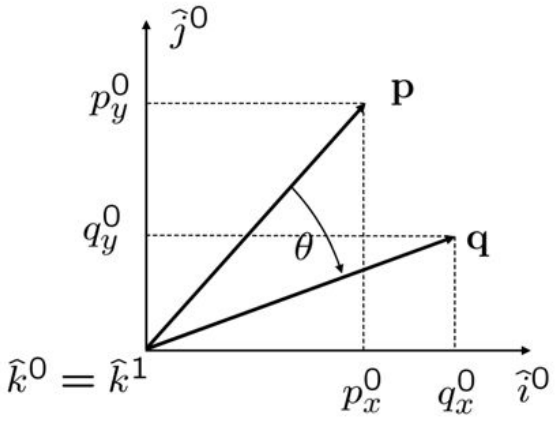

Figure 2.3: Rotation of **p** about the z-axis

consider a left handed rotation of a vector $\mathbf{p}^0$ in frame $\mathcal{F}^0$ about the z-axis as shown in Fig. 2.3. Using the rotation formula we get

$$\mathbf{q}^0 = (1-\cos\theta)(\mathbf{p}\cdot\hat{n})\hat{n} + \cos\phi\mathbf{p} - \sin\phi\hat{n}\times\mathbf{p} =$$
$$= (1-\cos\phi)p_z^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \cos\phi \begin{pmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{pmatrix} - \sin\phi \begin{pmatrix} -p_y^0 \\ p_x^0 \\ 0 \end{pmatrix} =$$
$$= \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{p}^0 =$$
$$= R_0^1\mathbf{p}^0.$$

Note that the rotation matrix R_0^1 can be interpreted in two different ways. The first interpretation is that it transforms the fixed vector **p** from an expression in frame $\mathcal{F}^0$ to an expression in frame $\mathcal{F}^1$ where $\mathcal{F}^1$ has been obtained from $\mathcal{F}^0$ by a right-handed rotation. The second interpretation is that it rotates a vector **p** though a left-handed rotation to a new vector **q** in the same reference frame.
Right-handed rotations of vectors are obtained by using $(R_0^1)^T$.

2.2 Multirotor Coordinate Frames

For multirotors there are several coordinate systems that are of interest. In this section we will define and describe the following coordinate frames:

- the inertial frame,
- the vehicle frame,
- the vehicle-1 frame,
- the vehicle-2 frame,
- and the body frame.

Throughout the thesis we assume a flat, non-rotating earth: a valid assumption for multirotors.

2.2.1 The inertial frame $\mathcal{F}^i$

The inertial coordinate system is an earth fixed coordinate system with origin at the defined home location. As shown in Fig. 2.4, the unit vector $\hat{i}^i$ is directed North, $\hat{j}^i$ is directed East and $\hat{k}^i$ is directed toward the center of the earth.

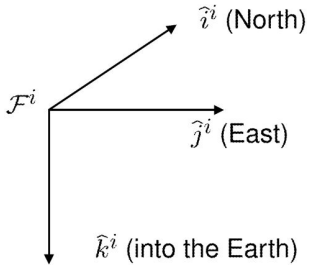

Figure 2.4: The inertial coordinate frame. The x-axis points North, the y-axis points East and the z-axis points into the earth

2.2.2 The vehicle frame $\mathcal{F}^v$

The origin of the vehicle frame is at the center of mass of the multirotor. However, the axes of $\mathcal{F}^v$ are aligned with the axis of the inertial frame $\mathcal{F}^i$. In other words, the unit vector $\hat{i}^v$ points North, $\hat{j}^v$ points East and $\hat{k}^v$ points toward the center of the earth, as shown in Fig. 2.5.

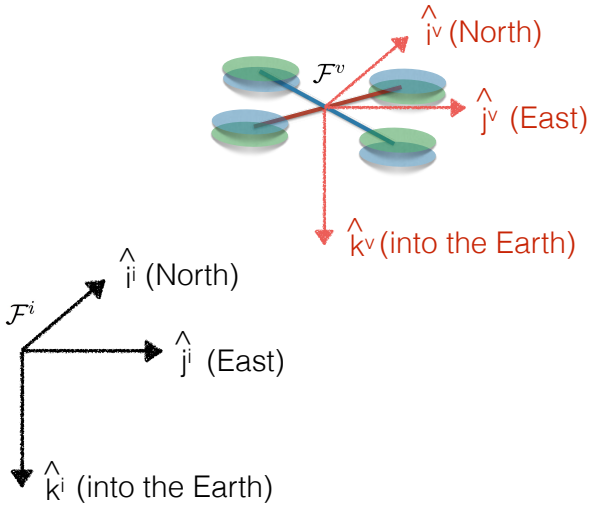

Figure 2.5: The vehicle coordinate frame. The x-axis points North, the y-axis points East and the z-axis points into the earth

2.2.3 The vehicle-1 frame $\mathcal{F}^{v1}$

The origin of the vehicle-1 frame is identical to the vehicle frame, i.e, the center of gravity. However, $\mathcal{F}^{v1}$ is positively rotated about $\hat{k}^v$ by the yaw angle ψ so that if the airframe is not rolling or pitching, then $\hat{i}^{v1}$ would point out the nose of the airframe, $\hat{j}^{v1}$ points out the right, $\hat{k}^{v1}$ is aligned with $\hat{k}^v$ and points into the earth. The vehicle-1 frame is shown in Fig. 2.6. The transformation from $\mathcal{F}^v$ to $\mathcal{F}^{v1}$ is given by

$$\mathbf{p}^{v1} = R_v^{v1}(\psi)\mathbf{p}^v,$$

where

$$R_v^{v1}(\psi) = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

2.2.4 The vehicle-2 frame $\mathcal{F}^{v2}$

The origin of the vehicle-2 frame is again the center of gravity and is obtained by rotating the vehicle-1 frame in a right-handed rotation about the $\hat{j}^{v1}$ axis by the pitch angle θ. If the roll angle is zero, then $\hat{i}^{v2}$ points out the nose of the airframe, $\hat{j}^{v2}$ points out the right and $\hat{k}^{v2}$ points out the belly, as shown in Fig. 2.7. The transformation

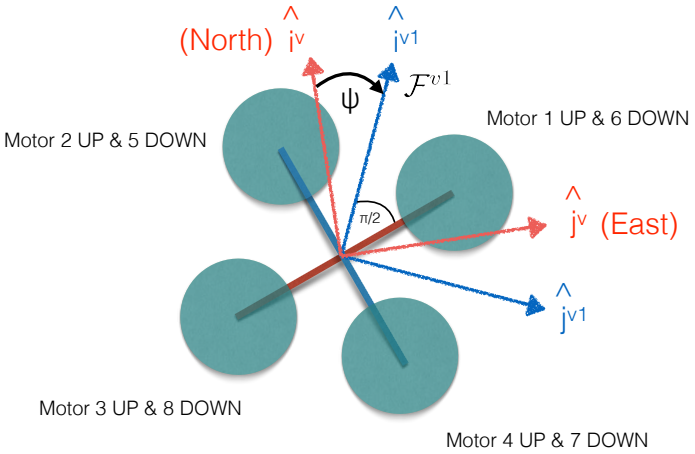

Figure 2.6: The vehicle-1 frame. If the roll and pitch angles are zero, then the x-axis points out the nose of the airframe, the y-axis points out to the right and the z-axis points into the earth

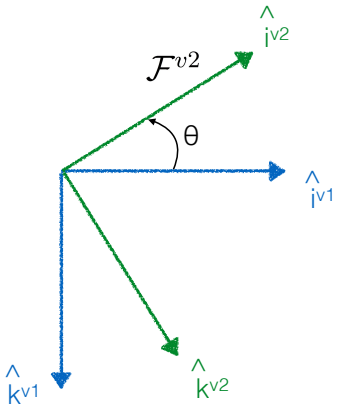

Figure 2.7: The vehicle-2 frame. If the roll angle is zero, then the x axis points out the nose of the airframe, the y-axis points out the right and the z axis points out the belly

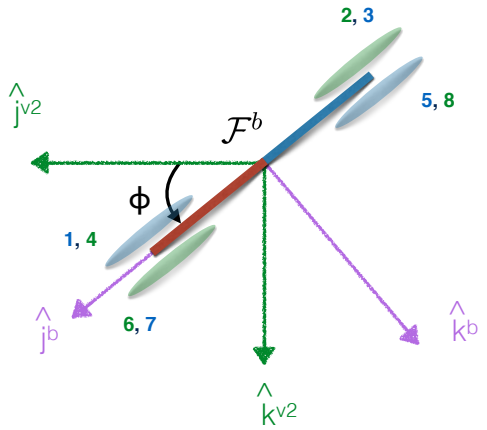

Figure 2.8: The body frame. The x-axis points out the nose of the airframe, the y-axis points out the right and the z-axis points out the belly

from $\mathcal{F}^{v1}$ to $\mathcal{F}^{v2}$ is given by

$$\mathbf{p}^{v2} = R_{v1}^{v2}(\theta)\mathbf{p}^{v1},$$

where

$$R_{v1}^{v2}(\theta) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}.$$

2.2.5 The body frame $\mathcal{F}^b$

The body frame is obtained by rotating the vehicle-2 frame in a right handed rotation about $\hat{i}^{v2}$ by the roll angle ϕ. Therefore, the origin is the center of gravity, $\hat{i}^b$ points out the nose of the airframe, $\hat{j}^b$ points out the right and $\hat{k}^b$ points out the belly. The body frame is shown in Fig. 2.8. The transformation from $\mathcal{F}^{v2}$ to $\mathcal{F}^b$ is given by

$$\mathbf{p}^b = R_{v2}^b(\phi)\mathbf{p}^{v2},$$

where

$$R_{v2}^b(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}.$$

The transformation from the vehicle frame to the body frame is given by
$R_v^b(\phi,\ \theta,\ \psi) = R_{v2}^b(\phi)R_{v1}^{v2}(\theta)R_v^{v1}(\psi) =$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\phi s\theta c\psi - c\phi s\psi & s\phi s\theta s\psi + c\phi c\psi & s\phi c\theta \\ c\phi s\theta c\psi + s\phi s\psi & c\phi s\theta s\psi - s\phi c\psi & c\phi c\theta \end{pmatrix},$$

where $c\phi \triangleq \cos\phi$ and $s\phi \triangleq \sin\phi$.

2.3 Equation of Coriolis

In this section we provide a simple derivation of the famous equation of Coriolis. We will again follow the derivation given in [36]. Suppose that we are given two coordinate frames $\mathcal{F}^i$ and $\mathcal{F}^b$ as shown in Fig. 2.9. For example, $\mathcal{F}^i$ might represent the inertial frame and $\mathcal{F}^b$ might represent the body frame of a multirotor. Suppose that the vector $\mathbf{p}$ is moving in $\mathcal{F}^b$ and that $\mathcal{F}^b$ is rotating and translating with respect to $\mathcal{F}^i$. Our objective is to find the time derivative of $\mathbf{p}$ as seen from frame $\mathcal{F}^i$. We will derive the appropriate equation through two steps. Assume first that $\mathcal{F}^b$ is not rotating with respect to $\mathcal{F}^i$. Denoting the time derivative of $\mathbf{p}$ in frame $\mathcal{F}^i$ as $\frac{d}{dt_i}\mathbf{p}$ we get

$$\frac{d}{dt_i}\mathbf{p} = \frac{d}{dt_b}\mathbf{p}. \tag{2.4}$$

On the other hand, assume that $\mathbf{p}$ is fixed in $\mathcal{F}^b$ but that $\mathcal{F}^b$ is rotating with respect to $\mathcal{F}^i$, let $\hat{s}$ be the instantaneous axis of rotation and $\delta\phi$ the (right-handed) rotation angle. Then the rotation formula, Eq. (2.3), gives

$$\mathbf{p} + \delta\mathbf{p} = (1 - \cos(-\delta\phi))(\mathbf{p}\cdot\hat{s})\hat{s} + \cos(-\delta\phi)\mathbf{p} - \sin(-\delta\phi)\left(\hat{s}\times\mathbf{p}\right).$$

Using a small angle approximation and dividing both sides by δt gives

$$\frac{\delta\mathbf{p}}{\delta t} \approx \frac{\delta\phi}{\delta t}\left(\hat{s}\times\mathbf{p}\right).$$

Taking the limit as $\delta t \to 0$ and defining the angular velocity of $\mathcal{F}^b$ with respect to $\mathcal{F}^i$ as $\boldsymbol{\omega}_{b/i} \triangleq \hat{s}\dot{\phi}$ we get

$$\frac{d}{dt_i}\mathbf{p} = \boldsymbol{\omega}_{b/i}\times\mathbf{p}. \tag{2.5}$$

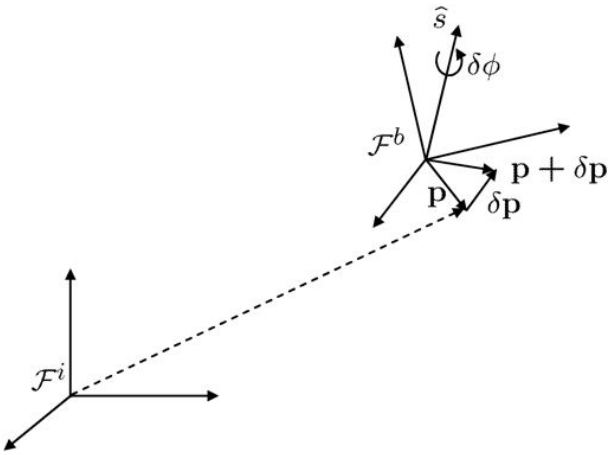

Figure 2.9: Derivation of the equation of Coriolis

Since differentiation is a linear operator we can combine Eq. (2.4) and Eq. (2.5) to obtain

$$\frac{d}{dt_i}\mathbf{p} = \frac{d}{dt_b}\mathbf{p} + \boldsymbol{\omega}_{b/i} \times \mathbf{p}, \tag{2.6}$$

which is the equation of Coriolis.

3

Kinematics and Dynamics

In this chapter we derive the expressions for the kinematics and the dynamics of a rigid body following the approach given in [37]. While the expressions derived in this chapter are general to any rigid body, we will use notation and coordinate frames that are typical in the aeronautics literature. In particular, in section 3.1 we define the notation that will be used for the state variables of a multirotor. In section 3.2 we derive the expressions for the kinematics and in section 3.3 we derive the dynamics.

3.1 Multirotor State Variables

The state variables of the multirotor are the following twelve quantities:

p_n = the inertial (north) position of the multirotor along $\hat{i}^i$ in $\mathcal{F}^i$,

p_e = the inertial (east) position of the multirotor along $\hat{j}^i$ in $\mathcal{F}^i$,

h = the altitude of the aircraft measured along $\hat{k}^i$ in $\mathcal{F}^i$,

u = the body frame velocity measured along $\hat{i}^b$ in $\mathcal{F}^b$,

v = the body frame velocity measured along $\hat{j}^b$ in $\mathcal{F}^b$,

w = the body frame velocity measured along $\hat{k}^b$ in $\mathcal{F}^b$,

ϕ = the roll angle defined with respect to $\mathcal{F}^{v2}$,

θ = the pitch angle defined with respect to $\mathcal{F}^{v1}$,

ψ = the yaw angle defined with respect to $\mathcal{F}^{v}$,

p = the roll rate measured along $\hat{i}^b$ in $\mathcal{F}^b$,

q = the pitch rate measured along $\hat{j}^b$ in $\mathcal{F}^b$,

r = the yaw rate measured along $\hat{k}^b$ in $\mathcal{F}^b$.

The state variables are shown schematically in Fig. 3.1. The position (p_n, p_e, h) of the multirotor is given in the inertial frame, with positive h defined along the negative Z axis in the inertial frame. The translational velocity $(u,\ v,\ w)$ and the angular velocity (p, q, r) of the multirotor are given with respect to the body frame. The Euler angles (roll ϕ, pitch θ and yaw ψ) are given with respect to the vehicle-2 frame, the vehicle-1 frame and the vehicle frame respectively.

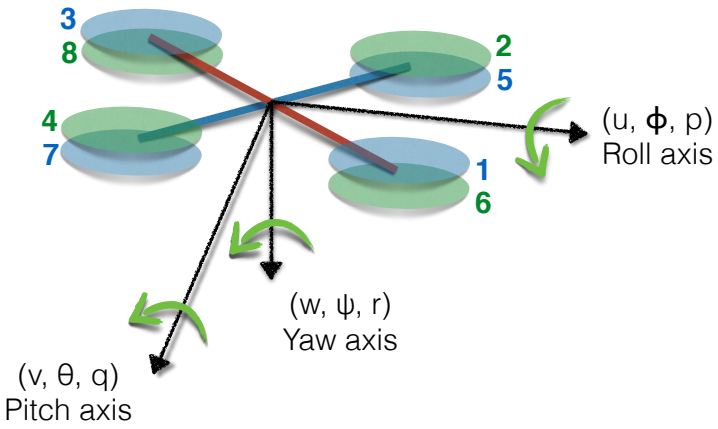

Figure 3.1: Definition of axes

3.2 Multirotor Kinematics

The state variables $p_n,\ p_e,$ and $-h$ are inertial frame quantities, whereas the velocities $u,\ v$ and w are body frame quantities. Therefore the relationship between position and

velocities is given by

$$\frac{d}{dt}\begin{pmatrix} p_n \\ p_e \\ -h \end{pmatrix} = R_b^v \begin{pmatrix} u \\ v \\ w \end{pmatrix} =$$

$$= (R_v^b)^T \begin{pmatrix} u \\ v \\ w \end{pmatrix} =$$

$$= \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix},$$

where $c\phi \triangleq \cos\phi$ and $s\phi \triangleq \sin\phi$.

The relationship between absolute angles $\phi,\ \theta$ and ψ and the angular rates $p,\ q$ and r is also complicated by the fact that these quantities are defined in different coordinate frames. The angular rates are defined in the body frame $\mathcal{F}^b$, whereas the roll angle ϕ is defined in $\mathcal{F}^{v2}$, the pitch angle θ is defined in $\mathcal{F}^{v1}$ and the yaw angle ψ is defined in the vehicle frame $\mathcal{F}^v$.

We need to relate p, q and r to $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$. Since $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ are small and noting that

$$R_{v2}^b(\dot{\phi}) = R_{v1}^{v2}(\dot{\theta}) = R_v^{v1}(\dot{\psi}) = I,$$

we get

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = R_{v2}^b(\dot{\phi}) \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + R_{v2}^b(\phi) R_{v1}^{v2}(\dot{\theta}) \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + R_{v2}^b(\phi) R_{v1}^{v2}(\theta) R_v^{v1}(\dot{\psi}) \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} =$$

$$= \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{pmatrix} \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{pmatrix} \begin{pmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}, \tag{3.1}$$

where $c\phi \triangleq \cos\phi$ and $s\phi \triangleq \sin\phi$.

Inverting we get

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \tag{3.2}$$

3.3 Rigid Body Dynamics

Let $\mathbf{v}$ be the velocity vector of the multirotor. Newton's laws only hold in inertial frames, therefore Newton's law applied to the translational motion is

$$m\frac{d\mathbf{v}}{dt_i} = \mathbf{f},$$

where m is the mass of the multirotor considered constant, $\mathbf{f}$ is the total force applied to the multirotor and $\dfrac{d}{dt_i}$ is the time derivative in the inertial frame. From the equation of Coriolis we have

$$m\frac{d\mathbf{v}}{dt_i} = m\left(\frac{d\mathbf{v}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{v}\right) = \mathbf{f}, \tag{3.3}$$

where $\boldsymbol{\omega}_{b/i}$ is the angular velocity of the airframe with respect to the inertial frame. Since the control force is computed and applied in the body coordinate system and since $\boldsymbol{\omega}_{b/i}$ is measured in body coordinates, we will translate Eq. (3.3) in body coordinates obtaining

$$m\left(\frac{d\mathbf{v}^b}{dt_b} + \boldsymbol{\omega}^b_{b/i} \times \mathbf{v}^b\right) = \mathbf{f}^b, \tag{3.4}$$

where $\mathbf{v}^b \triangleq (u,\ v,\ w)^T$, $\boldsymbol{\omega}^b_{b/i} \triangleq (p,\ q,\ r)^T$ and $\mathbf{f}^b \triangleq (f_x,\ f_y,\ f_z)^T$ are all expressed in the body reference frame. Therefore Eq. (3.4) becomes

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m}\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}. \tag{3.5}$$

For rotational motion, Newton's second law states that

$$\frac{d\mathbf{h}}{dt_i} = \mathbf{m},$$

where $\mathbf{h}$ is the angular momentum and $\mathbf{m}$ is the applied torque. Using the equation of Coriolis we have

$$\frac{d\mathbf{h}}{dt_i} = \frac{d\mathbf{h}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{h} = \mathbf{m}. \tag{3.6}$$

Again, Eq. (3.6) is most easily resolved in body coordinates giving

$$\frac{d\mathbf{h}^b}{dt_b} + \boldsymbol{\omega}^b_{b/i} \times \mathbf{h}^b = \mathbf{m}^b, \tag{3.7}$$

where $\mathbf{h}^b = \mathbf{J}\boldsymbol{\omega}^b_{b/i}$ and $\mathbf{J}$ is the constant inertia matrix given by

$$\mathbf{J} = \begin{pmatrix} \int (y^2+z^2)\,dm & -\int xy\,dm & -\int xz\,dm \\ -\int xy\,dm & \int (x^2+z^2)\,dm & -\int yz\,dm \\ -\int xz\,dm & -\int yz\,dm & \int (x^2+y^2)\,dm \end{pmatrix} \triangleq$$

$$\triangleq \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{pmatrix}.$$

As shown in Fig. 1.19, the multirotor is essentially symmetric about all three axes, therefore $J_{xy} = J_{xz} = J_{yz} = 0$, which implies that

$$\mathbf{J} = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix}.$$

Therefore

$$\mathbf{J}^{-1} = \begin{pmatrix} \frac{1}{J_x} & 0 & 0 \\ 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & \frac{1}{J_z} \end{pmatrix}.$$

Defining $\mathbf{m}^b \triangleq (\tau_\phi,\ \tau_\theta,\ \tau_\psi)^T$ we can write Eq. (3.7) as

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} =$$

$$= \begin{pmatrix} \frac{1}{J_x} & 0 & 0 \\ 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & \frac{1}{J_z} \end{pmatrix} \left[\begin{pmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{J_x} & 0 & 0 \\ 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & \frac{1}{J_z} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} + \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} \right] =$$

$$= \begin{pmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{pmatrix} + \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{pmatrix}.$$

The six degree of freedom model for the multirotor kinematics and dynamics can be summarized as follows:

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{3.8}$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} \tag{3.9}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & s\phi \tan\theta & c\phi \tan\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{3.10}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{pmatrix} + \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{pmatrix}, \tag{3.11}$$

where $c\phi \triangleq \cos\phi$ and $s\phi \triangleq \sin\phi$.

4

Physical architecture

4.1 Physical architecture and parameters

The primary structure of the prototype is constituted by the central fiberglass housing which accommodates the control electronics, the power distribution board, the flight sensors and the power supply. Four arms, equally spaced, depart from the primary structure, each one sustaining, at its extremity, two brushless coaxial, counterrotating DC motors and the relevant electronic speed controllers (ESCs).

The body fixed reference frame ($\mathcal{F}^b$) of the multirotor is set, as shown in Fig. 1.19, in its centre of gravity (COG), with:

- the X_b axis along the front direction;
- the Y_b axis in the right direction;
- the Z_b axis resulting downwards.

The used control electronics, available on the market, is named Ardupilot Mega board (A.P.M.); the board can be coupled with an Inertial Measurement Unit (IMU) platform, namely OilPan IMU, that contains accelerometers and gyroscopes from whose measurements we derive the attitude and velocity of the multirotor. The chosen board allows linking other useful devices like a sonar for height measurements. The A.P.M. is supplied by a lithium polymer battery and a dedicated board distributes power to sensors and rotors. Communication with a ground control station is possible through

the use of XBee radios. For a more thorough description of the used electronics refer to [38].

Qx-Rotor is composed of several major (e.g. motors, batteries, etc...) and minor components (e.g. bolt, screws, etc...); in order to reduce the mass analysis complexity, all the single masses have been concentrated in some major units, as presented in Tab. 4.1. The mass of the whole multirotor results accordingly to be equal to $m = 2.645\,[Kg]$. With the aim of simplifying the identification process and seeking a good approximation

Table 4.1: Main components masses

Component	Quantity	Mass value [Kg]	Symbol
Motor	8	0.071	m_m
Propeller	8	0.020	m_p
Battery	2	0.432	m_b
Central structure	1	0.937	m_{cs}
Arm	4	0.029	m_a
Totale mass	//	2.645	m

of the parameters, the multirotor inertial structure is regarded as two perpendicular rods, with four concentrated masses, at the four ends. Each one of these four concentred masses, m_c, is given by Eq. (4.1).

$$m_c = 2m_m + 2m_p + 1/2m_a = 0.197\,[Kg] \tag{4.1}$$

Recalling that the $\mathcal{F}^b$ is set as in Fig. 1.19, the distance of these four concentrated masses from the Z_b axis is equal to $\lambda = 0.220\,[m]$, while the distances from the X_b and Y_b axes are both equal to $\lambda_c = \lambda cos(\pi/4) = 0.156\,[m]$.

The aircraft mass balance (i.e. the central structure, the batteries and the half part of each arm) is assumed to be homogeneously distributed inside a sphere of radius $\rho = 0.070\,[m]$, centred at the origin of the body axes. The mass of this sphere, m_s, is shown in Eq. (4.2).

$$m_s = m_{cs} + 2m_b + 1/2 \cdot 4 \cdot m_a = 1.859\,[Kg] \tag{4.2}$$

Hence we can calculate the moments of inertia with respect to the body axes as in Eqs. (4.3) and (4.4).

$$J_x = J_y = 4m_c{\lambda_c}^2 + \frac{2}{5}m_s\rho^2 = 0.023[Kg\ m^2] \tag{4.3}$$

$$J_z = 4m_c\lambda^2 + \frac{2}{5}m_s\rho^2 = 0.042[Kg\ m^2] \tag{4.4}$$

4.2 The Ardupilot Mega board

The Ardupilot Mega board (A.P.M.) is a printed circuit board (P.C.B.) provided with an embedded processor and combined with circuitry to switch between the radio control (R.C.) and the autopilot control. The A.P.M. is shown in Fig. 4.1.

