

# AUTONOMOUS HOVER FLIGHT FOR A

## QUAD ROTOR HELICOPTER

- LINEAR QUADRATIC CONTROL
- PIECEWISE AFFINE  
HYBRID SYSTEMS CONTROL

MASTER'S THESIS, 2007

GROUP 1037A

**AALBORG UNIVERSITY · AAU**  
AUTOMATION and CONTROL  
Department of electronic systems





**TITLE:**

Autonomous hover flight for  
a quad rotor helicopter  
- Linear Quadratic Controller  
- Piecewise Affine Hybrid  
Systems Controller

**PROJECT PERIOD:**

9<sup>th</sup> - 10<sup>th</sup> semester,  
September. 4<sup>th</sup> 2006 - June 7<sup>th</sup> 2007

**PROJECT GROUP:**

1037a

**GROUP MEMBERS:**

Jakob Bjørn  
Morten Kjærgaard  
Martin Sørensen

**SUPERVISOR:**

Rafael Wisniewski  
Jesper Abildgaard Larsen

**NUMBER PRINTED:** 6

**NUMBER OF PAGES:** 209

**FINISHED:** June 7<sup>th</sup> 2007

**ABSTRACT:**

This master thesis concerns two control approaches for the quad rotor helicopter. A Linear Quadratic controller with the aim of autonomous flight, and a Piecewise Affine Hybrid Systems (PAHS) controller on a theoretical level.

A new hardware platform to the quad rotor is developed, equipped with sensors and computing power. Software is developed in order to interface to the sensors and render an environment in which an estimator and controller can be implemented. The platform is through an acceptance test verified to fulfill the requirements.

A revision of a previously derived non-linear model for the quad rotor is carried out, with the aim of deriving a model based LQ state feedback controller. The controller is tested by simulation on the model and proves to stabilize the quad rotor. The controller is however not implemented.

A method designated the *Combinatoric Controllability Method* is derived in the framework of PAHS, in order to generate a control law on convex polytopes applicable to a simplified quad rotor system. The method is applied to a 3-cube and succeeds in generating a control law, that through simulation eventually proves to fulfill the control objective. The results however, indicate that one specific trajectory plot, briefly exits a non-admissible facet of the 3-cube, for then again to enter and converge towards the expected facet.



# PREFACE

This extended master's thesis is written by group 07gr1037a at the Department of Control Engineering, Aalborg University (AAU).

The thesis documents the work done in the period from Autumn 2006 to Spring 2007, and is specifically intended for the supervisors, the external examiner, Intelligent Autonomous Systems (IAS) students and other individuals interested in optimal control of a quad rotor helicopter and Piecewise Affine Hybrid Systems approaches.

In this report references are carried out according to the Harvard-method, e.g. [Cra05] and [Cra05, p. 100] if a more precise reference to e.g. a page or equation is needed. The full reference list can be located on page 149. Equations are numbered continuously throughout the report in the form of (6.1), while tables and figures are in the form of 6.1.

Abbreviations are written in their full length the first time, followed by the abbreviation itself in parentheses. The full abbreviation list can be located on page 153.

A CD-ROM containing project relevant material such as Simulink<sup>®</sup> models, MATLAB<sup>®</sup> code, various animations, video clips etc. can be found at the end of the report. If e.g. a motor model MATLAB<sup>®</sup> file is referred to, it is done in the following way, `/matlab/model/motor`, meaning that the file resides in the mentioned directory seen from the CD-ROM root.

The four rotor helicopter is throughout the report denoted as either the quad rotor helicopter or the Draganflyer X-Pro (X-Pro).

Aalborg University, 2007

---

Jacob Bjørn

---

Morten Kjærgaard

---

Martin Sørensen



# TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Course of events . . . . .	2
1.2	Project Objectives . . . . .	3
1.3	Outline of Thesis . . . . .	5
<b>I</b>	<b>Platform</b>	<b>7</b>
<b>2</b>	<b>Hardware architecture</b>	<b>11</b>
2.1	Connection chart . . . . .	11
2.2	Sensor description . . . . .	11
2.3	Actuators . . . . .	15
2.4	Safety equipment . . . . .	15
2.5	Computing power . . . . .	16
2.6	Depiction of the hardware platform . . . . .	18
<b>3</b>	<b>Software environment</b>	<b>21</b>
3.1	Robostix software . . . . .	21
3.2	Gumstix and Development Host Machine (DHM) software . . . . .	27
<b>4</b>	<b>Acceptance test</b>	<b>33</b>
4.1	Verification of sensor data . . . . .	33
<b>II</b>	<b>Modelling</b>	<b>37</b>
<b>5</b>	<b>Modelling overview</b>	<b>39</b>
<b>6</b>	<b>Motor-Gear model</b>	<b>41</b>
6.1	Complete motor model . . . . .	41
6.2	Gearing . . . . .	42
6.3	Verification of Motor-Gear model . . . . .	45
6.4	Conclusion . . . . .	45
<b>7</b>	<b>Rotor model</b>	<b>47</b>
7.1	Forces and Torques . . . . .	47
7.2	Linearization of induced velocity . . . . .	51
7.3	Verification of rotor model . . . . .	51