It is based on a 16 MHz Atmega 2560 processor and provided with a built-in hardware

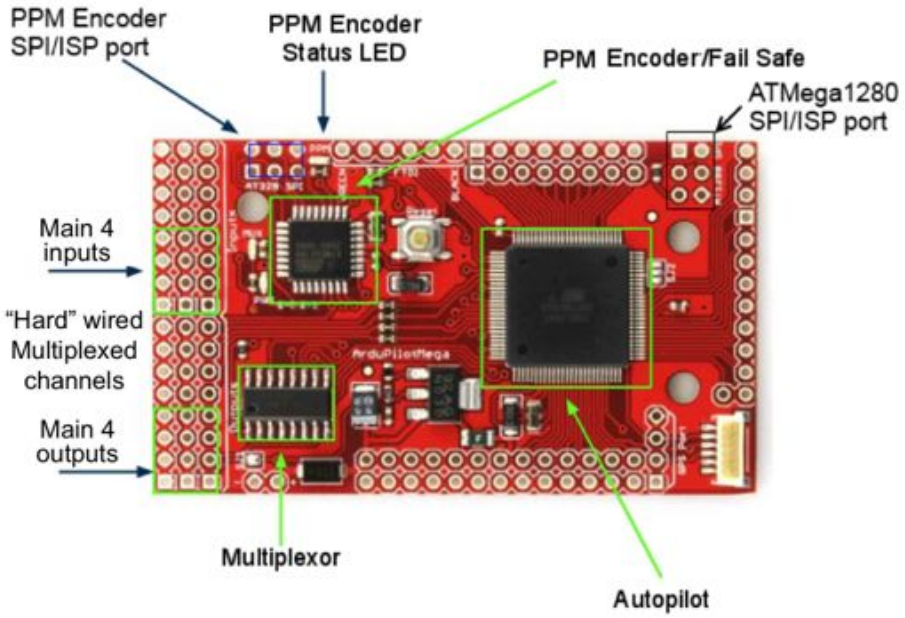

Figure 4.1: The Ardupilot Mega board

failsafe that uses a separate circuit (multiplexer chip and Atmega 328 processor) to transfer control from the R.C. system to the autopilot and back again.

The A.P.M. dual processor design allows 32 million instructions per second (M.I.P.S.) and supports up to 700 waypoints memorization, thanks to the 4 Kb EEPROM. Other memory embedded devices are a 128 Kb Flash Program Memory and a 8 Kb SRAM. The board is equipped with a 6-pin G.P.S. connector, 16 spare analog inputs - each one provided with an analog to digital converter (A.D.C.) - and 40 digital inputs/outputs for additional sensors. Four dedicated serial ports for two-way telemetry (using XBee modules) are available. Finally 8 R.C. channels, including the autopilot on/off one, can

be processed by the autopilot. Sensors are mounted on a different P.C.B., the Inertial Measurement Unit (I.M.U.) board, commercially known as OilPan I.M.U., shown in Fig. 4.2. This P.C.B. is equipped with

- a dual 3.3 volts voltage regulator,
- 3-axis accelerometers,
- 3-axis gyros,
- 12-bit A.D.C. for gyros,
- a built-in 16 MB data logger (*"the black box"*),
- a temperature sensor,
- a barometric pressure sensor for altitude sensing,
- a voltage sensor for battery status.

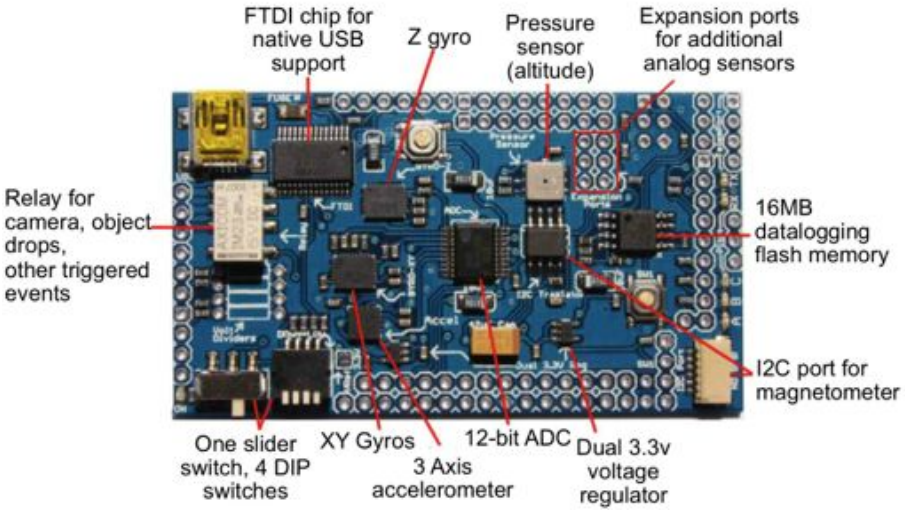

Figure 4.2: The OilPan Inertial Measurement Unit

4.3 Competitiveness aspects

Two are the main competitiveness aspects linked to the described autopilot system. The first one is the overall price of about 400 $ comprising the A.P.M., the OilPan

I.M.U., the sensors and the telemetry modules. This price is far away from the other C.O.T.S. platforms.
The main factor is, of course, the open source nature of this autopilot system, since it is made of both open source software and hardware. This means that the hardware schematics, P.C.B. files, parts list are all freely available on the web, published under a Creative Commons license that allows free use and modifications as long as the resulting product retains the producer credit.
Also the software is open source, published under a Lesser General Public License (L.G.P.L.) that allows free use and modification, as long as also the resulting product is open source and the producer attribution is retained.

4.4 Endurance

One of the main issue with multirotors is the endurance. Multirotors are generally supplied by lithium polymer (LiPo) batteries. A LiPo battery can consist of multiple cells. One cell delivers approximatively 3.7 Volts [V]. A three cells LiPo battery is called a 3S LiPo and can supply 11.1 [V]. Another important parameter to know when dealing with LiPo batteries is the discharge rate identified with the letter "C". For example a discharge rate of "25C to 40C" means that 25C is the nominal discharge rate and 40C is the maximum burst discharge rate. It is advisable to stay on or below nominal discharge level to preserve battery future life. Not all brands say something about the peak discharge rate on the battery itself. The battery capacity is defined in milli-Ampere per hour [mAh]. A battery with a 1000 [mAh] capacity can deliver 1000 milli-Ampere [mA] for 1 hour as well as 1 [mA] for 1000 hours and so on so far. The battery capacity, together with the LiPo battery discharge rate, will define its maximum current output (Ampére, [A]). This fact is very important to keep in mind, when choosing a LiPo battery. The combination of capacity and discharge rate is what we have to focus on. It is important to know that the battery cells do not count up for maximum current draw. The amount of cells only determine the voltage of the LiPo, as needed for the equipment.

4.4.1 Calculation of discharge rate

For the multirotor described in the thesis, two 3S LiPo batteries, in parallel connection, have been used. Each one of them has a capacity of 5000 [mAh] giving a total availability of 10000 [mAh] which is equal to 10 [Ah]. The nominal discharge rate is 30C and the maximum burst discharge rate is 50C. When connected in parallel the whole pack has a discharge rate of 30C as well. Therefore the maximum current draw can be calculated as

$$C \cdot dr_n = 10[Ah] \cdot 30C = 300[A] \tag{4.5}$$

where

- C is the battery capacity,
- dr_n is the nominal discharge rate.

So, a 10000[mAh]/30C LiPo battery can only handle a maximum current draw of 300 [A]. This clarifies why the combination capacity/discharge rate is of such importance for selecting the right LiPo for the own project.

4.4.2 Calculation of maximum flying time

During this research it has been decided to evaluate the *Qx-Rotor* endurance characteristics for several payload masses.
Two *flight tests techniques* have been considered. The first one could have been a *brute force* approach, consisting in measuring the flight times for several payload configurations. The second one, more elegant technique, consisted in designing opportune manoeuvres to highlight the endurance for several *virtual* payload masses although keeping the real payload constant. The second approach has been followed and the chosen manoeuvres were a simple sequence of vertical accelerations. As shown in Fig. 4.3 during this sequence of manoeuvres the vertical position, the current draw and the voltage were measured. These graphs allow us to observe that the average current draw to keep the *Qx-Rotor* in hovering is approximatively 50 [A]. Moreover it can be observed that obviously the current peaks correspond, with a small delay, to the altitude peaks reached by the multirotor. The second time derivative of the altitude position is the vertical acceleration. The vertical acceleration multiplied by the *Qx-Rotor* mass, equal

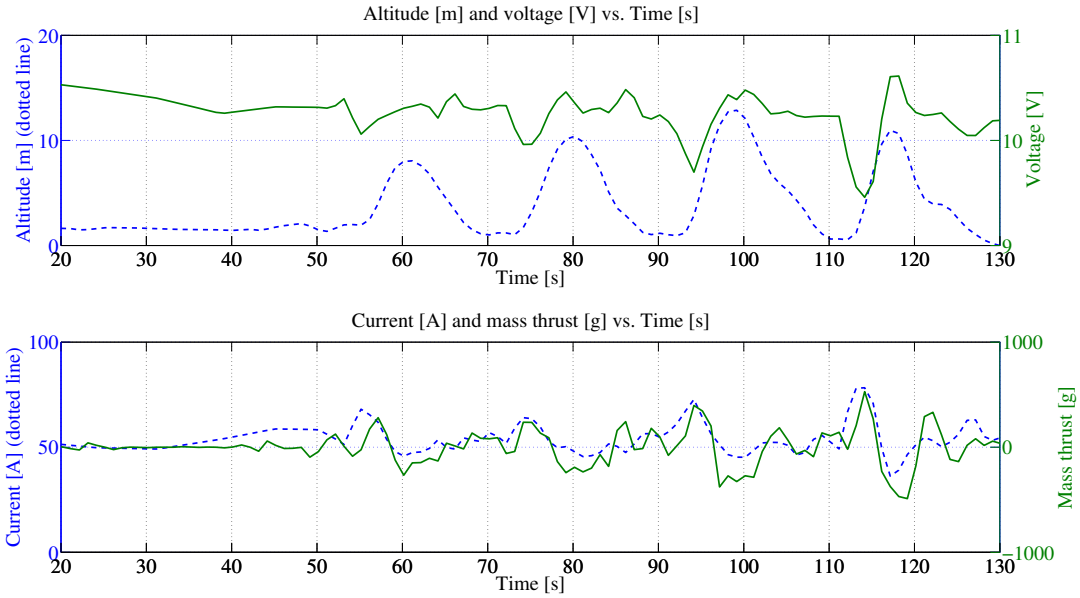

Figure 4.3: Parameters measured during the sequence of vertical accelerations

to 2.645 $[Kg]$, gives the *excess or the lack* of thrust with respect to the hovering needed thrust which allows the multirotor to climb or descend. This thrust is a force but in order to give a better perception of its magnitude it has been preferred to calculate the relevant mass. Therefore the *exceeding or lacking thrust* has been divided by the gravity acceleration obtaining the values shown in the second graph of Fig. 4.3, graphically aligned with the relevant drawn current values. These graphs are truly significative since they give the indication of the current draws relevant to the peaks values of the *exceeding thrust*. By analysing numerically the relation between these peaks we observe that there exist an average proportionality factor, $\bar{p}_f$, between the current draw and the *exceeding thrust*. This value, derived experimentally and valid only in the *Qx-Rotor* case, is presented in Eq. (4.6),

$$\bar{p}_f \triangleq \frac{I_d}{T_{e,m}} = 0.039 \frac{[A]}{[g]} \tag{4.6}$$

where

- I_d is the drawn current expressed in Ampére,
- $T_{e,m}$ is the *exceeding thrust* expressed in terms of mass grams,

- $\bar{p}_f$ is the average proportionality factor.

This value allows to express the *Qx-Rotor* endurance as a function of the payload mass as in Eq. (4.7),

$$E(p_g) = \frac{C}{I_d|_{hov} + p_g \cdot \bar{p}_f} \cdot 60 \tag{4.7}$$

where

- p_g is the payload mass, expressed in grams, that is the function independent variable,
- C is the battery capacity, expressed in Ampere per hour [Ah], equal to 10 [Ah] in the *Qx-Rotor* case,
- $I_d|_{hov}$ is the current drawn while the multirotor is in hovering condition, expressed in Ampére, equal to 50 [A] in the *Qx-Rotor* case,
- $\bar{p}_f$ is, again, the average proportionality factor, equal to 0.039 in the *Qx-Rotor* case,
- E is the endurance, expressed in minutes [min], that is the function dependent variable.

Equation (4.7) is graphically translated into Fig. 4.4 considering the whole range of possible payload masses, until the maximum value of 800 [g], evaluated experimentally. During the flight tests a *safety endurance* value, equal to the 80% of the total endurance, has been considered. This is extremely advisable due to the high risk of damage in case of impact and since it is not recommended to completely drain the LiPo batteries. The curve of the *safety endurance* is also illustrated in Fig. 4.4.

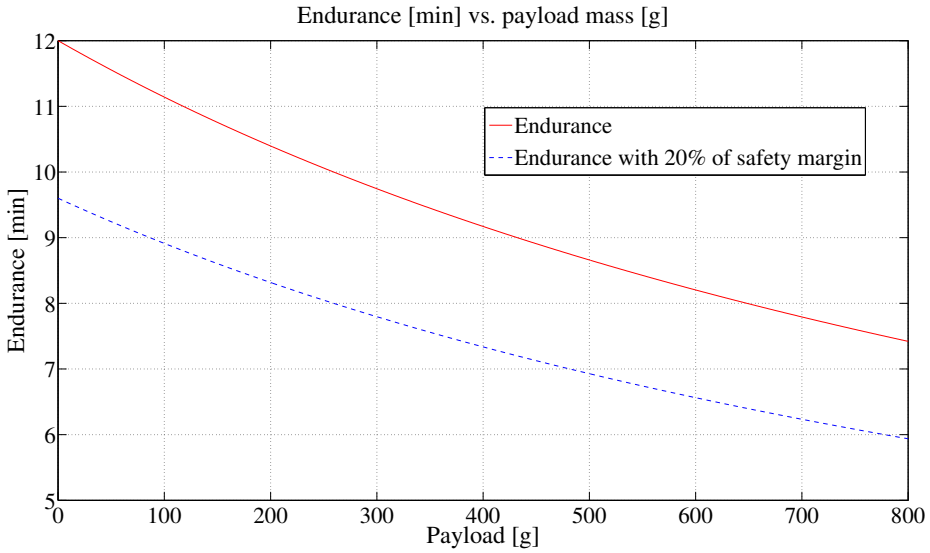

Figure 4.4: Multirotor endurance as a function of the payload mass

4.5 Coaxial rotors

One advantage of the contrarotating coaxial rotors design is that the net size of the rotors is reduced (for a given multirotor gross weight) because each rotor now provides vertical thrust. However, the two rotors and their wakes interact with one another, producing a somewhat more complicated flow field than is found with a single rotor and this interacting flow incurs a loss of net rotor system aerodynamic efficiency. In [39] Coleman gives a good summary of coaxial helicopter rotors and a comprehensive list of relevant citations on performance, wake characteristics and method of analysis. Following [40], consider a simple momentum analysis of the hovering coaxial rotor problem. Assume that the rotor planes are sufficiently close together and that each rotor provides an equal fraction of the total system thrust, $2T$, where $T = \frac{W}{2}$. The effective induced velocity of the rotor system will be

$$(v_i)_e = \sqrt{\frac{2T}{2\rho A}}. \tag{4.8}$$

Therefore, the induced power is

$$(P_i)_{tot} = 2T(v_i)_e = \frac{(2T)^{\frac{3}{2}}}{\sqrt{2\rho A}}. \tag{4.9}$$

However, if we treat each rotor separately then the induced power for either rotor will be Tv_i and for the two separate rotors

$$P_i = \frac{2T^{\frac{3}{2}}}{\sqrt{2\rho A}}. \tag{4.10}$$

If the interference-induced power factor κ_{int} is considered to be the ratio of Eqs. (4.9) and (4.10) then

$$\kappa_{int} = \frac{(P_i)_{tot}}{P_i} = \left(\frac{(2T)^{\frac{3}{2}}}{\sqrt{2\rho A}}\right)\left(\frac{2T^{\frac{3}{2}}}{\sqrt{2\rho A}}\right)^{-1} = \sqrt{2}, \tag{4.11}$$

which is a 41% increase in induced power relative to the power required to operate the two rotors in complete isolation. This simple momentum analysis of the problem has been shown to be overlay pessimistic when compared with experimental measurements for closely spaced coaxial rotors as reported in [39, 41]. The main reason for the over prediction of induced power is related to the actual (finite) spacing between the two rotors. Generally, on coaxial designs the rotors are spaced sufficiently far apart that the lower rotor operates in the vena contract of the upper rotor. This is justified from the flow visualisation result of Taylor reported in [42]. Based on ideal flow considerations this means that only half of the area of the lower rotor operates in an effective climb velocity induced by the upper rotor. This problem can be tackled by means of the simple momentum theory and the application of the mass, momentum and energy conservation equation in the integral form. We will assume that the performance of the upper rotor is not influenced by the lower rotor. The induced velocity at the upper rotor is

$$v_u = \sqrt{\frac{T}{2\rho A}} = v_h, \tag{4.12}$$

where A is the disk area and T is the thrust on the upper rotor. The vena contract of the upper rotor is an area of $\frac{A}{2}$ with velocity $2v_u$. Therefore, at the plane of the lower rotor there is a velocity of $2v_u + v_l$ over the inner one-half of the disk area as shown in Fig. 4.5. Over the outer one-half of the disk area, the induced velocity is v_l. Assume that the velocity in the fully developed slipstream of the lower rotor (plane 3 in Fig. 4.5) is uniform with velocity w_l. The mass flow through the upper rotor is $(\rho A v_u)2v_u = 2\rho A v_u^2$. This is the momentum of the fluid into the lower rotor. The

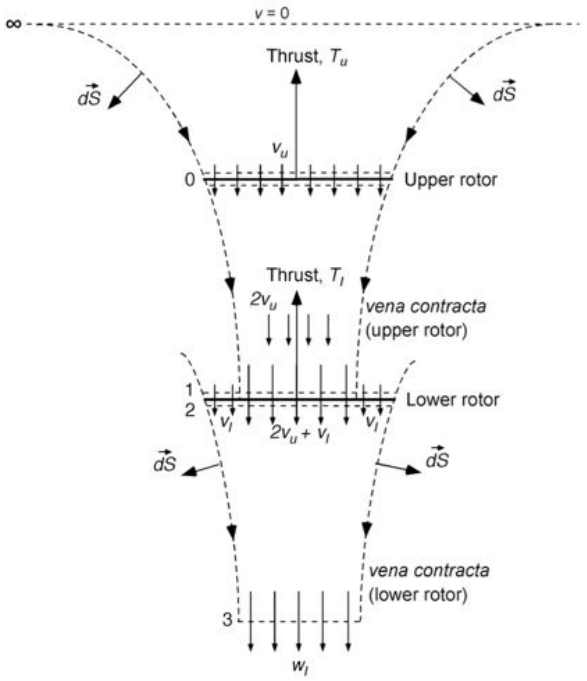

Figure 4.5: Flow model for a coaxial rotor analysis where the lower rotor is considered to operate in the fully developed slipstream of the upper rotor [2]

mass flow rates over the inner and outer parts of the lower rotor are $\rho(\frac{A}{2})(2v_u + v_l)$ and $\rho(\frac{A}{2})v_l$, respectively. Therefore,

$$\dot{m} = \rho\frac{A}{2}\left(2v_u + v_l\right) + \rho\left(\frac{A}{2}\right)v_l = \rho A\left(v_u + v_l\right). \tag{4.13}$$

The momentum flow out of plane 3 is $\dot{m}w_l$, so the thrust on the lower rotor is

$$T_l = \rho A(v_u + v_l)w_l - 2\rho A v_u^2. \tag{4.14}$$

The work done by the lower rotor is

$$P_l = T_l(v_u + v_l), \tag{4.15}$$

and this is equal to the gain in kinetic energy of the slipstream. Therefore,

$$T_l(v_u+v_l) = \frac{1}{2}\rho A(v_u+v_l)w_l^2 - \frac{1}{2}\rho\left(\frac{A}{2}\right)(2v_u)(2v_u)^2 = \frac{1}{2}\rho A(v_u+v_l)w_l^2 - 2\rho A v_u^3. \tag{4.16}$$

Assuming $T_l = T_u = T$, then $T = 2\rho A v_u^2$, then from Eq. (4.14) we get

$$T_l = T = \frac{1}{2}\rho A(v_u + v_l)w_l, \tag{4.17}$$

and from Eq. (4.16) we get

$$T(2v_u + v_l) = \frac{1}{2}\rho A(v_u + v_l)w_l^2. \tag{4.18}$$

Using Eqs. (4.17) and (4.18) gives $w_l = 2v_u + v_l$, substituting this into Eq. (4.17) and remembering that $T = 2\rho A v_u^2$ gives

$$4\rho A v_u^2 = \rho A(v_u + v_l)w_l = \rho A(v_u + v_l)(2v_u + v_l). \tag{4.19}$$

Rearranging as a quadratic in terms of v_l and solving gives

$$v_l = \left(\frac{-3 + \sqrt{17}}{2}\right) v_u = 0.5616 v_u. \tag{4.20}$$

The power for the upper rotor is $P_u = Tv_u = Tv_h$ and for the lower rotor $P_l = T(v_u + v_l) = 1.5616 T v_h$. Therefore, for both rotors the total power is $2.5616 T v_h$. This is compared to $2Tv_h$ when the rotors are operating in isolation. This means that the induced power factor from interference, κ_{int}, is given by

$$\kappa_{int} = \frac{(P_i)_{coax}}{(2P_i)_{isolated}} = \frac{2.5616 T v_h}{2Tv_h} = 1.281, \tag{4.21}$$

which is 28.1% increase compared to a 41% increase when the two rotors have no vertical separation. This is closer to the values deduced from experiments for which $\kappa_{int} \approx 1.160$, as reported in [43], but the theory still overpredicts the interference value. When the coaxial is operated at equal rotor torque, it can be shown that the induced power factor is given by

$$\kappa_{int} = \frac{2P_u}{(T_u + T_l)v_u} = 2\sqrt{2}\left(\frac{T_u}{T_l}\right)^{\frac{3}{2}}\left(1 + \frac{T_u}{T_l}\right)^{-\frac{3}{2}} = 1.281, \tag{4.22}$$

which is the same result as for the thrust balanced case when compared to two isolated rotors operated at the same thrust. When compared to two isolated rotors at the thrusts needed for a torque balance, then $\kappa_{int} = 1.266$.

4.6 Classical vs. X8 octorotor configuration

Figures 1.16 and 1.19 show two possible octorotor configurations. The choice among them depends on the specific needs and mission requirements. Two main parameters can be used to compare them:

- endurance,
- minimum possible size.

4.6.1 Endurance comparison

From section 4.5 it is known that two coaxial rotors are less efficient than two isolated identical rotors. This means that, at equal overall produced thrust, the coaxial system absorbs more power with respect to the two isolated rotors. Section 4.5 explained also that the ratio between these values is between $\kappa_{int} \approx 1.160$ (experimental) and $\kappa_{int} = 1.281$ (theoretical). Since the absorbed power is directly proportional to drawn current which, in turn, is inversely proportional to the endurance, we have that the endurance of the classical octorotor system of Fig. 1.16 is κ_{int} times higher than the endurance of the X8 octorotor shown in Fig. 1.19.
Unfortunately this is not exactly true since a classical octorotor requires four more arms compared to an X8 shaped one. Moreover the arms need to be longer to guarantee adequate spacing among the rotors. In our case we have that the arm mass per unit length is $m_{as} = 1.32g/cm$. As illustrated in Fig. 4.6, considering propellers with a diameter equal to $d_p = 25.4cm$ (10 *inches*), the minimum arm length for an X8 octorotor is equal to

$$l_{aX8} = \frac{1}{2} \cdot \sqrt{2} \cdot d_p = 17.96cm, \tag{4.23}$$

which corresponds to a mass of

$$m_{aX8} = m_{as} \cdot l_{aX8} = 23.70g. \tag{4.24}$$

Therefore the total minimum arms mass for the X8 octorotor case is equal to

$$m_{aX8_{tot}} = 4 \cdot m_{aX8} = 94.80g. \tag{4.25}$$

The minimum arm length for a classical octorotor is instead equal to

$$l_{a8} = \sqrt{a_8^2 + \left(\frac{1}{2} \cdot d_p\right)^2} = 33.18cm, \tag{4.26}$$

where a_8 is the apothem of the octagon depicted in Fig. 4.6 that can be calculated as

$$a_8 = \frac{1}{2 \cdot \tan\left(\frac{\pi}{8}\right)} \cdot d_p = 1.207 \cdot d_p = 30.66cm. \tag{4.27}$$

Therefore the minimum arm mass for the classical octorotor case is equal to

$$m_{a8} = m_{as} \cdot l_{a8} = 43.80g \tag{4.28}$$

and the total minimum arms mass for the classical octorotor case is equal to

$$m_{a8_{tot}} = 8 \cdot m_{a8} = 350.40g. \tag{4.29}$$

This calculations guide us to define that there is a minimum difference between the total arms mass of the X8 octorotor with respect to the classical octorotor that is equal to

$$m_{diff_{min}} = m_{a8_{tot}} - m_{aX8_{tot}} = 255.60g \tag{4.30}$$

and the ratio between these two masses is equal to

$$m_{ratio_{min}} = \frac{m_{a8_{tot}}}{m_{aX8_{tot}}} = 3.70. \tag{4.31}$$

It follows that, if we want to derive the endurance for the classical octorotor configuration, we have to consider the effect of the increased mass which, obviously, has a negative impact on the endurance. In Fig. 4.7 the classical and X8 octorotor endurances are compared. The red continuous line represents the X8 octorotor endurance. The blue dash-dot line represents the endurance of the classical octorotor derived from the X8 endurance times $\kappa_{int} = 1.281$ and translated to the left by the $m_{diff_{min}} = 255.60g$ mass. The black dashed line is instead obtained for $\kappa_{int} = 1.160$. Thus, from Fig. 4.7, we observe that the maximum possible endurance gain that can be derived from the classical configuration is, on average, equal to 11%. In particular when considering $\kappa_{int} = 1.160$, that is the interference factor derived experimentally, the endurance of the two configurations are, on average, approximately equal; in other words there is a very low endurance advantage in using the classical octorotor configuration.
From Fig. 4.6 we observe also another important aspect that is the overall size.
Indeed the ratio between the areas of the circles circumscribed around the minimum size classical octorotor and the X8 octorotor is equal to 2.24. This can be considered a significative advantage of the X8 configuration.
For the reasons expressed above the configuration chosen for this work is the X8 one.

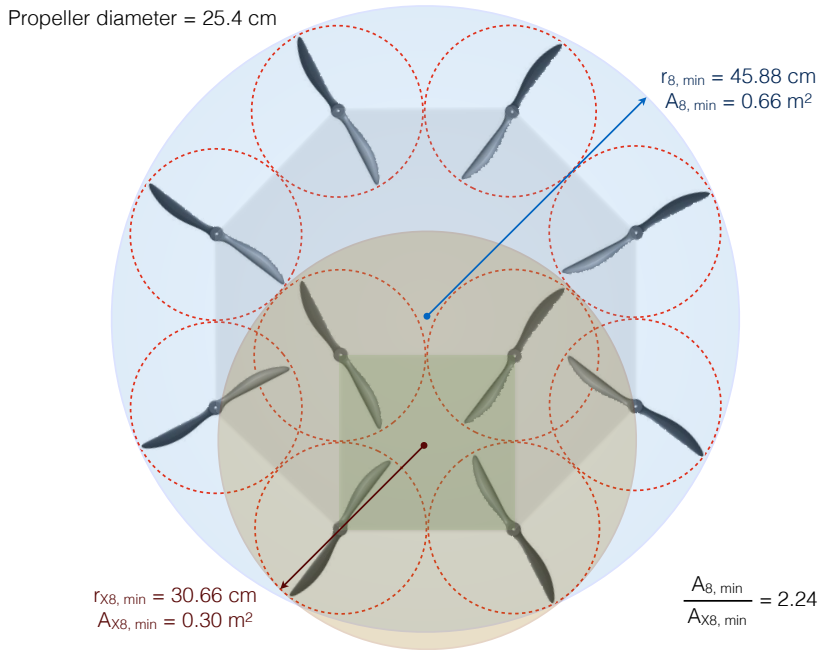

Figure 4.6: Comparison between the classical and X8 octorotor minimum size

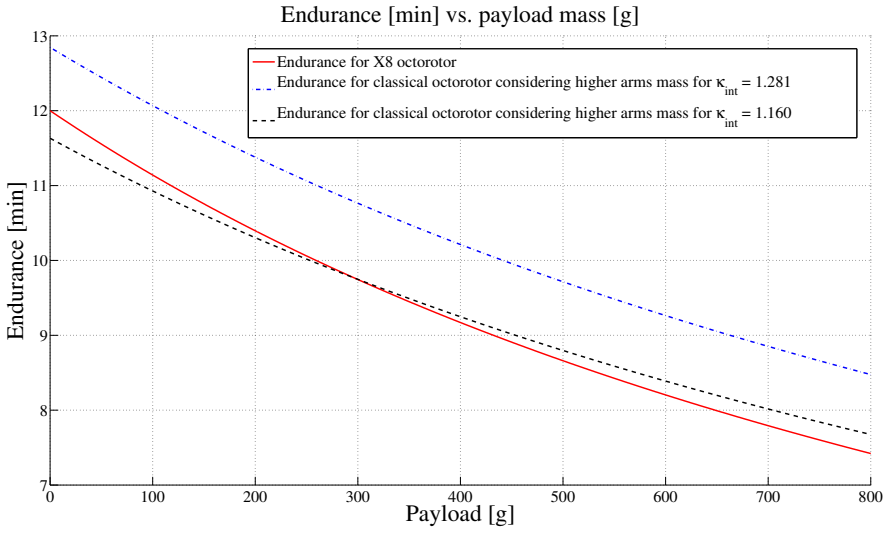

Figure 4.7: Comparison between the classical and X8 octorotor endurance

4.7 Telemetry and Ground Control Station

Wireless telemetry has been added using two 433 MHz XBee modules, the airborne and the ground transceiver. This solution has been chosen to avoid interferences with the 2.4 GHz R.C. gear and because of its guaranteed outdoor line-of-sight range of 1.5 Km.
In Fig. 4.8 it is possible to see the realised Ground Control Station (G.C.S.). It is com-

Figure 4.8: Portable Ground Control Station

posed of a Fujitsu STYLISTIC Q572 tablet running Microsoft Windows 7, a Futaba 8FG radio transmitter with 14 channels and a radio tray attached to a shoulder harness. The total weight of the G.C.S. hardware is 2.1 Kg therefore it can be considered easily portable. The G.C.S. software is open source and it is named *Mission Planner*. This software is based on the open source MAVLink Micro Aerial Vehicle Communication Protocol. This hardware and software setup allows visualising and recording several telemetry information like

- the UAV attitude in terms of roll, pitch and yaw angles,
- the current three dimensional G.P.S. positioning and heading,
- the throttle level,

- the airborne lithium-polymer battery level,
- the airspeed (if airspeed sensors are connected),
- an aerial image of the overflown area, downloaded in real time from the Google servers, through an on site web connection.

The main features of this software are linked to the possibility to change the flight parameters while the aircraft is airborne. Several data can be modified from the G.C.S., for instance the controller gains. Another characteristic that needs to be pointed out is the ability of changing and assigning waypoints directly from the G.C.S. while the UAV is operating and flying. Moreover the MAVLink Protocol supports up to 255 vehicles simultaneously, opening the panorama of *autonomous formation flying vehicles*.

5

Multirotor modelling

The objective of this chapter is to describe the multirotor behaviour firstly with a complete nonlinear system of equations and then with a simplified version of the same model suitable for state estimation and control design.

5.1 Nonlinear model

The forces and moments that act on the multirotor are primarily due to gravity and to the eight propellers.

Fig. 1.19 shows a schematic view of the multirotor systems. Each motor produces a force F and a torque τ. The total force acting on the multirotor is given by

$$F = \sum_{i=1}^{8} F_i \ .$$

The rolling torque is produced by the forces of the right and left motors as

$$\tau_\phi = \lambda_c \ (F_2 + F_3 + F_5 + F_8 - F_1 - F_4 - F_6 - F_7),$$

where λ_c is defined as

$$\lambda_c = \lambda \ cos\left(\frac{\pi}{4}\right) = 0.156 \ [m],$$

and $\lambda = 0.220 \ [m]$ is the length of the multirotor arm. Similarly, the pitching torque is produced by the forces of the front and back motors as

$$\tau_\theta = \lambda_c \ (F_1 + F_2 + F_5 + F_6 - F_3 - F_4 - F_7 - F_8).$$

Due to Newton's third law, the drag of the propellers produces a yawing torque on the body of the multirotor. The direction of the torque will be in the oppositive direction of the motion of the propeller. Therefore the total yawing torque is given by

$$\tau_\psi = (F_1 + F_3 + F_5 + F_7 - F_2 - F_4 - F_6 - F_8)c \ ,$$

where the term c is a proportional constant between the force and the reaction torque produced by the propellers rotation.
Forces and torques need to be converted into the *controller board language*. The thrust vector is therefore translated into a Pulse Width Modulation (PWM) input vector to the hardware. PWM is a commonly used technique for controlling power to inertial electrical devices, made practical by modern electronic power switches. The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a high frequency. The longer the switch is on compared to the off periods, the higher the power supplied to the load is.
In our case, the relation between the single motor thrust (F_i) and the relevant PWM_i input has been derived experimentally. The test setup included a precision balance and a computer to monitor the given input PWM values. The computer was connected to the multirotor processor by means of a serial port. Several tests were performed changing the throttle, i.e. the PWM, and measuring the equivalent mass, thus calculating the generated thrust. The gathered experimental data were interpolated using the cubic function shown in Eq. (5.1) and shown in Fig. 5.1.

$$PWM_i = 6.16F_i^3 - 50.47F_i^2 + 195.5F_i + 1064 \tag{5.1}$$

Note that the PWM commands are defined to be between zero and one. The presented PWM values are not in this range because they are directly scaled to the input requested by the used Arduino controller.
In addition to the force exerted by the motor, also gravity exerts a force on the multirotor. In the vehicle frame $\mathcal{F}^v$, the gravity force acting on the center of mass is given by

$$\mathbf{f}_g^v = \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix}.$$

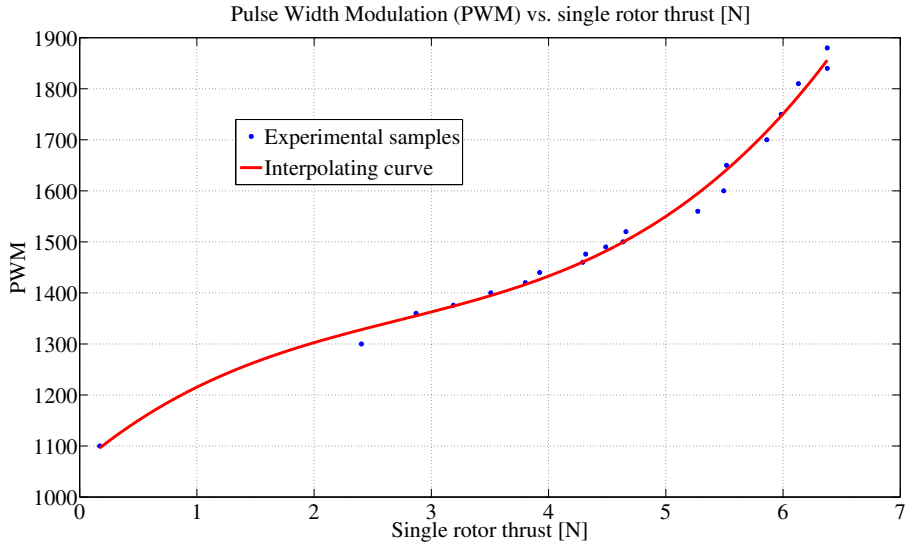

Figure 5.1: $PWM_i(F_i)$ relation

However, since $\mathbf{v}$ in Eq. (3.9) is expressed in $\mathcal{F}^b$, we must transform to the body frame to give

$$\mathbf{f}_g^b = R_v^b \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = \\ = \begin{pmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\cos\theta\cos\phi \end{pmatrix}.$$

Therefore, Eqs. (3.8)-(3.11) become

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{5.2}$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \begin{pmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{pmatrix} + \frac{1}{m}\begin{pmatrix} 0 \\ 0 \\ -F \end{pmatrix} \tag{5.3}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{5.4}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{pmatrix} + \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{pmatrix}, \tag{5.5}$$

where $c\phi \triangleq \cos\phi$ and $s\phi \triangleq \sin\phi$.

5.2 Simplified Models

Equations (5.2)-(5.5) are the equations of motion to be used in our six degree-of-freedom simulator. However, they are not appropriate for control design for several reasons. The first reason is that they are too complicated to gain significant insight into the motion. The second reason is that the position and orientation are relative to the inertial world fixed frame, whereas camera measurements will measure position and orientation of the target with respect to the camera frame.

5.2.1 Model for estimation

For the multirotor, we are not able to estimate the inertial position or the heading angle ψ. Rather, we will be interested in the relative position and heading of the multirotor with respect to a ground target. The relative position of the multirotor will be measured in the vehicle-1 frame, i.e., the vehicle frame after it has been rotated by the heading vector ψ. The vehicle-1 frame is convenient since x, y, and z positions are still measured relative to a flat earth, but they are vehicle centered quantities as opposed to inertial quantities. Let p_x, p_y, and p_z denote the relative position vector between the target and the vehicle resolved in the vehicle-1 frame. Therefore Eq. (5.2) becomes

$$\begin{pmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{pmatrix} = \begin{pmatrix} c\theta & s\phi s\theta & c\phi s\theta \\ 0 & c\phi & -s\phi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}. \tag{5.6}$$

5.2.2 Model for control design

Assuming that ϕ and θ are small, Eq. (5.4) can be simplified as

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \tag{5.7}$$

Similarly, Eq. (5.5) is simplified by assuming that the Coriolis terms qr, pr and pq are small to obtain

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{pmatrix}. \tag{5.8}$$

Combining Eq. (5.7) and (5.8) we get

$$\begin{pmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{pmatrix} = \begin{pmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{pmatrix}. \tag{5.9}$$

Differentiating Eq. (5.2) and neglecting $\dot{R}_b^v$ gives

$$\begin{pmatrix} \ddot{p}_n \\ \ddot{p}_e \\ \ddot{h} \end{pmatrix} = \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix}. \tag{5.10}$$

Neglecting the Coriolis terms and plugging Eq. (5.3) into Eq. (5.10) and simplifying gives

$$\begin{pmatrix} \ddot{p}_n \\ \ddot{p}_e \\ \ddot{h} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} + \begin{pmatrix} -c\phi s\theta c\psi - s\phi s\psi \\ -c\phi s\theta s\psi + s\phi c\psi \\ -c\phi c\theta \end{pmatrix} \frac{F}{m}. \tag{5.11}$$

Therefore, the simplified inertial model is given by

$$\ddot{p}_n = (-\cos\phi \sin\theta \cos\psi - \sin\phi \sin\psi)\frac{F}{m} \tag{5.12}$$

$$\ddot{p}_e = (-\cos\phi \sin\theta \sin\psi + \sin\phi \cos\psi)\frac{F}{m} \tag{5.13}$$

$$\ddot{h} = g - (\cos\phi \cos\theta)\frac{F}{m} \tag{5.14}$$

$$\ddot{\phi} = \frac{1}{J_x}\tau_\phi \tag{5.15}$$

$$\ddot{\theta} = \frac{1}{J_y}\tau_\theta \tag{5.16}$$

$$\ddot{\psi} = \frac{1}{J_z}\tau_\psi. \tag{5.17}$$

The dynamics given in Eqs. (5.12)-(5.17) are expressed in the inertial frame. This is necessary for the simulator. However, we will be controlling position, altitude, and heading using camera frame measurements of a target position. In this setting heading is irrelevant. Therefore, instead of expressing the translational dynamics in the inertial frame, we will express them in the vehicle-1 frame $\mathcal{F}^{v1}$, which is equivalent to the inertial frame after rotating by the heading angle.

Differentiating Eq. (5.6) and neglecting $\dot{R}_b^{v1}$ gives

$$\begin{pmatrix} \ddot{p}_x \\ \ddot{p}_y \\ \ddot{p}_z \end{pmatrix} = \begin{pmatrix} c\theta & s\phi s\theta & c\phi s\theta \\ 0 & c\phi & -s\phi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix}. \tag{5.18}$$

Neglecting the Coriolis terms, plugging Eq. (5.3) into Eq. (5.18) and simplifying gives

$$\begin{pmatrix} \ddot{p}_x \\ \ddot{p}_y \\ \ddot{p}_z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} + \begin{pmatrix} -c\phi s\theta \\ s\phi \\ -c\phi c\theta \end{pmatrix} \frac{F}{m}. \tag{5.19}$$

Therefore, the simplified model in the vehicle-1 frame is given by

$$\ddot{p}_x = -\cos\phi \sin\theta \cdot \frac{F}{m} \tag{5.20}$$

$$\ddot{p}_y = \sin\phi \cdot \frac{F}{m} \tag{5.21}$$

$$\ddot{p}_z = g - \cos\phi\cos\theta \cdot \frac{F}{m} \tag{5.22}$$

$$\ddot{\phi} = \frac{1}{J_x}\tau_\phi \tag{5.23}$$

$$\ddot{\theta} = \frac{1}{J_y}\tau_\theta \tag{5.24}$$

$$\ddot{\psi} = \frac{1}{J_z}\tau_\psi. \tag{5.25}$$

6

Sensors

The objective of this chapter is to describe the main characteristics of the sensors primarily used during the experimental research activity detailed in this thesis. These sensors are rate gyros and accelerometers. Moreover a deep insight into camera sensors is provided.

6.1 Rate Gyros

A MEMS rate gyro contains a small vibrating lever. When the lever undergoes an angular rotation, Coriolis effects change the frequency of the vibration, thus detecting the rotation. The output of the rate gyro is given by

$$y_{gyro} = k_{gyro}\Omega + \beta_{gyro} + \eta_{gyro},$$

where

- y_{gyro} is in Volts,
- k_{gyro} is a gain,
- Ω is the angular rate in radians per second,
- β_{gyro} is a bias term,
- η_{gyro} is zero mean white noise.

The gain k_{gyro} should be given on the datasheet of the sensor. However, due to variations in manufacturing it is imprecisely known. The bias term β_{gyro} is strongly dependent on temperature and should be calibrated prior to each flight. If three rate gyros are aligned along the x, y and z axes of the multirotor, then the rate gyros measure the angular body rates p, q and r as follows:

$$y_{gyro,x} = k_{gyro,x}p + \beta_{gyro,x} + \eta_{gyro,x}$$

$$y_{gyro,y} = k_{gyro,y}q + \beta_{gyro,y} + \eta_{gyro,y}$$

$$y_{gyro,z} = k_{gyro,z}r + \beta_{gyro,z} + \eta_{gyro,z}.$$

MEMS gyros are analog devices that are sampled by the on-board processor. We will assume that the sample rate is given by T_s.

6.2 Accelerometers

A MEMS accelerometer contains a small plate attached to torsion levers. The plate rotates under acceleration and changes the capacitance between the plate and the surrounding walls [12].

The output of the accelerometers is given by

$$y_{acc} = k_{acc}A + \beta_{acc} + \eta_{acc},$$

where

- y_{acc} is in Volts,
- k_{acc} is a gain,
- A is the acceleration in meters per second,
- β_{acc} is a bias term,
- η_{acc} is zero mean white noise.

The gain k_{acc} should be given on the datasheet of the sensor. However, due to variations in manufacturing it is imprecisely known. The bias term β_{acc} is strongly dependent on temperature and should be calibrated prior to each flight.

Accelerometers measure the specific force in the body frame of the vehicle. Another interpretation is that they measure the difference between the acceleration of the aircraft and the gravitational acceleration. A physically intuitive explanation is given in [36]. Additional explanation is given in [44]. Mathematically the accelerometers readings are

$$\begin{pmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{pmatrix} = \frac{d\mathbf{v}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{v} - R_v^b \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix},$$

which can be expressed in component form as

$$\alpha_x = \dot{u} + qw - rv + g\sin\theta$$

$$\alpha_y = \dot{v} + ru - pw - g\cos\theta\sin\phi$$

$$\alpha_z = \dot{w} + pv - qu - g\cos\theta\cos\phi.$$

It can be seen that each accelerometer measures elements of linear acceleration, Coriolis acceleration and gravitational acceleration. The voltage output of an accelerometer is converted into a number corresponding to the voltage inside the autopilot microcontroller by an analog-to-digital converter at a sample rate T_s. Through calibration, this voltage can be converted to a numerical representation of the acceleration in meters per second squared. Assuming that the biases can be removed through the calibration process, the accelerometer signals inside the autopilot can be modelled as

$$y_{acc,x} = \dot{u} + qw - rv + g\sin\theta + \eta_{acc,x}$$

$$y_{acc,y} = \dot{v} + ru - pw - g\cos\theta\sin\phi + \eta_{acc,y}$$

$$y_{acc,z} = \dot{w} + pv - qu - g\cos\theta\cos\phi + \eta_{acc,z}.$$

where $\eta_{acc,x}$, $\eta_{acc,y}$ and $\eta_{acc,z}$ are zero-mean Gaussian processes with variance $\sigma^2_{acc,x}$, $\sigma^2_{acc,y}$ and $\sigma^2_{acc,z}$ respectively. Because of the calibration, the units of $y_{acc,x}$, $y_{acc,y}$ and $y_{acc,z}$ are in $\frac{m}{s^2}$.

6.3 Camera

The control objective is to hold the position of the multirotor over a ground based target that is detected using the vision sensor. In this section we will briefly describe how to estimate p_x and p_y in the vehicle-1 frame.

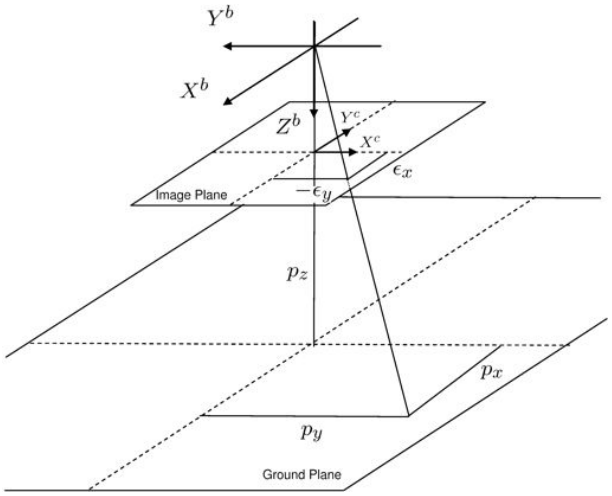

Figure 6.1: Camera model for the multirotor

We will assume that the camera is mounted so that the optical axis of the camera is aligned with the body frame z-axis and so that the x-axis of the camera points out the right of the multirotor and the y-axis of the camera points to the back of the multirotor. The camera model is shown in Fig. 6.1. The position of the target in the vehicle-1 frame is $(p_x,\ p_y,\ p_z)$. The pixel location of the target in the image is $(\epsilon_x,\ \epsilon_y)$. The geometry for p_y is shown in Fig. 6.2. From the geometry shown in Fig. 6.2, we can see that

$$p_y = p_z \tan\left(\phi - \epsilon_x \frac{\eta}{M_y}\right), \tag{6.1}$$

where η is the camera field-of-view and M_y is the number of pixels along the camera y-axis. In Fig. 6.2, both p_y and ϵ_x are negative. Positive values are toward the right rotor. A similar equation can be derived for p_x as

$$p_x = -p_z \tan\left(\theta - \epsilon_y \frac{\eta}{M_y}\right). \tag{6.2}$$

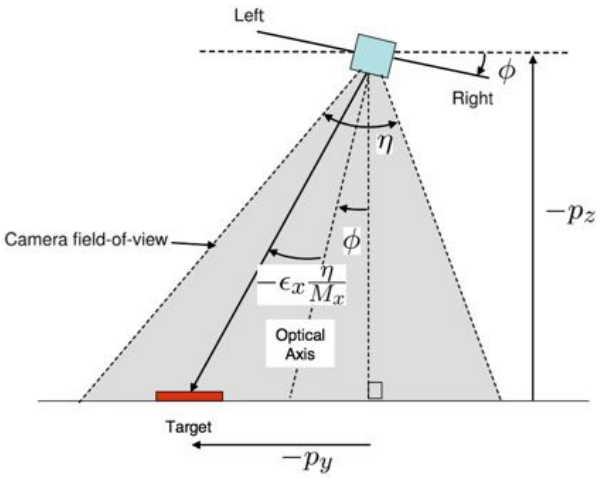

Figure 6.2: Geometry introduced by the vision system. The height above ground is given by $-p_z$, the lateral position error is p_y, the roll angle is ϕ, the field-of-view of the camera is η, the lateral pixel location of the target in the image is ϵ_x and the total number of pixels along the lateral axis of the camera is M_x

7

State Estimation

The objective of this chapter is to describe techniques for estimating the state of the multirotor from sensor measurements. We need to estimate the following states: p_x, p_y, p_z, u, v, w, ϕ, θ, ψ, p, q, r.

The angular rates p, q and r can be obtained by low pass filtering the rate gyros. The remaining states require a Kalman filter. Both are discussed below.

7.1 Low Pass Filters

The Laplace transforms representation of a simple low-pass filter is given by

$$Y(s) = \frac{a}{s+a}U(s),$$

were $u(t)$ is the input of the filter and $y(t)$ is the output. Inverse Laplace transforming we get

$$\dot{y} = -ay + au. \tag{7.1}$$

Using a zeroth order approximation of the derivative we get

$$\frac{y(t+T) - y(t)}{T} = -ay(t) \ + au(t),$$

where T is the sample rate. Solving for $y(t \ + T)$ we get

$$y(t \ + T) = (1 - aT)y(t) \ + aTu(t).$$

For the zeroth order approximation to be valid we need $\alpha T \ll 1$. If we let $\alpha = aT$ then we get the simple form

$$y(t\ +T) = (1-\alpha)y(t)\ +\alpha u(t).$$

Note that this equation has a nice physical interpretation: the new value of y (filtered value) is a weighted average of the old value of y and u (unfiltered value). If u is noisy, then $\alpha \in [0,1]$ should be set to a small value. However, if u is relatively noise free, then α should be close to unity.

In the derivation of the discrete-time implementation of the low-pass filter, it is possible to be more precise. In particular, returning to Eq. (7.1), from linear systems theory, it is well known that the solution is given by

$$y(t\ +T) = e^{-aT}y(t)\ +a\int_0^T e^{-a(T-\tau)}u(\tau)d\tau.$$

Assuming that $u(t)$ is constant between sample periods results in the expression

$$y(t\ +T) = e^{-aT}y(t)\ +a\int_0^T e^{-a(T-\tau)}d\tau u(t) y(t\ +T) =$$

$$= e^{-aT}y(t) + (1-e^{-aT})u(t). \tag{7.2}$$

Note that since

$$e^x = 1 + x + \frac{x^2}{2!} + \ldots$$

we have that

$$e^{-aT} \approx 1 - aT \text{ and } 1 - e^{-aT} \approx aT.$$

Therefore, if the sample rate is not fixed (common for micro-controllers) and it is desired to have a fixed cut-off frequency, then Eq. (7.2) is the preferable way to implement a low-pass filter in digital hardware.

We will use the notation $LPF(\cdot)$ to represent the low-pass filter operator. Therefore $\hat{x} = LPF(x)$ is the low-pass filtered version of x.

7.2 Angular rates p, q and r

The angular rates p, q and r can be estimated by low-pass filtering the rate gyro signals:

$$\hat{p} = LPF(y_{gyro,\mathrm{x}}) \tag{7.3}$$

$$\hat{q} = LPF(y_{gyro,\mathrm{y}}) \tag{7.4}$$

$$\hat{r} = LPF(y_{gyro,\mathrm{z}}). \tag{7.5}$$

7.3 Dynamic Observer Theory

The objective of this section is to briefly review the observer theory. Suppose that we have a linear time-invariant system modelled by the equations

$$\dot{x} = Ax + Bu$$

$$y = Cx.$$

A continuous-time observer for this system is given by the equation

$$\dot{\hat{x}} = \underbrace{A\hat{x} + Bu}_{copy\ of\ the\ model} + \underbrace{L(y - C\hat{x}),}_{correction\ due\ to\ sensor\ reading} \tag{7.6}$$

where $\hat{x}$ is the estimated value of x.
Letting $\tilde{x} = x - \hat{x}$ we observe that

$$\dot{\tilde{x}} = (A - LC)\tilde{x}$$

which implies that the observation error decays exponentially to zero if L is chosen such that the eigenvalues of A–LC are in the open left half of the complex plane.

In practice, the sensors are usually sampled and processed in digital hardware at some sample rate T_s. How do we modify the observer equation shown in Eq. (7.6) to account for sampled sensor readings?

The typical approach is to propagate the system model between samples using the equation

$$\dot{\hat{x}} = A\hat{x} + Bu \tag{7.7}$$

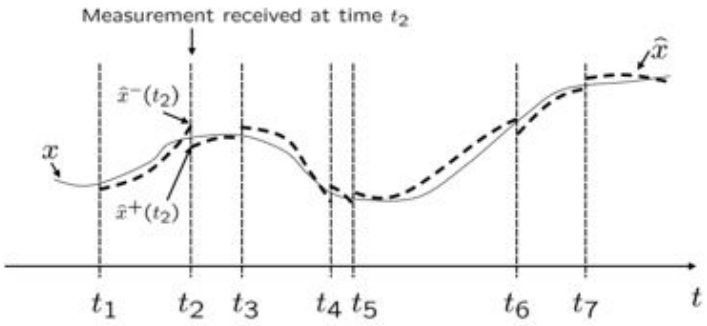

Figure 7.1: Observation process

and then to update the estimate when a measurement is received using the equation

$$\hat{x}^+ = \hat{x}^- + L(y(t_k) - C\hat{x}^-),$$

where t_k is the instant in time that the measurement is received and $\hat{x}^-$ is the state estimate produced by Eq. (7.7) at time t_k. Equation (7.7) is then reinstantiated with initial conditions given by $\hat{x}^+$. The continuous-discrete observer is summarized in Tab. 7.1.

The observation process is shown graphically in Fig. 7.1. Note that it is not necessary to have a fixed sample rate. The continuous-discrete observer can be implemented using Algorithm 1 presented in Tab.7.1.

Note that we did not use the fact that the process was linear. Suppose instead that we have a nonlinear system of the form,

$$\dot{x} = f(x,\ u) \tag{7.8}$$

$$y = c(x) \tag{7.9}$$

then the continuous discrete observer is given in Tab. 7.2.
The real question is how to pick the observer gain L.

Table 7.1: Continuous-Discrete Linear Observer

Algorithm 1 Continuous-Discrete Observer

1: Initialize: $\hat{x} = 0$.
2: Pick an output sample rate T_{out} which is much
less than the sample rates of the sensors.
3: At each sample time T_{out}:
4: **for** $i = 1$ to N **do** {Prediction: Propagate the state equation.}
5: $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right)(A\hat{x} + Bu)$
6: **end for**
7: **if** A measurement has been received from sensor i
then {Correction: Measurement Update.}
8: $\hat{x} = \hat{x} + L_i(y_i - C_i\hat{x})$
9: **end if**

System model :
$\dot{x} = Ax + Bu$
$y(t_k) = Cx(t_k)$
Initial Condition $x(0)$.
Assumptions :
Knowledge of $A,\ B,\ C,\ u(t)$.
No measurement noise.
Prediction : **In between measurements** ($t\ \in [t_{k-1},\ t_k)$):
Propagate $\dot{\hat{x}} = A\hat{x} + Bu$.
Initial condition is $\hat{x}^+(t_{k-1})$.
Label the estimate at time t_k as $\hat{x}^-(t_k)$.
Correction : **At sensor measurement** ($t\ = t_k$):
$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L(y(t_k) - C\hat{x}^-(t_k))$.

Table 7.2: Continuous-Discrete Nonlinear Observer

System model :
$\dot{x} = f(x,\ u)$
$y(t_k) = c(x(t_k))$
Initial Condition $x(0)$.
Assumptions :
Knowledge of $f,\ c,\ u(t)$. No measurement noise.
Prediction : **In between measurements** ($t\ \in [t_{k-1},\ t_k)$):
Propagate $\hat{x} = f(\hat{x},\ u)$.
Initial condition is $\hat{x}^+(t_{k-1})$.
Label the estimate at time t_k as $\hat{x}^-(t_k)$.
Correction : **At sensor measurement** ($t\ = t_k$):
$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L(y(t_k) - c(\hat{x}^-(t_k)))$.

7.4 Essentials from Probability Theory

Let $X = (x_1,\ \ldots,\ x_n)^T$ be a random vector whose elements are random variables. The mean or expected value of X is denoted by

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} E\{x_1\} \\ \vdots \\ E\{x_n\} \end{pmatrix} = E\{X\},$$

where

$$E\{x_i\} = \int \xi f_i(\xi) d\xi,$$

and $f(\cdot)$ is the probability density function for x_i. Given any pair of components x_i and x_j of X, we denote their covariance as

$$cov(x_i,\ x_j) = \Sigma_{ij} = E\{(x_i - \mu_i)(x_j - \mu_j)\}.$$

The covariance of any component with itself is the variance, i.e.,

$$var(x_i) = cov(x_i,\ x_i) = \Sigma_{ii} = E\{(x_i - \mu_i)(x_i - \mu_i)\}.$$

The standard deviation of x_i is the square root of the variance:

$$stdev\ (x_i) = \sigma_i = \sqrt{\Sigma_{ii}}.$$

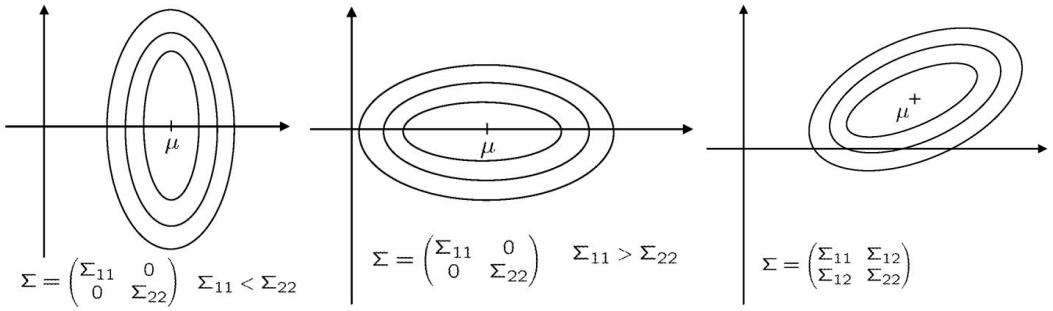

Figure 7.2: Level curves for the probability density function of a 2D Gaussian random variable

The covariances associated with a random vector X can be grouped into a matrix known as the covariance matrix:

$$\Sigma = \begin{pmatrix} \sum_{11} & \sum_{12} & \cdots & \sum_{1n} \\ \sum_{21} & \sum_{22} & \cdots & \sum_{2n} \\ \vdots & & \ddots & \vdots \\ \sum_{n1} & \sum_{n2} & \cdots & \sum_{nn} \end{pmatrix} = E\{(X-\boldsymbol{\mu})(X-\boldsymbol{\mu})^T\} = E\{XX^T\} - \boldsymbol{\mu}\boldsymbol{\mu}^T.$$

Note that $\Sigma = \Sigma^T$ so that Σ is both symmetric and positive semi-definite, which implies that its eigenvalues are real and nonnegative.

The probability density function for a Gaussian random variable is given by

$$f_x(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x-\mu_x)^2}{\sigma_x^2}},$$

where μ_x is the mean of x and σ_x is the standard deviation. The vector equivalent is given by

$$f_X(X) = \frac{1}{\sqrt{2\pi \det \Sigma}} \exp\left[-\frac{1}{2}(X-\boldsymbol{\mu})^T\Sigma^{-1}(X-\boldsymbol{\mu})\right],$$

in which case we write

$$X \sim \mathcal{N}(\boldsymbol{\mu},\ \Sigma),$$

and say that X is normally distributed with mean μ and covariance Σ. Fig. 7.2 shows the level curves for a 2D Gaussian random variable with different covariance matrices.

7.5 Derivation of the Kalman Filter

In this section we assume the following state model:

$$\dot{x} = Ax + Bu + G\xi$$

$$y_k = Cx_k + \eta_k,$$

where $y_k = y(t_k)$ is the k^{th} sample of y, $x_k = x(t_k)$ is the k^{th} sample of x, η_k is the measurement noise at time t_k, ξ is a zero-mean Gaussian random process with covariance Q and η_k is a zero-mean Gaussian random variable with covariance R. Note that the sample rate does not need to be be fixed. The observer will therefore have the form presented in Tab. 7.3. Our objective is to pick L to minimize $tr(P(t))$ where $P(t)$ is the covariance of the estimation error at time t, defined as

$$P(t) \triangleq E\{\tilde{x}(t)\tilde{x}(t)^T\}. \tag{7.10}$$

Table 7.3: Observer

Prediction : **In between measurements** ($t \in [t_{k-1},\ t_k]$):
Propagate $\dot{\hat{x}} = A\hat{x} + Bu$.
Initial condition is $\hat{x}^+(t_{k-1})$.
Label the estimate at time t_k as $\hat{x}^-(t_k)$.
Correction : **At sensor measurement** ($t = t_k$):
$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L(y(t_k) - C\hat{x}^-(t_k))$.

7.5.1 Between Measurements

Differentiating $\tilde{x}$ we get

$$\dot{\tilde{x}} = \dot{x} - \dot{\hat{x}} =$$

$$= Ax + Bu + G\xi - A\hat{x} - Bu =$$

$$= A\tilde{x} + G\xi.$$

Then we have that

$$\tilde{x}(t) = e^{At}\tilde{x}_0 + \int_0^t e^{A(t-\tau)}G\xi(\tau)d\tau.$$

We can therefore compute the evolution for P as

$$\dot{P} = \frac{d}{dt}E\{\tilde{x}\tilde{x}^T\} =$$

$$= E\{\dot{\tilde{x}}\tilde{x}^T + \tilde{x}\dot{\tilde{x}}^T\} =$$

$$= E\{A\tilde{x}\tilde{x}^T + G\xi\tilde{x}^T + \tilde{x}\tilde{x}^T A^T + \tilde{x}\xi^T G^T\} =$$

$$= AP + PA^T + GE\{\xi\tilde{x}^T\}^T + E\{\tilde{x}\xi^T\}G^T.$$

As in the previous section we get

$$E\{\xi\tilde{x}^T\} = E\Big\{\xi(t)\tilde{x}_0 e^{A^T t} + \int_0^t \xi(t)\xi^T(\tau)G^T e^{A^T(t-\tau)}d\tau \Big\} =$$

$$= \frac{1}{2}QG^T,$$

which implies that

$$\dot{P} = AP + PA^T + GQG^T.$$

7.5.2 At Measurements

At a measurement we have that

$$\tilde{x}^+ = x - \hat{x}^+ =$$

$$= x - \hat{x}^- - L(Cx + \eta - C\hat{x}^-) =$$

$$= \tilde{x}^- - LC\tilde{x}^- - L\eta.$$

Therefore

$$P^+ = E\{\tilde{x}^+\tilde{x}^{+T}\} =$$

$$= E\Big\{(\tilde{x}^- - LC\tilde{x}^- - L\eta)(\tilde{x}^- - LC\tilde{x}^- - L\eta)^T\Big\} =$$

$$= E\Big\{\tilde{x}^-\tilde{x}^{-T} - \tilde{x}^-\tilde{x}^{-T}C^T L^T - \tilde{x}^-\eta^T L^T +$$

$$-LC\tilde{x}^-\tilde{x}^{-T} + LC\tilde{x}^-\tilde{x}^{-T}C^T L^T + LC\tilde{x}^-\eta^T L^T =$$

$$= -L\eta\tilde{x}^{-T} + L\eta\tilde{x}^{-T}C^T L^T + L\eta\eta^T L^T\Big\} =$$

$$= P^- - P^-C^T L^T - LCP^- + LCP^-C^T L^T + LRL^T. \tag{7.11}$$

Our objective is to pick L to minimize $\text{tr}(P^+)$. A necessary condition is

$$\frac{\partial}{\partial L}\text{tr}(P^+) = -P^-C^T - P^-C^T + 2LCP^-C^T + 2LR = 0 \Rightarrow$$

$$\Rightarrow 2L(R + CP^-C^T) = 2P^-C^T \Rightarrow$$

$$\Rightarrow L = P^-C^T(R + CP^-C^T)^{-1}.$$

Plugging back into Eq. (7.11) gives

$$P^+ = P^- + P^-C^T(R + CP^-C^T)^{-1}CP^- - P^-C^T(R + CP^-C^T)^{-1}CP^- +$$

$$+P^-C^T(R + CP^-C^T)^{-1}(CP^-C^T + R)(R + CP^-C^T)^{-1}CP^- =$$

$$= P^- - P^-C^T(R + CP^-C^T)^{-1}CP^- =$$

$$= (I - P^-C^T(R + CP^-C^T)^{-1}C)P^- =$$

$$= (I - LC)P^-.$$

For linear systems, the continuous-discrete Kalman filter is summarized in Tab. 7.4. If the system is nonlinear, then the Kalman filter can still be applied but we need to linearize the nonlinear equations in order to compute the error covariance matrix P and the Kalman gain L. The extended Kalman filter (EKF) is given in Tab. 7.5 and a procedure to implement the EKF is the Algorithm 2 presented in Tab. 7.6.

Table 7.4: Continuous-Discrete Kalman Filter

System model :

$\dot{x} = Ax + Bu + \xi$

$y_i(t_k) = C_i x(t_k) + \eta_k$

Initial Condition $x(0)$.

Assumptions :

Knowledge of $A,\ B,\ C_i,\ u(t)$.

Process noise satisfies $\xi \sim \mathcal{N}(0,\ Q)$.

Measurement noise satisfies $\eta_k \sim \mathcal{N}(0,\ R)$.

Prediction : **In between measurements** $(t\ \in [t_{k-1},\ t_k])$:

Propagate $\dot{\hat{x}} = A\hat{x} + Bu$.

Propagate $\dot{P} = AP + PA^T + Q$.

Correction : **At the i$^{\text{th}}$ sensor measurement** $(t\ = t_k)$:

$L_i = P^- C_i^T (R_i + C_i P^- C_i^T)^{-1}$,

$P^+ = \ (I\ - L_i C_i) P^-$,

$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L_i(y_i(t_k) - C_i \hat{x}^-(t_k))$.

Table 7.5: Continuous-Discrete Extended Kalman Filter

System model :

$\dot{x} = f(x,\ u) + \xi$

$y_i(t_k) = c_i(x(t_k)) + \eta_k$

Initial Condition $x(0)$.

Assumptions :

Knowledge of $f,\ c_i,\ u(t)$.

Process noise satisfies $\xi \sim \mathcal{N}(0,\ Q)$.

Measurement noise satisfies $\eta_k \sim \mathcal{N}(0,\ R)$.

Prediction : **In between measurements** $(t\ \in [t_{k-1},\ t_k])$:

Propagate $\dot{\hat{x}} = f(\hat{x},\ u)$,

Compute $A = \dfrac{\partial f}{\partial x}|_{x=\hat{x}(t)}$,

Propagate $\dot{P} = AP + PA^T + Q$.

Correction : **At the i$^{\text{th}}$ sensor measurement** $(t\ = t_k)$:

$C_i = \frac{\partial c_i}{\partial x}|_{x=\hat{x}^-}$,

$L_i = P^- C_i^T (R_i + C_i P^- C_i^T)^{-1}$,

$P^+ = \ (I\ - L_i C_i) P^-$,

$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L_i(y_i(t_k) - c_i(\hat{x}^-(t_k)))$.

Table 7.6: Continuous-Discrete Extended Kalman Filter

Algorithm 2 Continuous-Discrete Extended Kalman Filter

Initialize: $\hat{x} = 0$.
Pick an output sample rate T_{out} which is much less than the sample rates of the sensors.
At each sample time T_{out}:
for $i = 1$ to N **do** {Prediction: Propagate the state equation.}
    $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) f(\hat{x},\ u)$
    $A = \frac{\partial f}{\partial x}(\hat{x},\ u)$
    $P = P + \left(\frac{T_{out}}{N}\right)(AP + PA^T + GQG^T)$
end for
if A measurement has been received from sensor i **then** {Correction: Measurement Update.}
    $C_i = \frac{\partial c_i}{\partial x}|_{x=\hat{x}^-}$,
    $L_i = P^- C_i^T (R_i + C_i P^- C_i^T)^{-1}$,
    $P^+ = (I\ - L_i C_i) P^-$,
    $\hat{x}^+(t_k) = \hat{x}^-(t_k) + L_i(y_i(t_k) - c_i(\hat{x}^-(t_k)))$.
end if

7.6 Application to the multirotor

In this section we will discuss the application of the Algorithm 2 shown in Tab. 7.6 to the multirotor. We would like to estimate the state

$$\hat{x} = \begin{pmatrix} \hat{p}_x \\ \hat{p}_y \\ \hat{p}_z \\ \dot{\hat{p}}_x \\ \dot{\hat{p}}_y \\ \dot{\hat{p}}_z \\ \hat{\phi} \\ \hat{\theta} \\ \hat{\psi} \end{pmatrix},$$

where the rate gyros and accelerometers will be used to drive the prediction step and an ultrasonic altimeter and camera will be used in the correction step.

The propagation model is obtained from Eqs. (5.4) and (5.20)-(5.22) as

$$f(x,\ u) = \begin{pmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \cos\phi \sin\theta\ \alpha_z \\ -\sin\phi\ \alpha_z \\ g + \cos\phi\cos\theta\ \alpha_z \\ p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ q\cos\phi - r\sin\phi \\ q\frac{\sin\phi}{\cos\theta} + r\frac{\cos\phi}{\cos\theta} \end{pmatrix},$$

where we used the fact that the z-axis of the accelerometer measures $\alpha_z = -F/m$. Differentiating we obtain

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -s_\phi s_\theta \alpha_z & -c_\phi c_\theta \alpha_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -c_\phi \alpha_z & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -s_\phi c_\theta \alpha_z & -c_\phi s_\theta \alpha_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & qc_\phi t_\theta - rs_\phi t_\theta & \frac{qs_\phi + rc_\phi}{c^2\theta} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -qs_\phi - rc_\phi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{qc_\phi - rs_\phi}{c\theta} & -(qs_\phi + rc_\phi)\frac{t_\theta}{c_\theta} & 0 \end{pmatrix}.$$

Note that it may be adequate (not sure) to use a small angle approximation in

the model resulting in

$$f(x,\ u) = \begin{pmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \theta\alpha_z \\ -\phi\alpha_z \\ g + \alpha_z \\ p \\ q \\ r \end{pmatrix},$$

and

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\alpha_z & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

If this form works, then the update equation for P can be coded by hand, significantly reducing the computational burden.

8

Control Design

The control design will be derived directly from Eqs. (5.20)-(5.25). Equations (5.23)-(5.25) are already linear. To simplify Eqs. (5.20)-(5.22) we define

$$u_x \triangleq -\cos\phi \sin\theta \cdot \frac{F}{m} \tag{8.1}$$

$$u_y \triangleq \sin\phi \cdot \frac{F}{m} \tag{8.2}$$

$$u_z \triangleq g - \cos\phi\cos\theta \cdot \frac{F}{m}, \tag{8.3}$$

to obtain

$$\ddot{p}_x = u_x \tag{8.4}$$

$$\ddot{p}_y = u_y \tag{8.5}$$

$$\ddot{p}_z = u_z. \tag{8.6}$$

The control design proceeds by developing PID control strategies for u_x, u_y and u_z. After u_x, u_y and u_z have been computed, we can compute the desired force F, the commanded roll angle ϕ^c and the commanded pitch angle θ^c from Eqs. (8.1)-(8.3) as follows. From Eq. (8.3) solve for F/m as

$$\frac{F}{m} = \frac{g - u_z}{\cos\phi\cos\theta}. \tag{8.7}$$

Substituting Eq. (8.7) into Eq. (8.2) gives

$$u_y = \frac{g - u_z}{\cos\theta}\tan\phi.$$

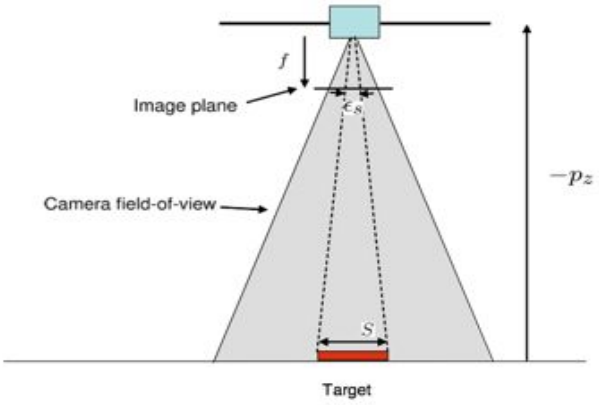

Figure 8.1: The size of the target is S in meters and the size of the target in the image plane is denoted by ϵ_s in pixels. The focal length is f and the height above the target is $-p_z$

Solving for ϕ and letting this be the commanded roll angle gives

$$\phi^c = \tan^{-1}\left(\frac{u_y \cos\theta}{g - u_z}\right). \tag{8.8}$$

Similarly, we can solve for the commanded pitch angle as

$$\theta^c = \tan^{-1}\left(\frac{u_x}{u_z - g}\right). \tag{8.9}$$

8.1 Vision Based Altitude Hold

For the altitude hold we need to develop an expression for u_z to drive p_z to a desired altitude based on the size of the object in the image. We will assume that the camera returns the size of the object in the image plane in pixels, which is denoted by ϵ_s. Fig. 8.1 shows the geometry of the multirotor hovering over a target of size S. From similar triangles we have that

$$\frac{-p_z}{S} = \frac{f}{\epsilon_s}.$$

Solving for p_z and differentiating we obtain

$$\dot{p}_z = \frac{fS\dot{\epsilon}_s}{\epsilon_s^2}. \tag{8.10}$$

Differentiating again gives

$$\ddot{p}_z = \frac{fS\ddot{\epsilon}_s}{\epsilon_s^2} - 2fS\frac{\dot{\epsilon}_s^2}{\epsilon_s^3}.$$

Substituting $u_z = \ddot{p}_z$ and solving for $\ddot{\epsilon}_s$ gives

$$\ddot{\epsilon}_s = \left(\frac{\epsilon_s^2}{fS}\right) u_z + 2\frac{\dot{\epsilon}_s^2}{\epsilon_s}.$$

Defining

$$u_s \triangleq \left(\frac{\epsilon_s^2}{fS}\right) u_z + 2\frac{\dot{\epsilon}_s^2}{\epsilon_s} \tag{8.11}$$

we get

$$\ddot{\epsilon}_s = u_s.$$

We can now derive a PID controller to drive $\epsilon_s \to \epsilon_s^c$ as

$$u_s = k_{p_s}\left(\epsilon_s^c - \epsilon_s\right) - k_{d_s}\dot{\epsilon}_s + k_{i_s}\int_0^t \left(\epsilon_s^c - \epsilon_s\right) d\tau.$$

Solving (8.11) for u_z we get

$$u_z = \frac{fS}{\epsilon_s^2} k_{p_s}\left(\epsilon_s^c - \epsilon_s\right) - \left(k_{d_s}\frac{fS}{\epsilon_s^2} + 2\frac{fS\dot{\epsilon}_s}{\epsilon_s^3}\right)\dot{\epsilon}_s + k_{i_s}\frac{fS}{\epsilon_s^2}\int_0^t \left(\epsilon_s^c - \epsilon_s\right) d\tau. \tag{8.12}$$

The downside to this equation is that it requires knowledge of the target size S and the focal length f. This requirement can be removed by incorporating fS into the PID gains by defining

$$\hat{k}_{p_s} \triangleq fSk_{p_s}$$

$$\hat{k}_{i_s} \triangleq fSk_{i_s}$$

$$\hat{k}_{d_s} \triangleq fSk_{d_s},$$

and by noting from Eq. (8.10) that $\frac{fS\dot{\epsilon}_s}{\epsilon_s^2} = w$. Therefore Eq. (8.12) becomes

$$u_z = \hat{k}_{p_s}\frac{\left(\epsilon_s^c - \epsilon_s\right)}{\epsilon_s^2} - \left(\frac{\hat{k}_{d_s}}{\epsilon_s^2} + 2\frac{w}{\epsilon_s}\right)\dot{\epsilon}_s + \frac{\hat{k}_{i_s}}{\epsilon_s^2}\int_0^t \left(\epsilon_s^c - \epsilon_s\right) d\tau. \tag{8.13}$$

8.2 Sonar Based Altitude Hold

The equation of motion for the altitude is given by Eq. (5.22). We will use a PID controller to regulate the altitude as

$$F = k_{p_{p_z}}\left(p_z^c - p_z\right) - k_{d_{p_z}}\dot{p}_z + k_{i_{p_z}}\int_0^t \left(p_z^c - p_z\right) d\tau, \tag{8.14}$$

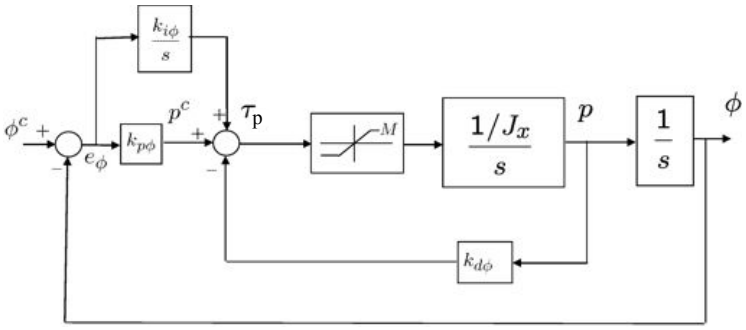

Figure 8.2: A block diagram of the roll attitude hold loop

where p_z^c is the commanded altitude position. In the Laplace domain, the Eq. (8.14) can be rewritten as in (8.15).

$$F(s) = \left(k_{p_{p_z}} + sk_{d_{p_z}} + \frac{k_{i_{p_z}}}{s}\right)(P_z^c(s) - P_z(s)) \tag{8.15}$$

Slight modifications could help improving this controller performance. In the application discussed here, we added a filter to the derivative of the error obtaining the controller transfer function presented in (8.16).

$$F(s) = \left(k_{p_{p_z}} + k_{d_{p_z}}\frac{N}{1 + N\frac{1}{s}} + \frac{k_{i_{p_z}}}{s}\right)(P_z^c(s) - P_z(s)) \tag{8.16}$$

The same approach has been applied to the roll, pitch and yaw angles regulation.

8.3 Roll Attitude Hold

The equation of motion for roll is given by Eq. (5.23) as $\ddot{\phi} = \frac{\tau_p}{J_x}$. We will use a PID controller to regulate the roll attitude as

$$\tau_p = k_{p_\phi}(\phi^c - \phi) - k_{d_\phi}p + k_{i_\phi}\int_0^t (\phi^c - \phi)\, d\tau.$$

A block diagram of the control structure is shown in Fig. 8.2. The gains k_{p_ϕ}, k_{i_ϕ} and k_{d_ϕ} are selected one at a time using a method called successive loop closure. To pick k_{p_ϕ} note that if the input command ϕ^c is a step of value A, then at time $t = 0$, before the integrator has time to begin winding up, τ_p is given by

$$\tau_p(0) = k_{p_\phi}A.$$

Therefore, setting $k_{p_\phi} = \frac{M}{A}$, where M is the command limit, will just saturate the actuators when a step value of A is placed on ϕ^c. To select k_{d_ϕ}, let k_{p_ϕ} be fixed and let $k_{i_\phi} = 0$ and solve for the characteristic equation in Evan's form verses k_{d_ϕ} to obtain

$$1 + k_{d_\phi} \frac{\frac{1}{J_x} s}{s^2 + \frac{k_{p_\phi}}{J_x}} = 0.$$

The derivative gain k_{d_ϕ} is selected to achieve a damping ratio of $\zeta = 0.9$.

The characteristic equation including k_{i_ϕ} in Evan's form verses k_{i_ϕ} is given by

$$1 + k_{i_\phi} \frac{\frac{1}{J_x}}{s^3 + \frac{k_{d_\phi}}{J_x} s^2 + \frac{k_{p_\phi}}{J_x} s} = 0.$$

The integral gain k_{i_ϕ} can be selected so that damping ratio is not significantly changed.

Also here we added a filter to the derivative of the error obtaining the controller transfer function shown in Eq. (8.17).

$$\tau_p(s) = \left(k_{p_\phi} + k_{d_\phi} \frac{N}{1 + N\frac{1}{s}} + \frac{k_{i_\phi}}{s} \right) \left(\Phi^c(s) - \Phi(s) \right). \tag{8.17}$$

8.4 Pitch Attitude Hold

The equation of motion for pitch is given by Eq. (5.24) as $\ddot{\theta} = \frac{\tau_q}{J_y}$. Similar to the roll attitude hold, we will use a PID controller to regulate pitch as

$$\tau_q = k_{p_\theta} \left(\theta^c - \theta \right) - k_{d_\theta} q + k_{i_\theta} \int_0^t \left(\theta^c - \theta \right) d\tau.$$

Also here we added a filter to the derivative of the error obtaining the controller transfer function shown in Eq. (8.18).

$$\tau_q(s) = \left(k_{p_\theta} + k_{d_\theta} \frac{N}{1 + N\frac{1}{s}} + \frac{k_{i_\theta}}{s} \right) \left(\Theta^c(s) - \Theta(s) \right). \tag{8.18}$$

8.5 Vision Based Position Tracking

From Eq. (8.5) the lateral dynamics are given by $\ddot{p}_y = u_y$, where p_y is the relative lateral position which we drive to zero using the PID strategy

$$u_y = -k_{p_y} p_y - k_{d_y} v + k_{i_y} \int_0^t p_y d\tau.$$

The relative position error p_y is given by Eq. (6.1).

From Eq. (8.4) the longitudinal dynamics are given by $\ddot{p}_x = u_x$, where p_x is the relative lateral position which we drive to zero using the PID strategy

$$u_x = -k_{p_x} p_x - k_{d_x} u + k_{i_x} \int_0^t p_x d\tau.$$

The relative position error p_x is given by Eq. (6.2).

8.6 Relative Heading Hold

The heading dynamics is given in Eq. (5.25) as $\ddot{\psi} = \frac{\tau_r}{J_z}$. We define ψ^d as the inertial heading of the target and $\tilde{\psi} \triangleq \psi - \psi^d$ as the relative heading. The camera directly measures $\tilde{\psi}$. Assuming that ψ^d is constant we get $\ddot{\tilde{\psi}} = \frac{\tau_r}{J_z}$. We regulate the relative heading with the PID strategy

$$\tau_r = -k_{p_\psi} \psi - k_{d_\psi} r - k_{i_\psi} \int_0^t \psi d\tau.$$

Also here we added a filter to the derivative of the error obtaining the controller transfer function shown in Eq. (8.19).

$$\tau_r(s) = \left(+k_{p_\psi} + k_{d_\psi} \frac{N}{1 + N\frac{1}{s}} + \frac{k_{i_\psi}}{s} \right) \left(\Psi^c(s) - \Psi(s) \right). \tag{8.19}$$

8.7 Feedforward

When the multirotor is tracking a ground robot, the motion of the robot will cause tracking errors due to the delayed response of the PID controller. If we can communicate with the robot and we know its intended motion, we should be able to use that

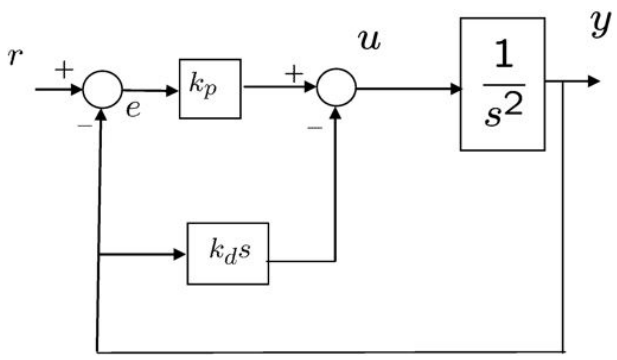

Figure 8.3: Standard PD loop

information to help the multirotor predict where the robot is moving. To motivate our approach, let's consider a simple example to gain intuition. Consider the double integrator system

$$\ddot{y} = u,$$

where y is position and u is commanded acceleration. If r is the reference signal then the standard PD loop is shown in Fig. 8.3.

Let $e = r - y$, then

$$\ddot{e} = \ddot{r} - \ddot{y} =$$

$$= \ddot{r} - u =$$

$$= \ddot{r} - k_p e + k_d \dot{y} =$$

$$= \ddot{r} - k_p e - k_d \dot{e} + k_d \dot{r}.$$

In other words we have

$$\ddot{e} + k_d \dot{e} + k_p e = \ddot{r} + k_d \dot{r}.$$

Therefore, the signal $\ddot{r} + k_d \dot{r}$ drives the error transfer function $\dfrac{1}{s^2 + k_d s + k_p}$.
From the expression $\ddot{e} = \ddot{r} - u$ we see that if instead of $u = k_p e - k_d \dot{y}$, we use

$$u = \ddot{r} + k_p e + k_d \dot{e},$$

then we get

$$\ddot{e} + k_d \dot{e} + k_p e = 0,$$

which ensures that $e(t) \to 0$ independent of $r(t)$. The associated block diagram is shown in Fig. 8.4.

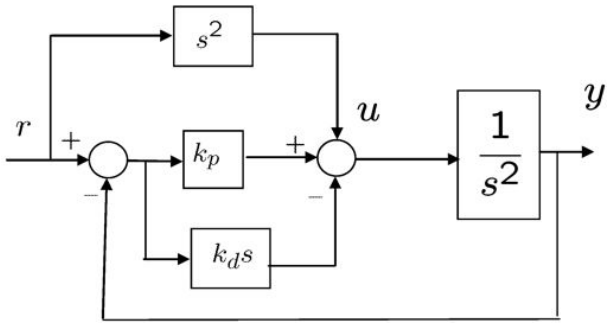

Figure 8.4: Feedforward term is added to a standard PD loop

8.8 Digital implementation of PID loops

In this section we briefly describe how PID loops can be implemented in discrete time. A general PID control signal is given by

$$u(t) = k_p e(t) + k_i \int_{-\infty}^{t} e(\tau)\, d\tau + k_d \frac{de}{dt}(t),$$

where $e(t) = y^c(t) - y(t)$ is the error between the commanded output $y^c(t)$ and the current output $y(t)$. In the Laplace domain, we have

$$U(s) = k_p E(s) + k_i \frac{E(s)}{s} + k_d s E(s).$$

Since a pure differentiator applied to a noisy signal gives unreliable results, the standard approach is to use a band-limited differentiator so that

$$U(s) = k_p E(s) + k_i \frac{E(s)}{s} + k_d \frac{s}{\tau s + 1} E(s).$$

To convert to discrete time, we use the Tustin or trapezoidal rule, where the Laplace variable s is replaced with the z-transform approximation

$$s \mapsto \frac{2}{T_s} \left(\frac{1 - z^{-1}}{1 + z^{-1}} \right),$$

where T_s is the sample period [45]. Letting $I(s) \triangleq \frac{E(s)}{s}$, an integrator in the z domain becomes

$$I(z) = \frac{T_s}{2} \left(\frac{1 + z^{-1}}{1 - z^{-1}} \right) E(z).$$

Transforming to the time domain, we have

$$I[n] = I[n-1] + \frac{T_s}{2} \left(E[n] + E[n-1] \right). \tag{8.20}$$

A formula for discrete implementation of a differentiator can be derived in a similar manner. Letting $D(s) \triangleq \left(\frac{s}{\tau s+1}\right) E(s)$, the differentiator in the z domain is

$$D(z) = \frac{\frac{2}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}{\frac{2\tau}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)+1} E(z) = \frac{\left(\frac{2}{2\tau+T_s}\right)\left(1-z^{-1}\right)}{1-\left(\frac{2\tau-T_s}{2\tau+T_s}\right)z^{-1}} E(z).$$

Transforming to the time domain, we have

$$D[n] = \left(\frac{2\tau - T_s}{2\tau + T_s}\right) D[n-1] + \left(\frac{2}{2\tau+T_s}\right)(E[n] - E[n-1]). \tag{8.21}$$

Matlab code that implements a general PID loop is shown below.

```
function u = pidloop(y_c, y, flag, kp, ki, kd, limit, Ts, tau)
persistent integrator;
persistent differentiator;
persistent error_d1;
if flag==1,
% reset (initialize) persistent variables when flag==1
integrator = 0;
differentiator = 0;
error_d1 = 0; % _d1 means delayed by one time step end
end
error = y_c - y; % compute the current error
integrator = integrator + (Ts/2)*(error + error_d1);
% update integrator
differentiator = (2*tau-Ts)/(2*tau+Ts)*differentiator...
+ 2/(2*tau+Ts)*(error - error_d1);
% update differentiator
error_d1 = error; % update the error for next time through
      % the loop
u=sat(... % implement PID control
kp * error + ... % proportional term
ki * integrator + ... % integral term
kd * differentiator, ... % derivative term
limit... % ensure abs(u)<=limit
);
```

```
25  % implement integrator anti-windup
26  if  ki ~=0
27  u_unsat = kp*error + ki*integrator + kd*differentiator;
28  integrator = integrator + Ts/ki * (u - u_unsat);
29  end
30  function out = sat(in, limit)
31  if in > limit; out = limit;
32  elseif in < -limit;  out = -limit;
33  else out = in;
34  end
```

The inputs on line 1 are:

- the commanded output y^c,
- the current output y,
- $flag$ used to reset the integrator,
- the PID gains k_p, k_i, and k_d,
- the saturation command $limit$,
- the sample time T_s,
- the time constant τ of the differentiator.

Line 11 implements Eq. (8.20) and Lines 12-13 implement Eq. (8.21). A potential problem with a straight-forward implementation of PID controllers is integrator wind up. When the error y^c - y is large and a large error persists for an extended period of time, the value of the integrator, as computed in Line 11, can become large or wind up. A large integrator will cause u, as computed in Lines 19-24, to saturate, which will cause the system to push with maximum effort in the direction needed to correct the error. Since the value of the integrator will continue to wind up until the error signal changes sign, the control signal may not come out of saturation until well after the error has changed sign, which can cause a large overshoot and may potentially destabilise the system. Since integrator wind up can destabilise the autopilot loops, it is important that each loop have an anti-wind-up scheme. A number of different

anti-wind-up schemes are possible. A particularly simple scheme, which is shown in Lines 26-29, is to subtract from the integrator exactly the amount needed to keep u at the saturation bound. In particular, let

$$u^-_{unsat} = k_p e + k_d D + k_i I^-$$

denote the unsaturated control value before updating the integrator, where I^- is the value of the integrator before applying the anti-wind-up scheme and let

$$u^+_{unsat} = k_p e + k_d D + k_i I^+$$

denote the unsaturated control value after updating the integrator, where

$$I^+ = I^- + \Delta I,$$

and ΔI is the update. The objective is to find ΔI so that $u^+_{unsat} = u$, where u is the value of the control after the saturation command is applied. Noting that

$$u^+_{unsat} = u^-_{unsat} + k_i \Delta I,$$

we can solve for ΔI to obtain

$$\Delta I = \frac{1}{k_i}\left(u - u^-_{unsat}\right).$$

The multiplication by T_s in line 28 is to account for the sampled-data implementation.

8.9 Linear Quadratic Regulator

An appealing alternative for the control of Multiple Input Multiple Output (MIMO) systems is the Linear Quadratic Regulator, usually referred to by the abbreviation LQ or even LQR. A detailed study of this topic can be found in [46].
Given a state-space description of the plant as in Eq.(8.22),

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{8.22}$$

where

- x is the state vector defined as

$$x = \begin{bmatrix} p_z & \dot{p}_z & \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} \end{bmatrix}^T, \tag{8.23}$$

- u is the command vector defined as

$$u = \begin{bmatrix} F & \tau_p & \tau_q & \tau_r \end{bmatrix}^T, \tag{8.24}$$

the state-feedback matrix of gains is determined, in the LQ approach, as

$$K_{lqr}(\infty) = R^{-1}B^T S(\infty) \tag{8.25}$$

where $S(\infty) = S$ is the solution of the Algebraic Riccati Equation (A.R.E.)

$$A^T S + SA - SBR^{-1}B^T S + Q = 0 \tag{8.26}$$

which minimizes the linear quadratic cost function

$$J_\infty = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u)dt \tag{8.27}$$

considering the infinite horizon situation. In order to improve the LQ regulation performances only the adjustment of the weighing matrices Q and R, in the minimisation criterion (8.27), is required. Q and R are diagonal and can be initialized as identity matrices. By increasing the weight of the $q_{i,i}$ element of matrix Q, the regulation on the i^{th} state becomes faster, usually requiring a stronger control action, whereas the penalisation of the j^{th} input is obtained by increasing the weight of the $r_{j,j}$ element of matrix R.

9

Simulations and tests

Several simulations of the nonlinear linear system have been realized using Matlab Simulink®. This environment can be used for multi-domain simulations and Model-Based Design for dynamic systems. It provides an interactive graphical environment and a customizable set of block libraries which allows to design, simulate, implement and test a variety of time-varying systems. The model of the whole system is composed of several interconnected blocks in a classical feedback structure.

9.1 PID control simulation

Target of the virtual simulation was to find the starting values of the constant control gains k_p, k_i, k_d and N which optimized the behaviour of the system in tracking the reference variables. These values, presented in Tab. 9.1, have been found empirically, with a *trial and error* approach and have been tested on a nonlinear Simulink® simulation of the multirotor system. Fig. 9.1 shows the behaviour of the PID regulated

Table 9.1: PID control gains

	k_p	k_i	k_d	N
p_z	2.21	0.05	8.46	6.41
ϕ	4.83	1.82	1.17	101.60
θ	4.83	1.82	1.17	101.60
ψ	4.22	1.10	1.47	70.30

system in case of an aggressive maneuver in which we want the multirotor to reach the state specified by (9.1).

$$\vec{x}_{ref} = \begin{bmatrix} p_{z_{ref}} \\ \phi_{ref} \\ \theta_{ref} \\ \psi_{ref} \end{bmatrix} = \begin{bmatrix} 1[m] \\ 0[deg] \\ 45[deg] \\ -45[deg] \end{bmatrix} \tag{9.1}$$

In Fig. 9.1, it is interesting to observe the secondary effect, on the roll dynamics, due

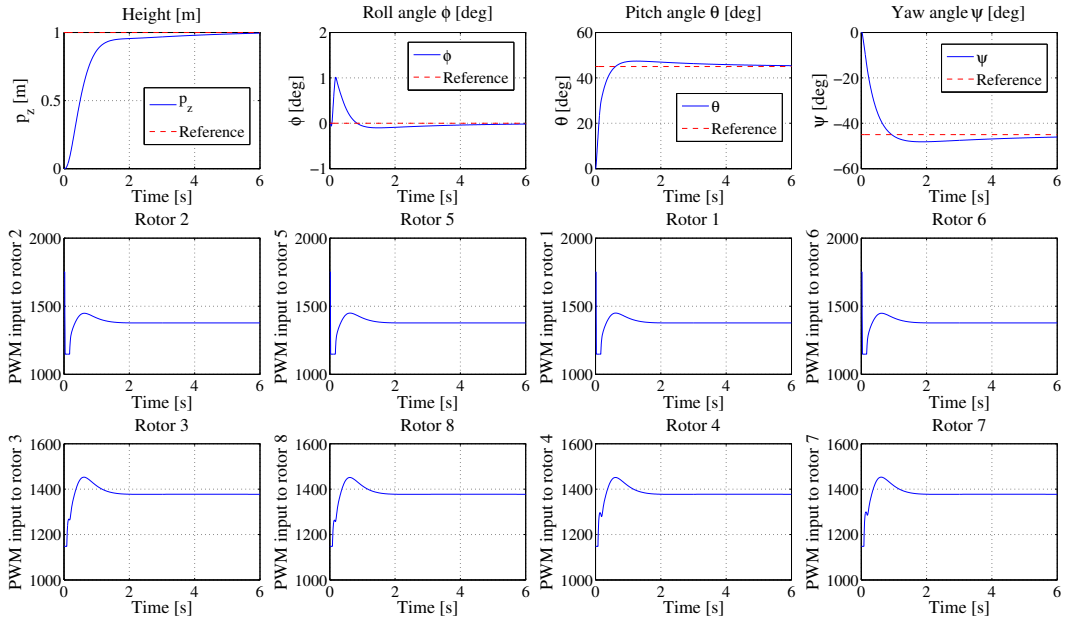

Figure 9.1: Nonlinear dynamics of a PID regulated aggressive maneuver; the rotors position in the figure recalls the physical rotors position in the multirotor

to the coupling with the imposed pitch dynamics. This effect is considered negligible, therefore not appreciable, in the linear modelling.

9.2 LQR control simulation

A virtual model of the LQ regulated dynamics has been realized in the Matlab Simulink® environment using a continuous time form of the model matrices. The error between the current state and the reference signal, times the K_{lqr} matrix, gives the command inputs. The simulation was conceived to help finding, iteratively, the elements of the matrices Q and R and, consequently, K_{lqr}, which optimized the behaviour of the closed loop system in pursuing the target variables. These matrices are presented in (9.2) and (9.3).

$$Q = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9.2}$$

$$R = \begin{bmatrix} 0.17 & 0 & 0 & 0 \\ 0 & 0.17 & 0 & 0 \\ 0 & 0 & 0.17 & 0 \\ 0 & 0 & 0 & 0.17 \end{bmatrix} \tag{9.3}$$

Consequently the K_{lqr} matrix has been calculated and it is shown in (9.4).

$$K_{lqr} = \begin{bmatrix} 5.48 & 0 & 0 & 0 \\ 5.91 & 0 & 0 & 0 \\ 0 & 5.48 & 0 & 0 \\ 0 & 2.50 & 0 & 0 \\ 0 & 0 & 5.48 & 0 \\ 0 & 0 & 2.50 & 0 \\ 0 & 0 & 0 & 3.46 \\ 0 & 0 & 0 & 2.51 \end{bmatrix}^T \tag{9.4}$$

Fig. 9.2 shows the behaviour of the LQ regulated system in case of an aggressive maneuver in which we want the multirotor to reach the state specified by (9.1). As observed in the PID regulated case, in Fig. 9.2, it is interesting to appreciate the secondary effect, on the roll dynamics, due to the coupling with the imposed pitch dynamics.

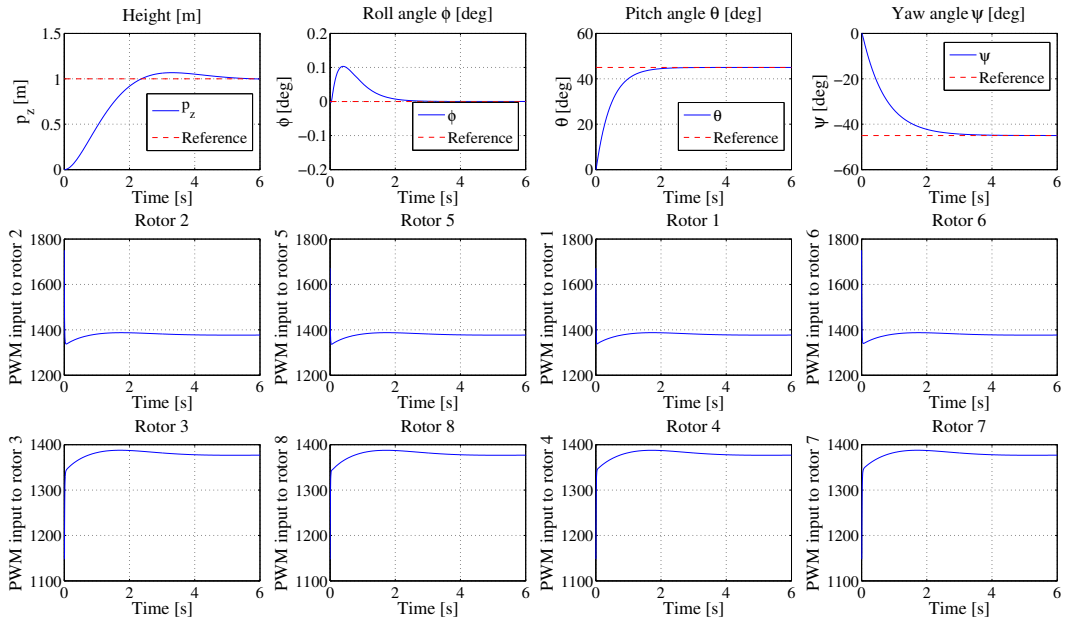

Figure 9.2: Nonlinear dynamics of a LQ regulated aggressive maneuver; the rotors position in the figure recalls the physical rotors position in the multirotor

9.3 PID vs. LQ

This section compares the performances of the two regulation techniques applied to the nonlinear model of the multirotor. In particular, Fig. 9.3 shows the height, roll, pitch and yaw closed loop responses to reach a step reference value. These responses can be synthetically evaluated on the basis of some classical parameters, known from the basic control theory as explained in [47]. In detail, Tab. 9.2 refers to the height dynamics control, Tab. 9.3 refers to the roll and pitch dynamics and Tab. 9.4 presents the yaw response properties. Practically both control methods allow to achieve very satisfactory results and it is not technically reasonable to state that one approach performs always better than the other, since both can be always further improved by means of a more accurate tuning.

Nevertheless it is worthwhile to mention that the two different control techniques, although resulting in comparable reference tracking performances, are characterized by totally different control inputs strategies, as illustrated in Fig. 10.21. This figure shows

the control inputs from the initial time instant to 0.5 [s] in order to zoom the difference between the two strategies. The analysis of the control inputs efforts denotes a higher and more quickly time varying control actions demand in the PID approach.

As already explained in the relevant theory description, the LQR method gives the chance to directly calibrate the references tracking error and the control actions demand, while in the PID approach the control inputs are not directly tuned but they result, as a consequence, from the PID gains definition. This characteristic represents, without any doubt, an advantage of the LQR with respect to the PID control technique.

Another interesting observation is relevant to the cross-coupling between the roll and pitch dynamics. As visible from Fig. 9.1 and Fig. 9.2, while requiring less aggressive control inputs, the LQR approach is able to minimize by ten times this undesired effect.

Table 9.2: Height: step response properties

	PID	LQR
Rise time	0.68 [s]	1.52 [s]
Settling time	64.08 [s]	10.23 [s]
Overshoot	68.72 %	27.86 %
Undershoot	0 %	0 %
Peak	1.69 [m]	1.28 [m]
Peak time	3.08 [s]	3.75 [s]

Table 9.3: Roll and pitch: step response properties

	PID	LQR
Rise time	0.43 [s]	0.99 [s]
Settling time	4.78 [s]	1.77 [s]
Overshoot	7.65 %	0 %
Undershoot	0 %	0 %
Peak	1.08 [rad]	1.00 [rad]
Peak time	1.30 [s]	15.02 [s]

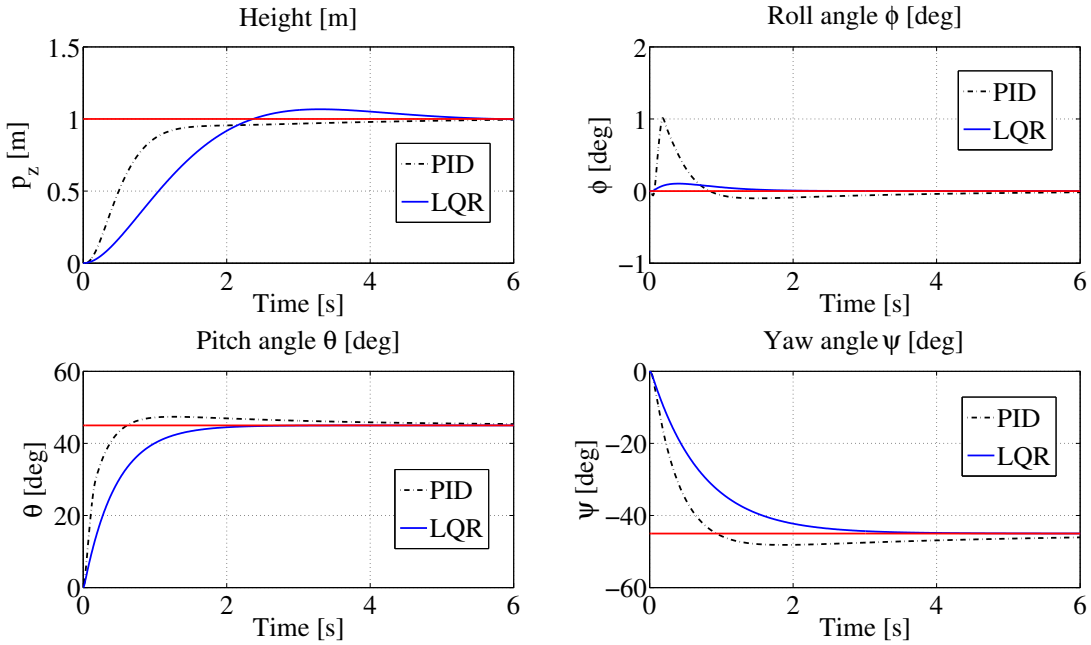

Figure 9.3: Height (p_z), roll (ϕ), pitch (θ) and yaw (ψ) angles regulation performance: PID vs. LQR

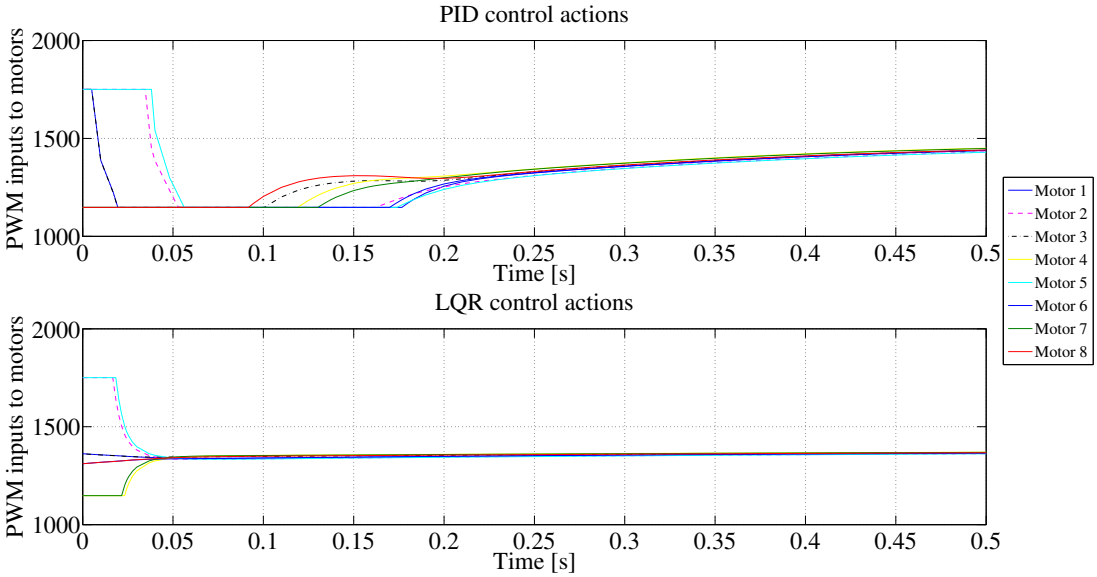

Figure 9.4: Control actions: PID vs. LQR

Table 9.4: Yaw: step response properties

	PID	LQR
Rise time	0.56 [s]	1.55 [s]
Settling time	5.81 [s]	2.79 [s]
Overshoot	5.64 %	0 %
Undershoot	0 %	0 %
Peak	1.06 [rad]	1.00 [rad]
Peak time	1.84 [s]	23.40 [s]

9.4 Failure simulation

The simulation developed in Matlab Simulink® was designed to describe the behaviour of the controlled system even in case of one or more motors failures. In particular it was simulated the failure of the front right upper motor, identified as motor "1" in Fig. 1.19. In order to have an acceptable behaviour of the PID controlled plant it was necessary to retune the control gains with respect to the case without failures. The new found values are presented in Tab. 9.5. The system behaviour is shown in Fig. 9.5

Table 9.5: PID control gains retuned for improved performance in case of one motor failure

	k_p	k_i	k_d	N
p_z	33.37	2.65	35.02	21.23
ϕ	31.75	25.68	2.74	212.29
θ	31.75	25.68	2.74	212.29
ψ	37.18	28.38	5.40	172.04

where the position of the motors inputs graphs recalls the disposition of the motors in the multirotor in order to improve the figure readability. In more details the first row of graphs in Fig. 9.5 shows the system height and attitude regulation capability, in the presence or in the absence of a motor failure. The second and third rows illustrate the relevant PWM inputs to the motors. One of the latter graphs can be distinguished for the different background color. That graph shows clearly the constant input given to the motor "1" that corresponds to a null force from rotor 1. It can be seen either as a failure of the motor or as a complete failure of the propeller. Fig. 9.5 highlights that,

with adequate gains, a PID control structure is able regulate the multirotor system even in case of loss or damage of one propeller and/or motor.

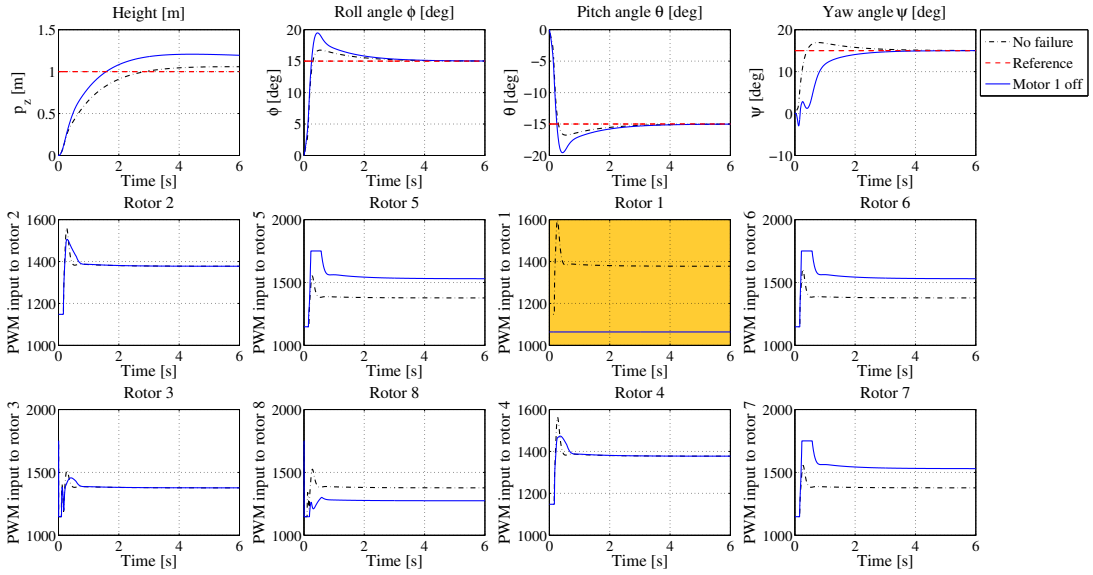

Figure 9.5: PID control simulation in case of motor "1" failure

9.4.1 Cooper-Harper rating evaluation

The behaviour of the system in case of a single motor failure has been evaluated also experimentally by means of the Cooper-Harper rating scale, described in [3]. The Cooper-Harper rating scale is a set of criteria used by test pilots and flight test engineers to evaluate the "handling qualities" of an aircraft during flight test. As shown in Fig. 9.6, the scale ranges from 1 to 10, with 1 indicating the best handling characteristics and 10 the worst. "Handling qualities" are defined in [48] as "those qualities or characteristics of an aircraft that govern the ease and precision with which a pilot is able to perform the tasks required in support of an aircraft role". From this definition, it is clear that handling quality is characteristic of the combined performance of the pilot and vehicle acting together as a system in support of an aircraft role. Fundamental to the subject of handling qualities is the definition of the system whose handling is to be assessed. Aircraft and flight control designers often focus on the dynamics of

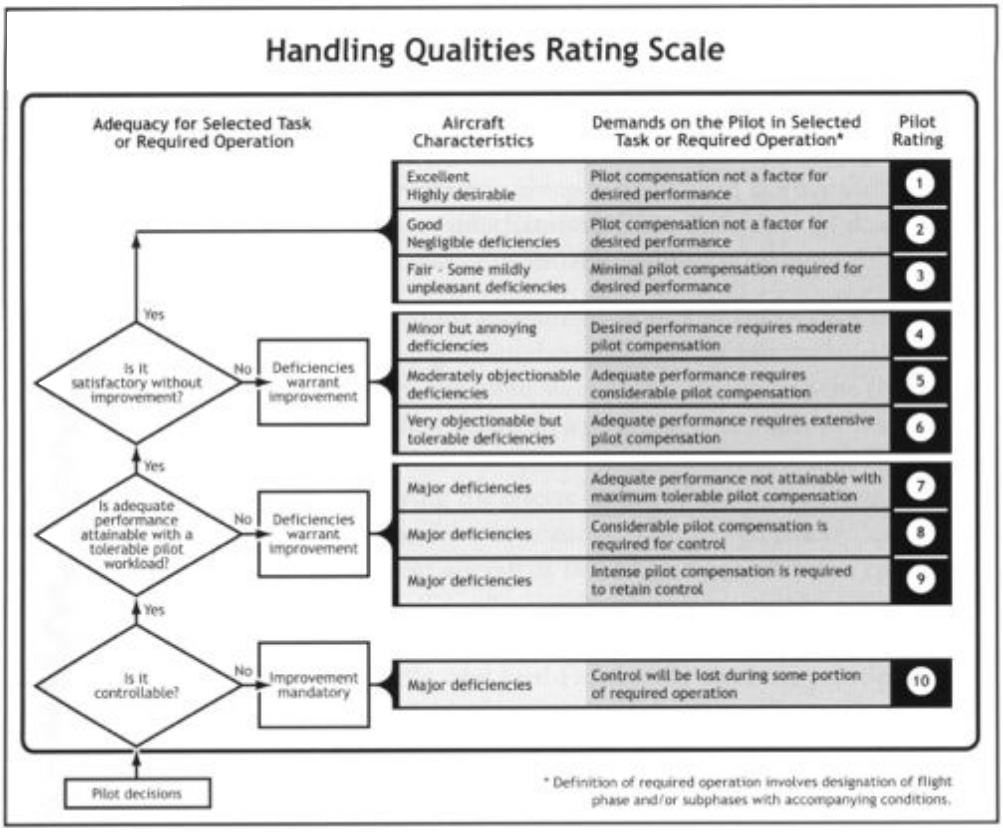

Figure 9.6: Cooper-Harper rating scale [3]

the vehicle, since that is the system element whose characteristics can be selected, the pilot is not readily alterable. The piloted-vehicle dynamics, however, are very much affected by the pilot actions as a controller; he is a key element in the system. In the functional diagram of Fig. 9.7, the pilot role is delineated as the decision-maker of what is to be done, the comparator of what is happening vs. what he wants to happen and the supplier of corrective inputs to the aircraft controls to achieve what he desires. This, then, is the system: the pilot and aircraft acting together as a closed-loop system, the dynamics of which may be significantly different from those of the aircraft acting without him. The handling qualities of a given aircraft are task dependent: what is

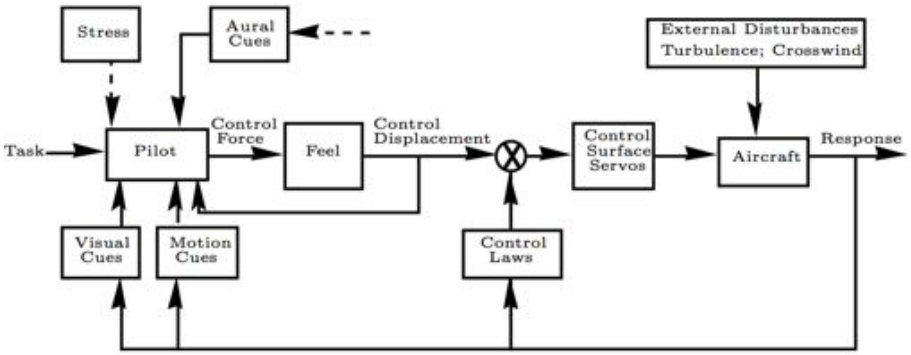

Figure 9.7: Pilot-vehicle dynamic system [3]

preferred for one task may be less desirable for another.

In our specific case two tasks have been selected: taking off and landing with one failed motor. With respect to the former task, we observed the multirotor was still controllable, the performance was attainable with a tolerable pilot workload and no improvements were considered necessary. The multirotor characteristics were evaluated as good since pilot compensation was not a factor for the desired take off performance. The Cooper-Harper rating given by the pilot was 2. Concerning the landing task the rating was 3 since a minimal pilot compensation was required to reach the desired performance.

9.5 Experimental tests

After designing and evaluating the simulated PID and LQ regulation performances, these control laws were implemented on the flight control computer, the APM. This allowed verifying the real physical behaviour of the automatically controlled vehicle. The first tests were conducted allowing one single multirotor rotation per time. As matter of example, Fig. 9.8 shows the data gathered during an experimental test in which the pitch rotation was the only free degree of freedom. The vertical dotted line, visible in Fig. 9.8, underlines the time instant at which the external solicitation was at its maximum and the system was realized to react automatically. The figure shows also the capability of the control architecture to regulate the system at the desired zero pitch angle with some minor residual oscillations depending on the unavoidable measurement noise. Fig. 9.8 shows also the motors inputs necessary to exhibit the regulated pitch angle behaviour.

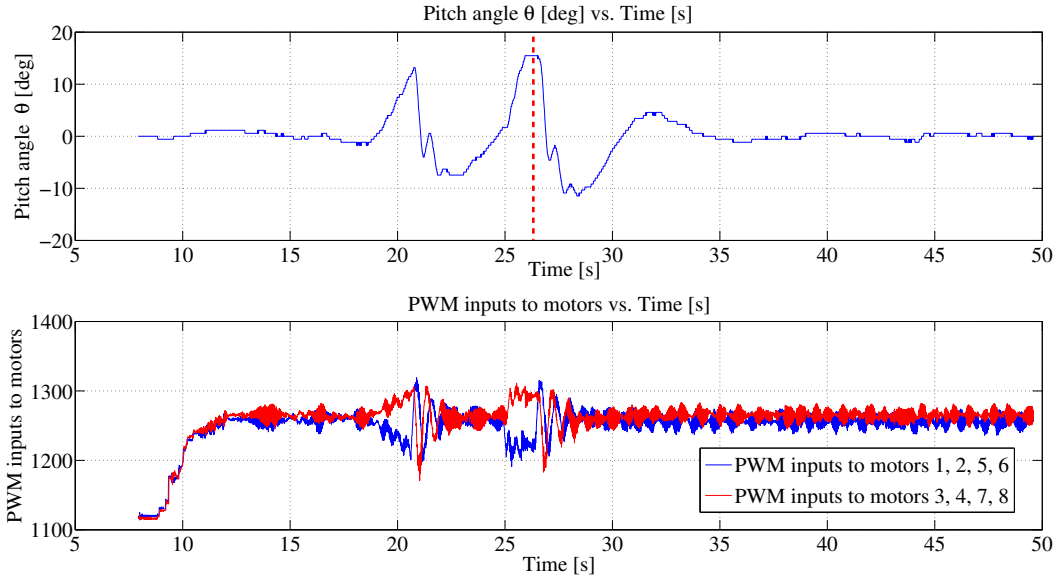

Figure 9.8: Experimental PID controlled pitch output and relevant motors inputs

While Fig. 9.8 refers to the PID regulation of the pitch angle dynamics, Fig. 9.9 refers to the LQ regulation of the yaw angle dynamics. As for the experimental application of the PID approach, the LQ control strategy demonstrated the required capability to

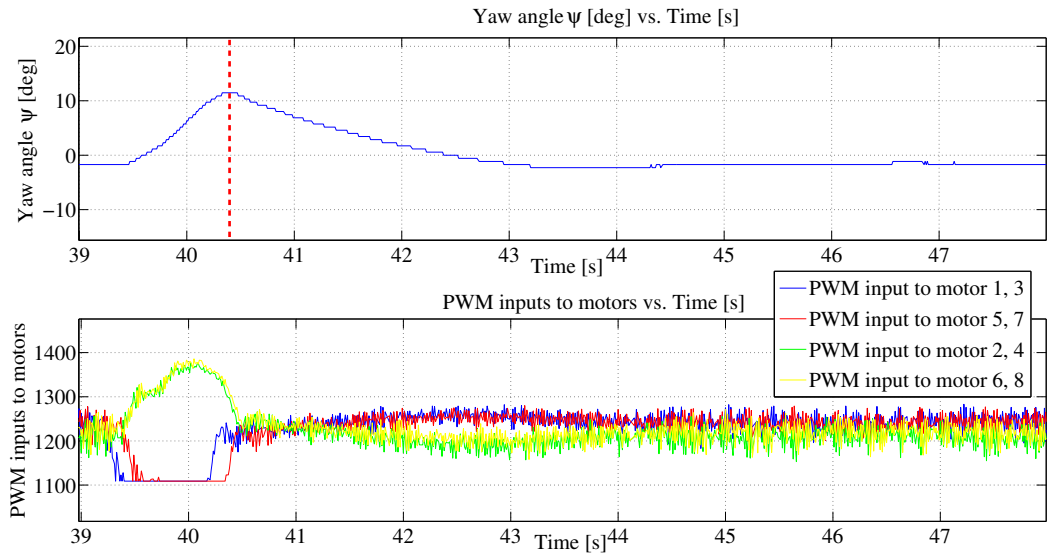

Figure 9.9: Experimental LQ regulated yaw output and relevant motors inputs

stabilize the physical system around the desired zero yaw angle.

The vertical and all the three rotation dynamics tests have been singly conducted, both for the PID and the LQR control architectures, requiring only some minor tunings of the control gains with respect to the same parameters tuned with the Matlab Simulink® simulations.

Finally, the multirotor was tested in a completely free flight. Further minor tunings of the control gains were requested to adequate the control sharing between the human pilot and the automatic controller. This sharing cannot be predefined with certainty, it may be chosen according to the human pilot skills and handling qualities preferences. Fig. 9.10 shows the height and attitude data registered during a free flight while, starting from the hovering condition, the multirotor was commanded to reach the height of 150 [cm] above the ground, keeping all the attitude angles equal the 0 [deg]. Fig. 9.10 shows an acceptable height tracking capability while the attitude angles errors can be bound within the interval [-0.2 0.2] [deg], during the whole interval of the vertical maneuver. This result is valid when applying both the PID and the LQR strategy.

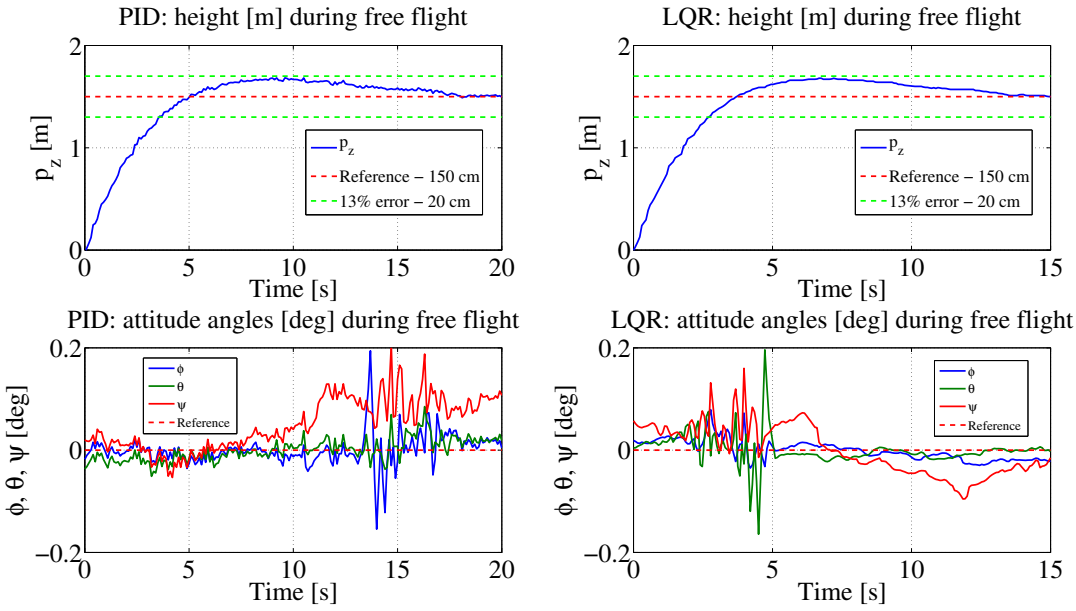

Figure 9.10: PID and LQR controlled height and attitude angles during free flight

10

Neural control

The purpose of this chapter is to provide a quick overview of neural networks and to explain how they can be used in control systems. We introduce the most common neural network architecture, the multilayer perceptron and describe how it can be used for function approximation. The backpropagation algorithm (including its variations) is the principal procedure for training multilayer perceptrons and it is briefly described here. Care must be taken, when training perceptron networks, to ensure that they do not overfit the training data and then fail to generalise well in new situations. Several techniques for improving generalisation are discussed. The chapter also presents the model reference adaptive control. We demonstrate the practical implementation of this controller on the automatic height control of a multirotor system.
Finally various sensors setups are conceived and their implications on the control law are evaluated. For the purposes of this chapter we will look at neural networks as *function approximators*. As shown in Fig. 10.1, we have some unknown function that we wish to approximate. We want to adjust the parameters of the network so that it will produce the same response as the unknown function, if the same input is applied to both systems. For our applications, the unknown function may correspond to a system we are trying to control, in which case the neural network will be the identified plant model. The unknown function could also represent the inverse of a system we are trying to control, in which case the neural network can be used to implement the controller. In the next section we will present the multilayer perceptron neural network and will demonstrate how it can be used as a function approximator.

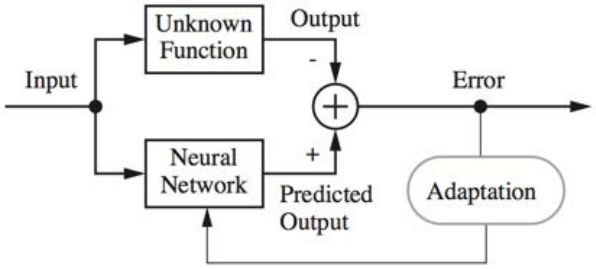

Figure 10.1: Neural network as function approximator

10.1 Multilayer perceptron architecture

10.1.1 Neuron model

The multilayer perceptron neural network is built up of simple components. We will begin with a single-input neuron, which we will then extend to multiple inputs. We will next stack these neurons together to produce layers. Finally, we will cascade the layers together to form the network.

A single-input neuron is shown in Fig. 10.2. The scalar input p is multiplied by the scalar weight w to form wp, one of the terms that is sent to the sum block. The other input, 1, is multiplied by a bias b and then passed to the sum block. The sum block output n, often referred to as the *net input*, goes into a *transfer function* f, which produces the scalar neuron output a.

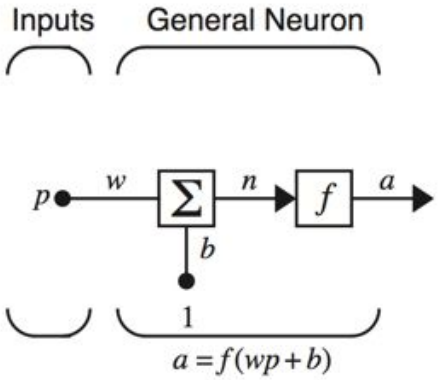

Figure 10.2: Single input neuron

The neuron output is calculated as

$$a = f(wp + b). \tag{10.1}$$

Note that w and b are both adjustable scalar parameters of the neuron. Typically the transfer function is chosen by the designer and then the parameters w and b are adjusted by some learning rule so that the neuron input/output relationship meets some specific goal. The transfer function in Fig. 10.2 may be a linear or a nonlinear function of n. One of the most commonly used functions is the log-sigmoid transfer function, which is shown in Fig. 10.3.

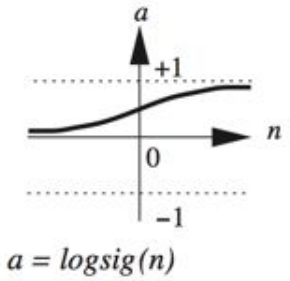

Figure 10.3: Log-sigmoid transfer function

This transfer function takes the input (which may have any value between plus and minus infinity) and squashes the output into the range 0 to 1, according to the expression

$$a = \frac{1}{1 + e^{-n}}. \tag{10.2}$$

The log-sigmoid transfer function is commonly used in multilayer networks that are trained using the backpropagation algorithm, in part because this function is differentiable. Tipically, a neuron has more than one input. A neuron with R inputs is shown in Fig. 10.4. The individual inputs p_1, p_2, ..., p_R are each one weighted by corresponding elements $w_{1,1}$, $w_{1,2}$, ..., $w_{1,R}$ of the *weight matrix* **W**.

The neuron has a bias b, which is summed with the weighted inputs to form the net input n:

$$n = w_{1,1}p_1 + w_{1,2}p_2 + w_{1,R}p_R + b. \tag{10.3}$$

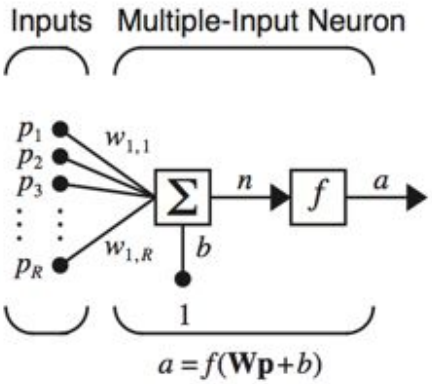

Figure 10.4: Multiple-input neuron

This expression can be written in matrix form as

$$n = \mathbf{W}\mathbf{p} + b, \tag{10.4}$$

where the matrix $\mathbf{W}$ for the single neuron case has only one row.
Now the neuron output can be written as

$$a = f(\mathbf{W}\mathbf{p} + b). \tag{10.5}$$

Fig. 10.5 represents the neuron in matrix form.

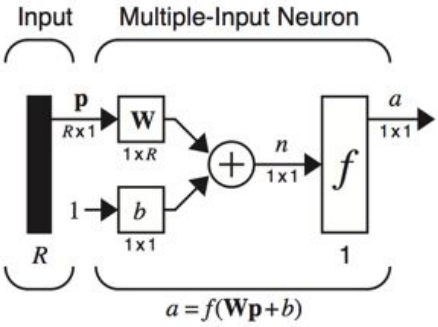

Figure 10.5: Neuron with R inputs, matrix notation

10.1.2 Network architecture

Commonly one neuron, even with many inputs, is not sufficient. We might need five or ten, operating in parallel, in what is called a layer. A single-layer network of S

neurons is shown in Fig. 10.6. Note that each of the R inputs is connected to each of the neurons and that the weight matrix now has S rows. The layer includes the weight matrix $\mathbf{W}$, the sum blocks, the bias vector $\boldsymbol{b}$, the transfer function boxes and the output vector $\boldsymbol{a}$. Some authors refer to the inputs as another layer, but we will not do that here. It is common for the number of inputs to a layer to be different from the number of neurons (i.e., $R \neq S$).

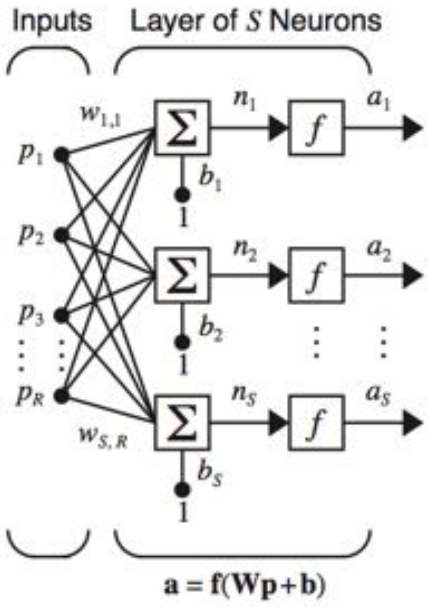

Figure 10.6: Layer of S neurons

The S-neuron, R-input, one-layer network also can be drawn in matrix notation, as shown in Fig. 10.7.

10.1.2.1 Multiple layers of neurons

Now consider a network with several layers. Each layer has its own weight matrix $\mathbf{W}$, its own bias vector $\boldsymbol{b}$, a net input vector $\boldsymbol{n}$ and an output vector $\boldsymbol{a}$. We need to introduce some additional notation to distinguish between these layers. We will use superscripts to identify the layers. Thus, the weight matrix for the first layer is written as $\mathbf{W}^1$ and the weight matrix for the second layer is written as $\mathbf{W}^2$. This notation is used in the three-layer network shown in Fig. 10.8. As shown, there are R inputs,

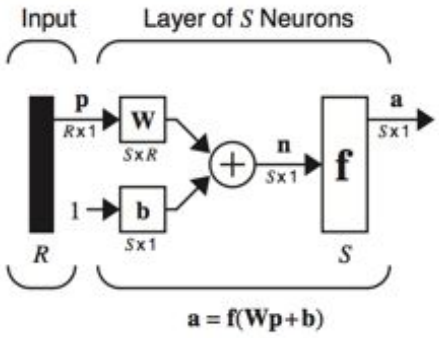

Figure 10.7: Layer of S neurons, matrix notation

S^1 neurons in the first layer, S^2 neurons in the second layer, etc.. As noted, different layers can have different numbers of neurons.

The outputs of layers one and two are the inputs for layers two and three. Thus layer 2 can be viewed as a one-layer network with $R = S^1$ inputs, $S = S^2$ neurons and an S^1 x S^2 weight matrix $\mathbf{W}^2$. The input to layer 2 is $\boldsymbol{a}^1$ and the output is $\boldsymbol{a}^2$. A layer whose output is the network output is called an output layer. The other layers are called hidden layers. The network shown in Fig. 10.8 has an output layer (layer 3) and two hidden layers (layers 1 and 2).

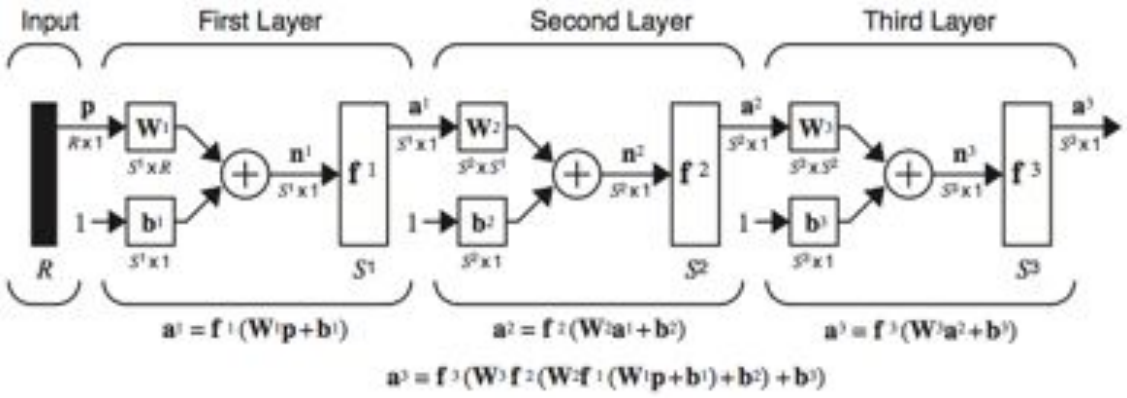

Figure 10.8: Three-layer network

10.2 Approximation capabilities

Two-layer networks, with sigmoid transfer functions in the hidden layer and linear transfer functions in the output layer, are *universal* approximators [49]. A simple example can demonstrate the power of this network for approximation.
Consider the two-layer, 1-2-1 network shown in Fig. 10.9. For this example the transfer function for the first layer is log-sigmoidal and the transfer function for the second layer is linear. In other words

$$f^1(n) = \frac{1}{1+e^{-n}} \quad and \quad f^2(n) = n. \tag{10.6}$$

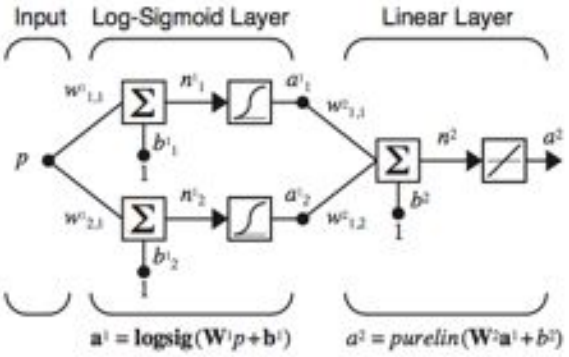

Figure 10.9: Example function approximation network

Suppose that the nominal values of the weights and biases for this network are

$$\begin{aligned} w^1_{1,1} &= 10 \\ w^1_{2,1} &= 10 \\ b^1_1 &= -10 \\ b^1_2 &= 10 \\ w^2_{1,1} &= 1 \\ w^2_{1,2} &= 1 \\ b^2 &= 0. \end{aligned}$$

The network response for these parameters is shown in 10.10, which plots the network output a^2 as the input p is varied over the range [-2, 2]. Notice that the response consists of two steps, one for each of the log-sigmoidal neurons in the first layer. By adjusting the network parameters we can change the shape and location of each step, as we will see in the following discussion. The centers of the steps occur where the net input to a neuron in the first layer is zero:

$$n_1^1 = w_{1,1}^1 p + b_1^1 = 0 \Rightarrow p = -\frac{b_1^1}{w_{1,1}^1} = -\frac{-10}{10} = 1 \tag{10.7}$$

$$n_1^2 = w_{2,1}^1 p + b_2^1 = 0 \Rightarrow p = -\frac{b_2^1}{w_{2,1}^1} = -\frac{10}{10} = -1. \tag{10.8}$$

The steepness of each step can be adjusted by changing the network weights.

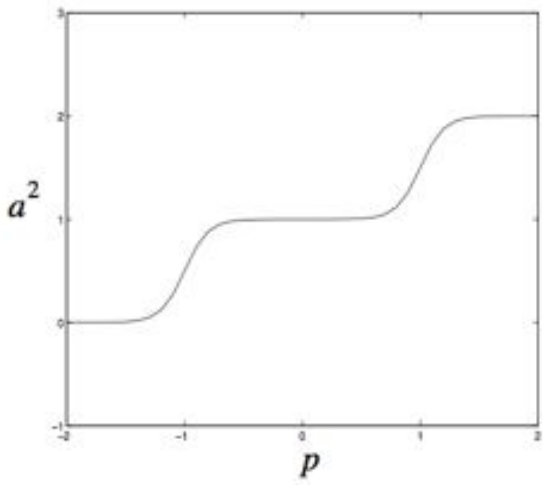

Figure 10.10: Nominal Response of Network of Fig. 10.9

Fig. 10.11 illustrates the effects of parameter changes on the network response. The nominal response is repeated from Fig. 10.10. The other curves correspond to the network response when one parameter at a time is varied over the following ranges:

$$\begin{aligned} -1 \leq & w_{1,1}^2 \leq 1 \\ -1 \leq & w_{1,2}^2 \leq 1 \\ 0 \leq & b_2^1 \leq 20 \\ -1 \leq & b^2 \leq 1. \end{aligned}$$

Fig. 10.11(a) shows how the network biases in the first (hidden) layer can be used to locate the position of the steps. Fig. 10.11(b) and Fig. 10.11(c) illustrate how the weights determine the slope of the steps. The bias in the second (output) layer shifts the entire network response up or down, as can be seen in Fig. 10.11(d).

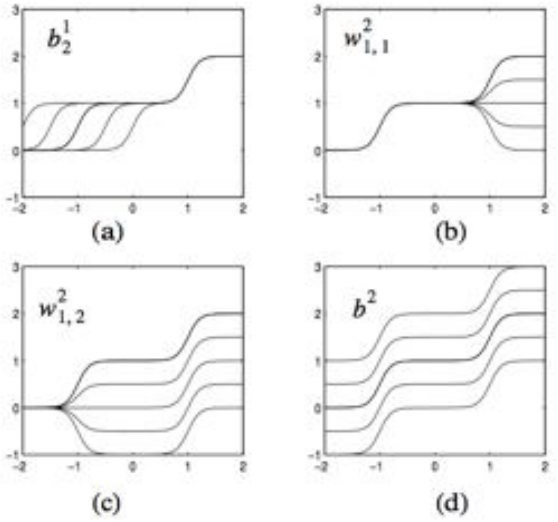

Figure 10.11: Effect of parameter changes on network response

From this example we can see how flexible the multilayer network is. It would appear that we could use such networks to approximate almost any function, if we had a sufficient number of neurons in the hidden layer. In fact, it has been shown that two-layer networks, with sigmoidal transfer functions in the hidden layer and linear transfer functions in the output layer, can approximate virtually *any* function of interest to *any* degree of accuracy, provided sufficiently many hidden units are available. It is beyond the scope of this chapter to provide detailed discussions of approximation theory but there are many papers in the literature that can provide a deeper discussion of this field. In [49], Hornik, Stinchcombe and White present a proof that multilayer perceptron networks are universal approximators. Pinkus gives a more recent review of the approximation capabilities of neural networks in [50]. Niyogi and Girosi, in [51], develop bounds on function approximation error when the network is trained on noisy data.

10.3 Training multilayer networks

Now that we know multilayer networks are universal approximators, the next step is to determine a procedure for selecting the network parameters (weights and biases) that will best approximate a given function. The procedure for selecting the parameters for a given problem is called *training the network*. In this section we will outline a training procedure called backpropagation, which is based on gradient descent (more efficient algorithms than gradient descent exist and are often used in neural network training). As we discussed earlier, for multilayer networks the output of one layer becomes the input to the following layer (see Fig. 10.8). The equations that describe this operation are

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a} + \mathbf{b}^{m+1})$$
$$for \quad m = 0, 1, ..., M-1, \tag{10.9}$$

where M is the number of layers in the network. The neurons in the first layer receive external inputs:

$$\mathbf{a}^0 = \mathbf{p} \tag{10.10}$$

which provides the starting point for Eq. (10.9). The outputs of the neurons in the last layer are considered the network outputs:

$$\mathbf{a} = \mathbf{a}^M. \tag{10.11}$$

10.3.1 Performance index

The backpropagation algorithm for multilayer networks is a gradient descent optimization procedure in which we minimize a mean square error performance index. The algorithm is provided with a set of examples of proper network behaviour:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, ..., \{\mathbf{p}_Q, \mathbf{t}_Q\} \tag{10.12}$$

where $\mathbf{p}_Q$ is an input to the network and $\mathbf{t}_Q$ is the corresponding target output. As each input is applied to the network, the network output is compared to the target.

The algorithm should adjust the network parameters in order to minimize the sum squared error

$$F(\mathbf{x}) = \sum_{q=1}^{Q} e_q^2 = \sum_{q=1}^{Q} (t_q - a_q)^2 \tag{10.13}$$

where $\mathbf{x}$ is a vector containing all network weights and biases. If the network has multiple outputs this generalizes to

$$F(\mathbf{x}) = \sum_{q=1}^{Q} \mathbf{e}_q^T \mathbf{e}_q = \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (\mathbf{t}_q - \mathbf{a}_q). \tag{10.14}$$

Using a stochastic approximation, we will replace the sum squared error by the error on the latest target:

$$\begin{aligned} \hat{F}(\mathbf{x}) = (\mathbf{t}(k) - \mathbf{a}(k))^T (\mathbf{t}(k) - \mathbf{a}(k)) = \\ = \mathbf{e}^T(k)\mathbf{e}(k) \end{aligned} \tag{10.15}$$

where the expectation of the squared error has been replaced by the squared error at iteration k.

The steepest descent algorithm for the approximate mean square error is

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^m} \tag{10.16}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \hat{F}}{\partial b_i^m} \tag{10.17}$$

where α is the learning rate.

10.3.2 Chain rule

For a single-layer linear network, these partial derivatives in Eq. (10.16) and Eq. (10.17) are conveniently computed, since the error can be written as an explicit linear function of the network weights. For the multilayer network, the error is not an explicit function of the weights in the hidden layers, therefore these derivatives are not computed so easily.

Because the error is an indirect function of the weights in the hidden layers, we will use the chain rule of calculus to calculate the derivatives in Eq. (10.16) and Eq. (10.17):

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \tag{10.18}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m}. \tag{10.19}$$

The second term in each of these equations can be easily computed, since the net input to layer m is an explicit function of the weights and bias in that layer:

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m. \tag{10.20}$$

Therefore

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}$$

$$\frac{\partial n_i^m}{\partial b_i^m} = 1. \tag{10.21}$$

If we now define

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m} \tag{10.22}$$

(the sensitivity of $\hat{F}$ to changes in the i^{th} element of the net input at layer m), then Eq. (10.18) and Eq. (10.19) can be simplified to

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \tag{10.23}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m. \tag{10.24}$$

We can now express the approximate steepest descent algorithm as

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1} \tag{10.25}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m. \tag{10.26}$$

In matrix form this becomes:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \tag{10.27}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m \tag{10.28}$$

where the individual elements of $\mathbf{s}^m$ are given by Eq. (10.22).

10.3.3 Backpropagating the sensitivities

It now remains for us to compute the sensitivities $\mathbf{s}^m$, which requires another application of the chain rule. It is this process that gives us the term *backpropagation*, because it describes a recurrence relationship in which the sensitivity at layer m is computed from the sensitivity at layer $m+1$:

$$\mathbf{s}^M = -2\dot{F}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a}) \tag{10.29}$$

$$\begin{aligned} \mathbf{s}^m &= \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1}, \\ m &= M-1,...,2,1 \end{aligned} \tag{10.30}$$

where

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \dots & 0 \\ 0 & \dot{f}^m(n_2^m) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \dot{f}^m(n_{s^m}^m) \end{bmatrix} \tag{10.31}$$

See reference [52], chapter 11, for a derivation of this result.

10.3.4 Variation of backpropagation

In some ways it is unfortunate that the algorithm we usually refer to as backpropagation, given by Eq. (10.27) and Eq. (10.28), is in fact simply a steepest descent algorithm. There are many other optimization algorithms that can use the backpropagation procedure, in which derivatives are processed from the last layer of the network to the first (as given in Eq. (10.30)). For example, conjugate gradient and quasi-Newton algorithms are generally more efficient than steepest descent algorithms and yet they can use the same backpropagation procedure to compute the necessary derivatives. The Levenberg-Marquardt algorithm is very efficient for training small to medium-size networks and it uses a backpropagation procedure that is very similar to the one given by Eq. (10.30).

10.3.5 Generalization (interpolation & extrapolation)

We now know that multilayer networks are universal approximators but we have not discussed how to select the number of neurons and the number of layers necessary to achieve an accurate approximation in a given problem. Moreover we have not discussed how the training data set should be selected. The trick is to use enough neurons to capture the complexity of the underlying function without having the network overfit the training data, in which case it will not generalize to new situations. We also need to have sufficient training data to adequately represent the underlying function. To illustrate the problems we can have in network training, consider the following general example. Assume that the training data is generated by the following equation:

$$\mathbf{t}_q = \mathbf{g}(\mathbf{p}_q) + \mathbf{e}_q \tag{10.32}$$

where $\mathbf{p}_q$ is the system input, $\mathbf{g}(.)$ is the underlying function we wish to approximate, $\mathbf{e}_q$ is measurement noise and $\mathbf{t}_q$ is the system output (network target). Fig. 10.12 shows an example of the underlying function $\mathbf{g}(.)$ (thick line), training data target values $\mathbf{t}_q$ (circles) and total trained network response (thin line). The two graphs of Fig. 10.12 and Fig. 10.13 represent different training strategies.

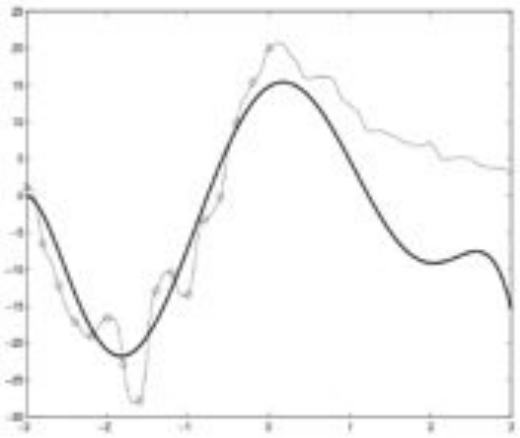

Figure 10.12: Example of overfitting

In the example shown in Fig. 10.12, a large network was trained to minimize squared error (Eq. (10.13)) over the 15 points in the training set. We can see that the network response exactly matches the target values for each training point. However, the total

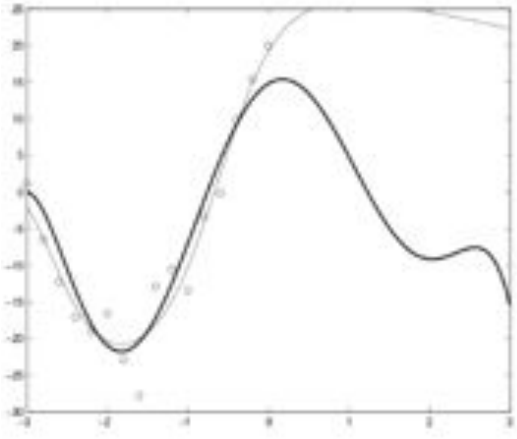

Figure 10.13: Example of a good fit

network response has failed to capture the underlying function. There are two major problems. First, the network has overfit on the training data. The network response is too complex, because the network has more than enough independent parameters and they have not been constrained in any way. The second problem is that there is no training data for values of p greater than 0. Neural networks (and other nonlinear black box techniques) cannot be expected to *extrapolate* accurately. If the network receives an input that is outside of the range covered in the training data, then the network response will always be suspect.

While there is little we can do to improve the network performance outside the range of the training data, we can improve its ability to interpolate between data points. Improved generalization can be obtained through a variety of techniques. In one method, called early stopping, we place a portion of the training data into a validation data set. The performance of the network on the validation set is monitored during training. During the early stages of training the validation error will come down. When overfitting begins, the validation error will begin to increase and at this point the training is stopped.

Another technique to improve network generalization is called regularization. With this method the performance index is modified to include a term which penalizes network complexity. The most common penalty term is the sum of squares of the network

weights:

$$F(\mathbf{x}) = \sum_{q=1}^{Q} \mathbf{e}_q^T \mathbf{e}_q + \rho \sum (w_{i,j}^k)^2. \tag{10.33}$$

This performance index forces the weights to be small, which produces a smoother network response. The trick with this method is to choose the correct regularization parameter ρ. If the value is too large, then the network response will be too smooth and will not accurately approximate the underlying function. If the value is too small, then the network will overfit. There are a number of methods for selecting the optimal ρ. One of the most successful is Bayesian regularization.

Fig. 10.13 shows the network response when the network is trained with Bayesian regularization. Notice that the network response no longer exactly matches the training data points, but the overall network response more closely matches the underlying function over the range of the training data. In the next section we will describe how multilayer networks can be used in neural control applications.

10.4 Control systems applications

Multilayer neural networks have been applied successfully in the identification and control of dynamic systems. Rather than attempt to survey the many ways in which multilayer networks have been used in control systems, we will concentrate on one typical neural network controller, the *model reference control*. This controller is representative of the ways in which multilayer networks are used in control systems. As with most neural controllers, it is based on standard linear control architectures.

There are typically two steps involved when using neural networks for control:

1. *system identification*,
2. *control design*.

In the system identification stage, we develop a neural network model of the plant that we want to control.

In the control design stage, we use the neural network plant model to design (or train) the controller. The next subsection of this chapter discuss the model reference control and will describe how it can be applied in practice.

10.4.1 Model reference control

10.4.1.1 System identification

The first stage of model reference control is to train a neural network to represent the forward dynamics of the plant. The prediction error between the plant output and the neural network output is used as the neural network training signal. The process is represented by Fig. 10.14.

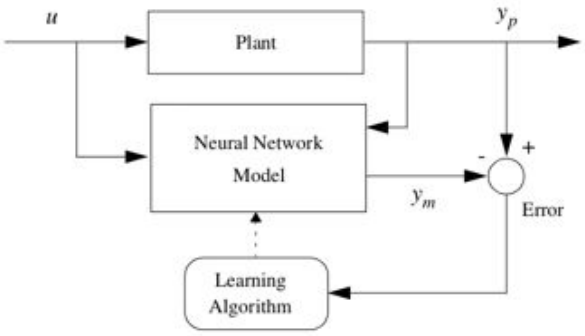

Figure 10.14: Plant identification

One standard model that can be used for nonlinear identification is the Nonlinear Autoregressive-Moving Average (NARMA) model - detailed in [53] -:

$$\begin{aligned} y(k+d) = h[&y(k), y(k-1), ... \\ &..., y(k-n+1), u(k), u(k-1), ... \\ &..., u(k-m+1)] \end{aligned} \tag{10.34}$$

where $u(k)$ is the system input, $y(k)$ is the system output and d is the system delay. For the identification phase, we train a neural network to approximate the nonlinear function h. The structure of the neural network plant model is given in Fig. 10.15, where the blocks labeled TDL are tapped delay lines that store previous values of the input signal. The equation for the plant model is given by

$$\begin{aligned} y_m(k+1) = \hat{h}[&y_p(k), ..., y_p(k-n+1), u(k), ..., \\ &..., u(k-m+1); \mathbf{x}], \end{aligned} \tag{10.35}$$

where $\hat{h}[.; \mathbf{x}]$ is the function implemented by the neural network and $\mathbf{x}$ it the vector containing all network weights and biases. We have modified our previous notation

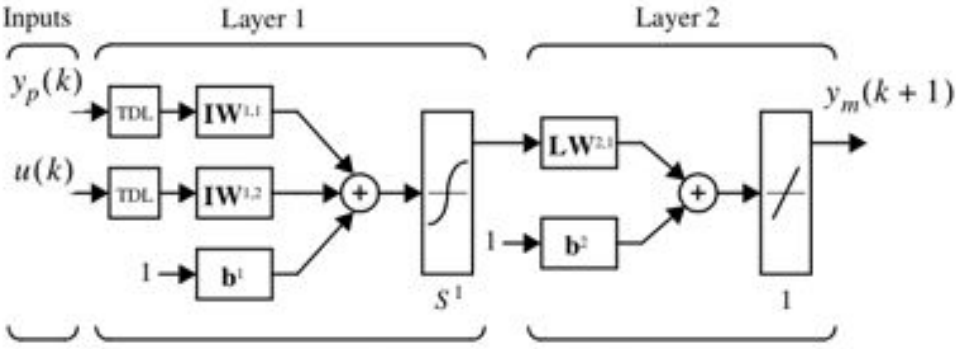

Figure 10.15: Neural network plant model

here, to allow more than one input into the network. $\mathbf{IW}^{i,j}$ is a weight matrix from input number j to layer number i. $\mathbf{LW}^{i,j}$ is a weight matrix from layer number j to layer number i.

Although there are delays in this network, they occur only at the network input and the network contains no feedback loops. For these reasons, the neural network plant model can be trained using the backpropagation methods for feedforward networks described in the first part of this chapter. It is important that the training data cover the entire range of plant operation because we know from previous discussions that nonlinear neural networks do not extrapolate accurately. The input to this network is an $(n_y + n_u)$-dimensional vector of previous plant outputs and inputs. It is this space that must be covered adequately by the training data.

10.4.1.2 Neural network controller

Model reference control architecture uses two neural networks: a controller network and a plant model network, as shown in Fig. 10.16. The plant model is identified first and then the controller is trained so that the plant output follows the reference model output. The online computation of the model reference controller is minimal. However the model reference architecture requires a separate neural network controller to be trained, in addition to the neural network plant model. The controller training is computationally expensive since it requires the use of dynamic backpropagation.

On the positive side, model reference control applies to a large class of plants which requires the plant to be approximated by a companion form model.

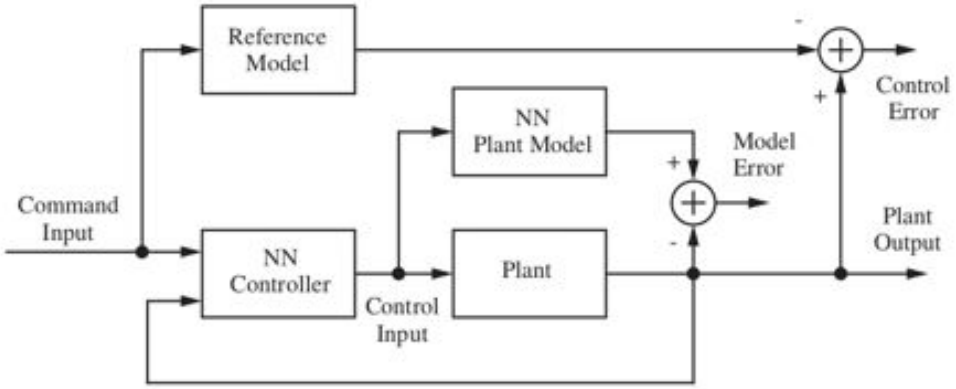

Figure 10.16: Model reference control architecture

Fig. 10.17 shows the details of the neural network plant model and the neural network controller. There are three sets of controller inputs:

1. delayed reference inputs,
2. delayed controller outputs (plant inputs),
3. delayed plant outputs.

For each of these inputs, we select the number of delayed values to use. Typically, the number of delays increases with the order of the plant. There are two sets of inputs to the neural network plant model: delayed controller outputs and delayed plant outputs. The plant identification process for model reference control uses the NARMA model given by Eq. (10.34). It is clear from Fig. 10.17 that the model reference control structure is a recurrent (feedback) network. This type of network is more difficult to train than the feedforward networks that were discussed in the first half of this chapter and that are used for plant identification. Suppose that we use the same gradient descent algorithm, Eq. (10.16), that is used in the standard backpropagation algorithm. The problem with recurrent networks is that when we try to find the equivalent of Eq. (10.23) (gradient calculation) we note that the weights and biases have two different effects on the network output. The first is the direct effect, which is accounted for by Eq. (10.23). The second is an indirect effect, since some of the inputs to the network, such as $u(t-1)$, are also functions of the weights and biases. To account for this indirect effect we must use dynamic backpropagation to compute the gradients for recurrent

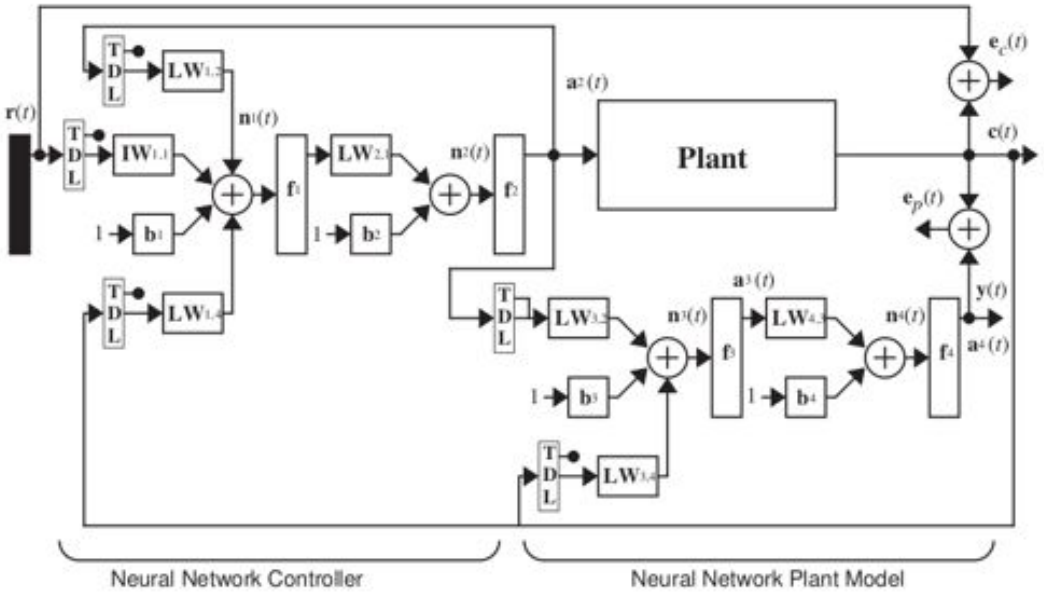

Figure 10.17: Detailed model reference control structure

networks. A detailed description of dynamic backpropagation is anyway beyond the scope of this chapter.

In addition to the difficulty in computing the gradients for recurrent networks, the error surfaces for these networks pose special difficulties for gradient-based optimization algorithms. Gradient-based training procedures that are well suited to training recurrent networks are discussed in reference [54]. The data used to train the model reference controller is generated while applying a random reference signal which consists of a series of pulses of random amplitude and duration. This data can be generated without running the plant, but using the neural network model output in place of the plant output.

10.4.2 Application: vertical position control of a multirotor

The potentiality of the model reference control technique will be demonstrated controlling the vertical dynamics of a multirotor. Referring to Eq. (5.22), the equation of motion that describes the vertical dynamics of a multirotor in the vehicle-1 frame is

$$\ddot{p}_z = g - \cos\phi\cos\theta \cdot \frac{F}{m} \tag{10.36}$$

where

- $\ddot{p}_z$ is the multirotor centre of gravity acceleration in the vehicle-1 frame,
- g is the gravity acceleration,
- ϕ and θ are the roll and pitch angles,
- F is the total thrust force acting on the multirotor,
- m is the multirotor mass.

10.4.2.1 Plant identification

The system was sampled at intervals of 0.05 seconds. To identify the system, input pulses with intervals between 0.01 and 5 seconds and amplitude between -11 [N] and 20 [N] have been used. The neural network plant model used two delayed values of thrust ($m = 2$), two delayed values of multirotor position ($n = 2$) as input to the network and 15 neurons were used in the hidden layer (a 5-15-1 network). Fig. 10.18 shows the Matlab tool that, opportunely tuned, allows defining the plant neural network.

Figure 10.18: Matlab graphical user interface for plant identification and training

10.4.2.2 Neural network controller

We define as objective for the control system that the multirotor vertical position tracks the reference model

$$\ddot{y}_r = -6\dot{y}_r - 9y_r + 9r \tag{10.37}$$

where y_r, is the output of the reference model and r is the input reference signal. For the controller network, it has been used a 5-13-1 architecture. The inputs to the controller consisted of two delayed reference inputs, two delayed plant outputs and one delayed controller output. The controller was trained using a BFGS (Broyden, Fletcher, Goldfarb and Shanno) quasi-Newton algorithm, with dynamic backpropagation used to calculate the gradients. Fig. 10.19 shows the Matlab tool that, opportunely tuned, allows to define the neural network controller.

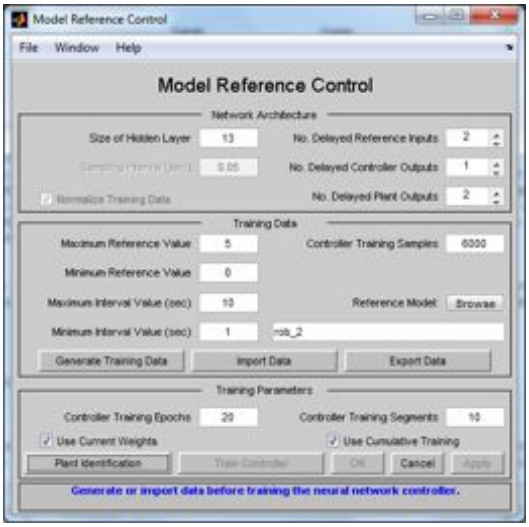

Figure 10.19: Matlab graphical user interface for the neural network controller identification and training

The graph of Fig. 10.20 shows the reference height signal and the multirotor position along the z axis of the vehicle-1 frame, using the trained model reference controller. The system is able to follow the reference and the control actions (shown in Fig. 10.21) are smooth. At certain set points there is some steady state error. This error could be reduced by adding more training data at those steady state conditions where the error is largest. The problem can occur in the plant model or in the controller network.

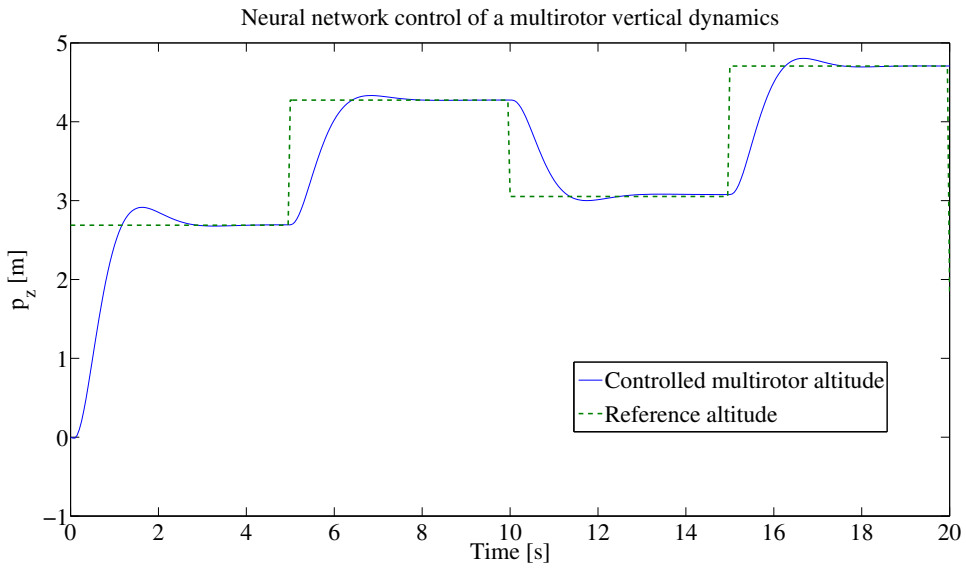

Figure 10.20: Neural network controlled vertical position

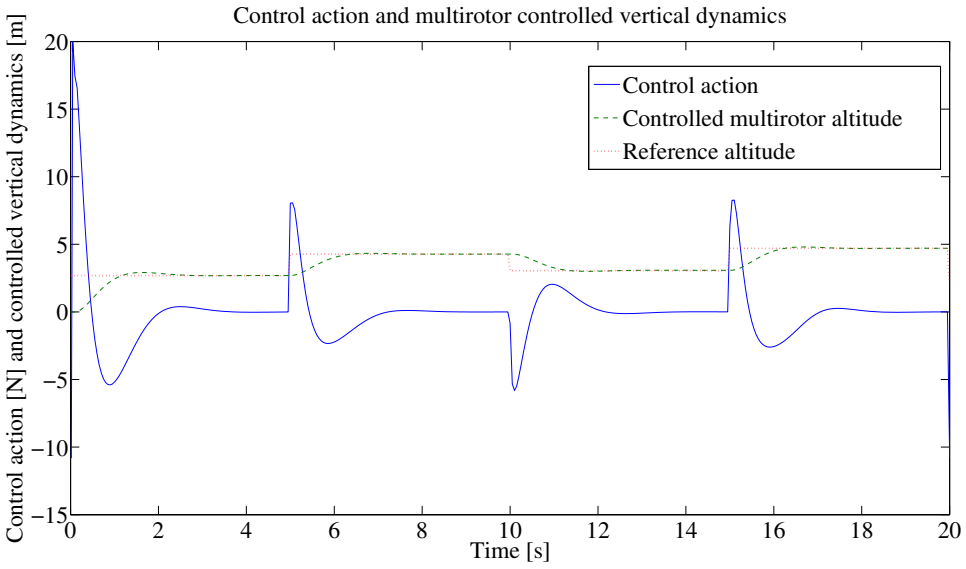

Figure 10.21: Control action and multirotor controlled vertical dynamics

10.5 Distance sensor embodiment

In order to allow the implementation of an automatic *sense* & *avoid* system a suitable sensor embodiment has been conceived. This concept is shown in Fig. 10.22.

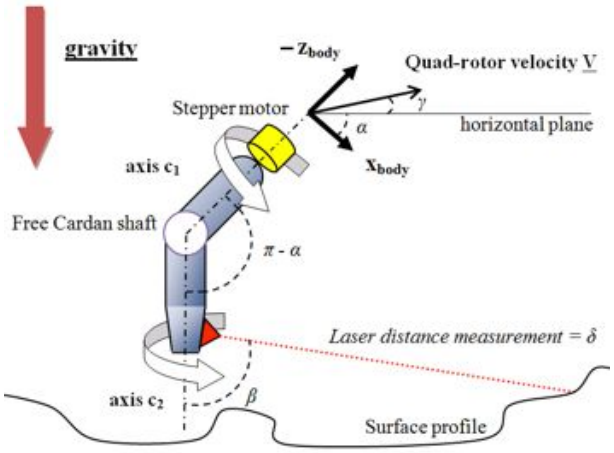

Figure 10.22: Distance sensor embodiment

It is mainly based on the Cardan shaft (see Fig. 10.23) mechanical property.

Figure 10.23: Cardan shaft

One arm of the shaft is connected to a stepper motor that is linked to the multirotor structure. This arm axis is named c_1 in the Fig. 10.22. This axis is aligned with the z_{body} axis. The stepper motor is inserted in order to allow rotations of the laser distance sensor around the c_2 axis. For the mechanical properties of the Cardan shaft, in fact, a certain angular velocity around the c_1 axis will be equally transmitted to the second arm, i.e. a rotation around c_2 will take place. The joint conjunction between

the first and the second arm is intended to be left free, i.e. without an active motor control, since, due to the gravity force it will constantly (with certain oscillations to be passively damped) maintain its vertical position.
Control requisite is then derived from the obstacles perception with the above explained sensor setup. The measured distance of the multirotor from the closest obstacle is

$$\Delta = \delta_{max} \sin \beta \tag{10.38}$$

where δ_{max} is defined as the maximum distance that can be measured by the sensor. With the aim to define the required control reaction rise time of our *sense & avoid* system a *time to collision* t_c value has to be defined. This time is approximately found as follow

$$t_c \approx \frac{\Delta}{V \cos \gamma}. \tag{10.39}$$

For example, if a multirotor is flying at

$$V cos(\gamma) = 10 m/s$$

and the maximum distance that can be measured by the sensor is

$$\delta_{max} = 30 m,$$

if β is equal to 80° then

$$t_c \approx \frac{30 \cdot sin(80°)}{10} = 2.95 s.$$

This value sets an important constraint for our controller performances. Having a rise time minor than 2.95 s, the neural network controller previously designed could sustain the above presented condition but for more restrictive cases, as for example if the speed increased above 10 m/s or for lower quality sensors, the previous neural controller could not be satisfying anymore, if we maintained the same network design.

10.5.1 Using only one forward looking distance sensor

The case in which only one sensor is used for height tracking has been simulated using Matlab Simulink® environment. The idea behind the simulation is that the unique sensor is mounted in a way that it allows a perception of the forward surface variation, t_c seconds before that the multirotor arrives on it. Simulation results are shown in Fig.

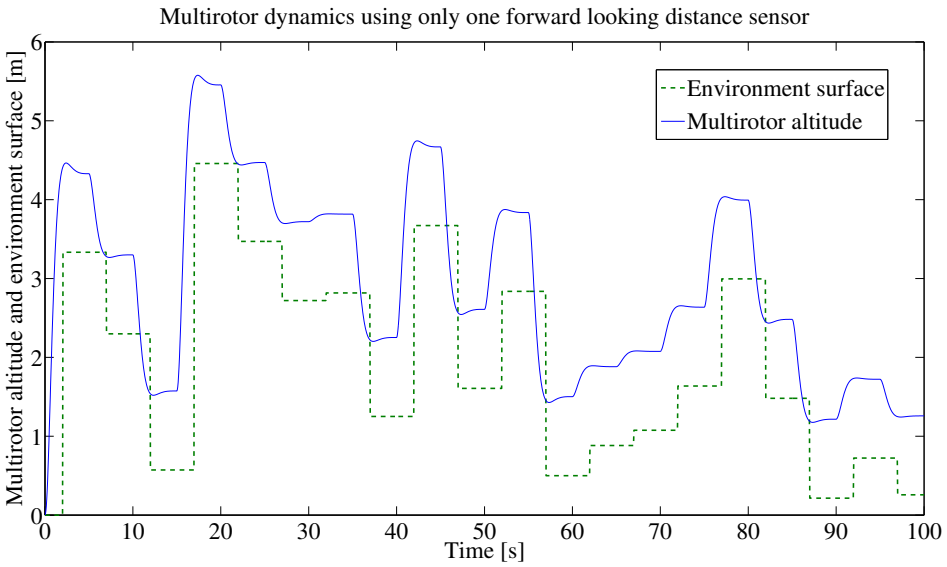

Figure 10.24: Multirotor dynamics using only one forward looking distance sensor

10.24. In this case t_c is equal to 2 s and the multirotor is required to fly 1 m above the surface. As it could be foreseeable, the multirotor has a good approach to the obstacle but while it flies beyond, it is no more able to see the obstacle under itself and it crashes against the surface.

For the sake of simplicity, lasers shadowing has not been accounted for. This because the simulation was conceived with the aim to highlight the not sufficient capability of this sensor setup for obstacles avoidance.

10.5.2 A three sensors setup

In order to solve the issue explained in the precedent paragraph a three sensors setup is proposed. This sensor embodiment is presented in Fig. 10.25. To be noted that the normal vector to the sensor "1", namely $z_{sensor1}$, has to be kept constantly aligned with the gravity vector. This target could be pursued following a passively controlled *cardan shaft* setup or using an actively controlled platform able to move contrasting with the pitch and roll multirotor motion. This requires the development of a specific control law. With the aim to have a draft sketch of what could be the multirotor vertical dynamics the simulation shown in Fig. 10.26 has been realized. This simulation shows a

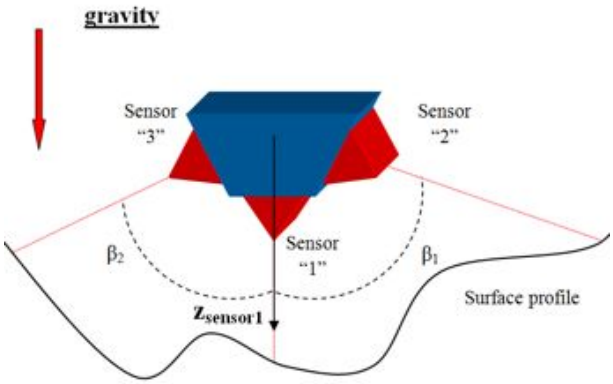

Figure 10.25: A three sensors setup

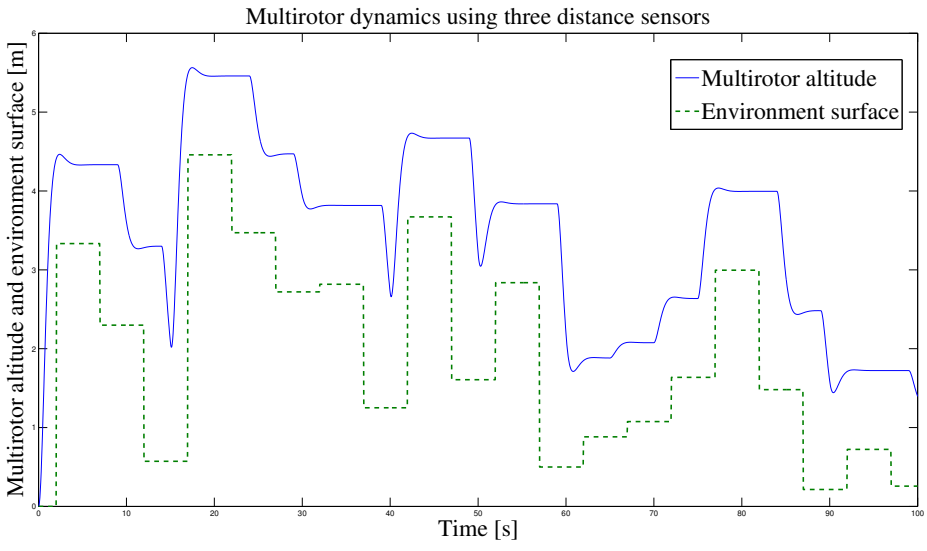

Figure 10.26: Multirotor dynamics using three distance sensors

comforting behaviour of the multirotor vertical position, giving the idea that this is the right way to follow. The reader has to be aware of the fact that the simulation presented in Fig. 10.26 does not account for the sensors shadowing. Further investigations could be done improving this geometrical model.

10.5.2.1 Limits of a three sensors setup

A three sensors setup like the one previously shown could be suited when the multirotor dynamics is bounded in the sensors plane. For this reason it is necessary to build a proper sensors platform able to rotate around the $z_{sensor1}$ axis. This rotation angle could be derived from the pitch and roll angle, following the equations of the dynamics of the multirotor system. This clarifies the necessity for another specific control law with the aim to keep, automatically, the three sensors in the motion plane.

11

Conclusions and future work

This thesis presented the mathematical modelling, validation and physical testing of two different control algorithms applied to the height and attitude regulation of a multirotor. To this end a nonlinear mathematical model was derived to reproduce the multirotor dynamics and two different linear control laws were applied. Several nonlinear simulations validated as successful the chosen control strategies and two prototypes were built to evaluate them experimentally. Single degree of freedom indoor tests demonstrated that all the regulated states followed the target with satisfactory time responses and control actions demands and these results have been confirmed by successful outdoor flights. More innovative control techniques have been investigated with special focus on the neural networks based one.

None of the investigated techniques can be considered superior to the others as a general rule. None of them requires an excessive online computation and therefore they are all suitable for multirotor attitude control applications. The PID technique is for sure the easiest, able to guarantee an acceptable result without an excessive designing effort. When there are no safety implications, PID gains can also be tuned in real time on the physical system. These characteristics make the PID technique the most widely diffused control approach. Also the LQR technique is characterised by simplicity but it requires a better modelling of the dynamical system and it is not as intuitive as the PID technique. The advantage can be found, instead, in the possibility to calibrate the weighing matrices to penalise the control energy demand or the tracking error. The neural networks based control is surely characterised by an inherent complexity. This

is due to the need for a very coherent model of the system or for experimental data to obtain an effective training of the controller. The second drawback is linked to the poor performance of the neural control outside the range of training. This is difficult to be accepted by the certification authorities responsible for the airworthiness of the air vehicles. The last issue is related to the difficulty in debugging the neural networks, since the calculated weights and biases are not directly related to physical quantities and therefore isolating possible errors is not straightforward. The advantage is instead in the theoretical capability to approximate any kind of function, even nonlinear ones. However the distinguishing ability of the neural networks is their learning capability, factor that opens the perspective to learning controllers that could be trained in real time by human users or by other trained controllers. To reach this objective a suitable, generally very large, database should be created. This database has to cover the widest possible range of input commands and output dynamics inside the flight envelope. The research in this field could be a natural progression of the studies presented in this thesis. Fault-tolerant control systems represent another interesting point that could be investigated as continuation of this research. Here this concept was introduced and a possible application was experimented but some aspects have not been treated as, for example, the automatic fault detection and the mapping of all the possible failure modes with the relevant control strategies. It is indisputable that a systematic solution to this issue will play a key role in the acceptance of the UAVs in the public air space.

References

[1] Anthony R.S. Bramwell, David Balmford, and George Done. *Bramwell's helicopter dynamics*. Butterworth-Heinemann, 2001. v, 2

[2] J. Gordon Leishman. *Principles of helicopter aerodynamics*. Cambridge University Press, 2006. vi, 5, 43

[3] Robert P. Harper and George E. Cooper. **Handling qualities and pilot evaluation**. *Journal of Guidance, Control, and Dynamics*, **9**(5):515–529, 1986. vii, 94, 95, 96

[4] I. Golightly and Dewi Jones. **Visual control of an unmanned aerial vehicle for power line inspection**. In *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, pages 288–295, 2005. 1

[5] S.R. Herwitz, L.F. Johnson, S.E. Dunagan, R.G. Higgins, D.V. Sullivan, J. Zheng, B.M. Lobitz, J.G. Leung, B.A. Gallmeyer, M. Aoyagi, et al. **Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support**. *Computers and Electronics in Agriculture*, **44**(1):49–61, 2004. 1

[6] Samir Bouabdallah, Pierpaolo Murrieri, and Roland Siegwart. **Towards autonomous indoor micro VTOL**. *Autonomous Robots*, **18**(2):171–183, 2005. 1

[7] Paul Edward Ian Pounds. *Design, construction and control of a large quadrotor micro air vehicle*. Australian National University, 2007. 1

[8] Michael J. Armitage. *Unmanned aircraft*, **3**. Brassey's Defence Publishers London, 1988. 3, 4

[9] A. Parsch. **Directory of US Military Rockets and Missiles, Appendix 4: Undesignated Vehicles**, 2005. 3

[10] **Gyrodyne Helicopter Historical Foundation, http://www.gyrodynehelicopters.com/** [online, cited October 2013]. 4

[11] G. Goebel. **Cruise Missiles, http://www.vectorsite.net/twcruz.html** [online, cited October 2013]. 5

[12] **MEMS Accelerometer, http://www.silicondesigns.com/tech.html** [online, cited October 2013]. 5, 57

[13] **The Bréguet-Richet Quad-Rotor Helicopter of 1907, http://terpconnect.umd.edu/ leishman/Aero/Breguet.pdf** [online, cited October 2013]. 5

[14] Jim Winchester. *X-planes and Prototypes*. Amber Books, London, United Kingdom, 2005. 5, 7

[15] J. Borenstein. **The Hoverbot - An Electrically Powered Flying Robot, https://ftp.eecs.umich.edu/people/johannb/paper99.pdf** [online, cited October 2013]. 7

[16] Ilan Kroo, Fritz Prinz, Michael Shantz, Peter Kunz, Gary Fay, Shelley Cheng, Tibor Fabian, and Chad Partridge. **The Mesicopter: A Miniature Rotorcraft Concept - Phase II Interim Report**, 2000. 8

[17] **Draganfly Official website, http://www.draganfly.com/** [online, cited October 2013]. 8

[18] N. Guenard, T. Hamel, and V. Moreau. **Dynamic modeling and intuitive control strategy for an "X4-flyer"**. In *International Conference on Control and Automation, 2005*, **1**, pages 141–146. IEEE, 2005. 9

[19] Nicolas Guenard, Tarek Hamel, and Robert Mahony. **A practical visual servo control for an unmanned aerial vehicle**. *Robotics, IEEE Transactions on*, **24**(2):331–340, 2008. 9

[20] Odile Bourquardez, Robert Mahony, Nicolas Guenard, François Chaumette, Tarek Hamel, Laurent Eck, et al. **Kinematic visual servo control of a quadrotor aerial vehicle**. *Institut de Recherche en Informatique et Systémes Aléatoires, Rennes Cedex, France*, 2007. 9

[21] Samir Bouabdallah and Roland Siegwart. **Towards intelligent miniature flying robots**. In *Field and Service Robotics*, pages 429–440. Springer, 2006. 9

[22] Samir Bouabdallah, Andre Noth, and Roland Siegwart. **PID vs LQ control techniques applied to an indoor micro quadrotor**. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, **3**, pages 2451–2456. IEEE, 2004. 10

[23] Samir Bouabdallah and Roland Siegwart. **Backstepping and sliding-mode techniques applied to an indoor micro quadrotor**. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2247–2252. IEEE, 2005. 10

[24] A. Tayebi and S. McGilvray. **Attitude stabilization of a four-rotor aerial robot**. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, **2**, pages 1216–1221. IEEE, 2004. 10

[25] Abdelhamid Tayebi and Stephen McGilvray. **Attitude stabilization of a VTOL quadrotor aircraft**. *IEEE Transactions on Control Systems Technology*, **14**(3):562–571, 2006. 10

[26] Gabe Hoffmann, Dev Gorur Rajnarayan, Steven L. Waslander, David Dostal, Jung Soon Jang, and Claire J. Tomlin. **The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)**. In *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, **2**, pages 12–E. IEEE, 2004. 10

[27] Steven Lake Waslander, Gabriel M. Hoffmann, Jung Soon Jang, and Claire J. Tomlin. **Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning**. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3712–3717. IEEE, 2005. 10

[28] Glenn P. Tournier, Mario Valenti, Jonathan P. How, and Eric Feron. **Estimation and control of a quadrotor vehicle using monocular vision and moire patterns**. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, pages 21–24, 2006. 10

[29] S. Park, D.H. Won, M.S. Kang, T.J. Kim, H.G. Lee, and S.J. Kwon. **RIC (robust internal-loop compensator) based flight control of a quad-rotor type UAV**. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3542–3547. IEEE, 2005. 10

[30] Pedro Castillo, Rogelio Lozano, and Alejandro Dzul. **Stabilization of a mini-rotorcraft having four rotors**. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, **3**, pages 2693–2698. IEEE, 2004. 11

[31] S. Salazar-Cruz, A. Palomino, and R. Lozano. **Trajectory tracking for a four rotor mini-aircraft**. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 2505–2510. IEEE, 2005. 11

[32] Pedro Castillo, Alejandro Dzul, and Rogelio Lozano. **Real-time stabilization and tracking of a four-rotor mini rotorcraft**. *Control Systems Technology, IEEE Transactions on*, **12**(4):510–516, 2004. 11

[33] Hugo Romero, Ryad Benosman, and Rogelio Lozano. **Stabilization and location of a four rotor helicopter applying vision**. In *American Control Conference, 2006*. IEEE, 2006. 11

[34] Erdinc Altüg, James P. Ostrowski, and Robert Mahony. **Control of a quadrotor helicopter using visual feedback**. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, **1**, pages 72–77. IEEE, 2002. 11

[35] Erdinc Altüg, 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*, **3**, pages 4294–4299. IEEE, 2003. 11

[36] Brian L. Stevens and Frank L. Lewis. *Aircraft Control and Simulation*. Wiley-Interscience, October 2003. 18, 25, 58

[37] Randal W. Beard and Timothy W. McLain. *Small unmanned aircraft: Theory and practice*. Princeton University Press, 2012. 27

[38] F. Rinaldi, F. Quagliotti, and S. Chiesa. **Design of an UAV controller based on the open source platform Arduino**. *Proceedings of 3^{rd} CEAS (Council of the European Aerospace Societies), Air&Space Conference, 21^{st} AIDAA Congress, Venice (Italy)*, **62**(1):4–10, October 2011. ISBN: 978-88-96427-18-7. 34

[39] Colin P. Coleman et al. *A survey of theoretical and experimental coaxial rotor aerodynamic research*, **3675**. National Aeronautics and Space Administration, Ames Research Center, 1997. 41, 42

[40] Peter Rowland Payne. *Helicopter dynamics and aerodynamics*. Macmillan, 1959. 41

[41] Robert D. Harrington. **Full-scale-tunnel investigation of the static-thrust performance of a coaxial helicopter rotor**. Technical report, DTIC Document, 1951. 42

[42] Marion K. Taylor. *A balsa-dust technique for air-flow visualization and its application to flow through model helicopter rotors in static thrust*. National Advisory Committee for Aeronautics, 1950. 42

[43] Richard C. Dingeldein. *Wind-tunnel studies of the performance of multirotor configurations*. National Advisory Committee for Aeronautics, 1954. 44

[44] M. Rauw. **FDC 1.2-A Simulink Toolbox for Flight Dynamics and Control Analysis**. *Zeist, The Netherlands*, **2**, 1997. 58

[45] Gene F. Franklin, Michael L. Workman, and Dave Powell. *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1997. 82

[46] F. Rinaldi, S. Chiesa, and F. Quagliotti. **Linear Quadratic Control for Quadrotors UAVs Dynamics and Formation Flight**. *Journal of Intelligent & Robotic Systems*, **70**(1-4):203–220, 2013. 85

[47] Paolo Bolzern, Riccardo Scattolini, and Nicola Schiavoni. *Fondamenti di controlli automatici*. McGraw-Hill Libri Italia, 1998. 90

[48] G.E. Cooper and R.P. Harper. **The use of pilot ratings in the evaluation of aircraft handling qualities (Report No. NASA TN-D-5153)**. *Moffett Field, CA: NASA Ames Research Center. retrieved April*, **23**:2008, 1969. 94

[49] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. **Multilayer feedforward networks are universal approximators**. *Neural networks*, **2**(5):359–366, 1989. 106, 108

[50] Allan Pinkus. **Approximation theory of the MLP model in neural networks**. *Acta Numerica*, **8**(1):143–195, 1999. 108

[51] Partha Niyogi and Federico Girosi. **Generalization bounds for function approximation from scattered noisy data**. *Advances in Computational Mathematics*, **10**(1):51–80, 1999. 108

[52] Martin T. Hagan, Howard B. Demuth, and M.H. Beale. **Neural network design**. *PWS, Publishing, Boston, MA, USA*, 1996. 112

[53] Kumpati S. Narendra and Snehasis Mukhopadhyay. **Adaptive control using neural networks and approximate models**. *Neural Networks, IEEE Transactions on*, **8**(3):475–485, 1997. 116

[54] Orlando De Jesús, Jason M. Horn, and Martin T. Hagan. **Analysis of recurrent network training and suggestions for improvements**. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, **4**, pages 2632–2637. IEEE, 2001. 119