7.4	Conclusion . . . . .	53
<b>8</b>	<b>Body model</b>	<b>55</b>
8.1	Kinematics . . . . .	55
8.2	Dynamics . . . . .	56
8.3	The complete body model . . . . .	58
8.4	Verification of body model . . . . .	60
8.5	Converting to Quaternions . . . . .	62
8.6	Conclusion . . . . .	63
<b>III</b>	<b>Hover Controller</b>	<b>65</b>
<b>9</b>	<b>Control strategy</b>	<b>69</b>
9.1	Interface towards estimation group . . . . .	69
9.2	Control analysis . . . . .	72
9.3	Controller requirement specification . . . . .	73
<b>10</b>	<b>Rotor speed controller</b>	<b>75</b>
10.1	Deriving controller equations . . . . .	75
10.2	Deriving controller parameters . . . . .	77
10.3	Conclusion . . . . .	80
<b>11</b>	<b>Linear Quadratic controller</b>	<b>81</b>
11.1	System equations . . . . .	81
11.2	Linearization . . . . .	82
11.3	State Space form . . . . .	88
11.4	Deriving controller parameters . . . . .	91
11.5	Verification . . . . .	94
11.6	Conclusion . . . . .	100
<b>IV</b>	<b>Piecewise-Affine Hybrid System Controller</b>	<b>101</b>
<b>12</b>	<b>Introduction</b>	<b>105</b>
12.1	Theories of PAHS . . . . .	105
12.2	Hypothesis . . . . .	110
12.3	Control method applied to case study . . . . .	110



12.4	Combinatoric Controllability method synthesis . . . . .	116
12.5	Method applied to quad rotor helicopter system . . . . .	118
<b>13</b>	<b>Combinatoric controllability method</b>	<b>121</b>
13.1	Axis division . . . . .	121
13.2	Construction of a d-cube. . . . .	124
13.3	Facets and normal vectors . . . . .	125
13.4	Facets to close . . . . .	126
13.5	Linearization . . . . .	127
13.6	Form the control space polytope . . . . .	128
13.7	Bad vertex identification and polytope trimming . . . . .	129
13.8	Control law generation . . . . .	134
<b>14</b>	<b>PAHS closure</b>	<b>139</b>
<b>V</b>	<b>Closure</b>	<b>141</b>
<b>15</b>	<b>Conclusion</b>	<b>143</b>
<b>16</b>	<b>Perspectives</b>	<b>147</b>
	<b>Bibliography</b>	<b>149</b>
	<b>Abbreviations</b>	<b>153</b>
<b>VI</b>	<b>Appendices</b>	<b>155</b>
<b>A</b>	<b>Body model structure verification</b>	<b>159</b>
A.1	Test scenarios . . . . .	159
<b>B</b>	<b>Quaternions</b>	<b>167</b>
B.1	Introduction . . . . .	167
B.2	Quaternion multiplication . . . . .	168
B.3	Rotation of quaternions . . . . .	168
B.4	Time derivative of a quaternion . . . . .	169
<b>C</b>	<b>Robostix software</b>	<b>171</b>
C.1	Assignment of timers . . . . .	171
C.2	Motor driver . . . . .	175

## TABLE OF CONTENTS

---

C.3	Magnetometer driver . . . . .	176
C.4	Tachometer driver . . . . .	179
C.5	Inertial Measurement Unit (IMU) driver . . . . .	182
C.6	Robostix/Gumstix interface . . . . .	184
<b>D</b>	<b>Acceptance test specification</b>	<b>191</b>
D.1	Verification of sensor data: . . . . .	191



# INTRODUCTION

The interest in unmanned aerial vehicles for a wide area of applications, e.g. surveillance of larger areas, crop dusting, fire fighting etc. has been increasing over the past decade, as the required technology has become available. The immediate machines that accommodate the Unmanned Aerial Vehicle (UAV) category are the autonomous aeroplane and helicopter. Helicopters have clear advantages over the aeroplanes in the sense of being able to hover and land/take off in limited spaces.

For the past three years, the Department of Control engineering at AAU have aimed at bringing traditional model helicopters to autonomous flight. Not more than two years ago a quad rotor model helicopter with the commercial name Draganflyer X-Pro, was acquired with the similar aim of hover achievement. The work in this master thesis revolves around the quad rotor helicopter.

The first question one is asked about the quad rotor helicopter is how it stands out from the traditional one. Hence a small discussion of this seems impartial and necessary. The quad rotor helicopter stands out from the traditional one in the sense that it has four rotors that provide the lift and control instead of two. With respect to hover, the main difference is best explained by considering how the helicopters compensate from gyroscopic torques. Traditional helicopters basically compensate from the torque generated by the main rotor, through the tail rotor. However the tail rotor compensation conducts a sideways displacement of the helicopter, thus counter steering by tilting the main rotor blades is necessary. In this way hover is an ongoing and complex process. The quad rotor has four rotors which are fixed to a certain spin axis. The spinning directions of the rotors are set in pairs to balance the torques, therefore eliminating the need for a tail rotor. As long as the rotors rotate at the same speed the quad rotor helicopter essentially hovers, this proving to be a less complex manoeuvre to retain.

With regard to application and functionality the quad rotor helicopter has the same immediate advantages as the traditional helicopter. However a number of issues especially regarding the mechanical construction proves to be interesting from a control perspective point of view. The first and foremost issue concerns the modeling of the quad rotor as this proves to be a different and more feasible task.

A lot of work and effort has been put into the quad rotor helicopter since it was acquired, ultimately with the goal of autonomous flight. The preceding section will describe the course of events over the past year and a half.

## 1.1 COURSE OF EVENTS

This section will briefly give insight into the intermediate goals and objectives on the way towards achieving autonomous flight. The following Figure 1.1 provides an overview of the course of events from January 2006 where the helicopter was acquired, to June 2007 where autonomous flight is expected.

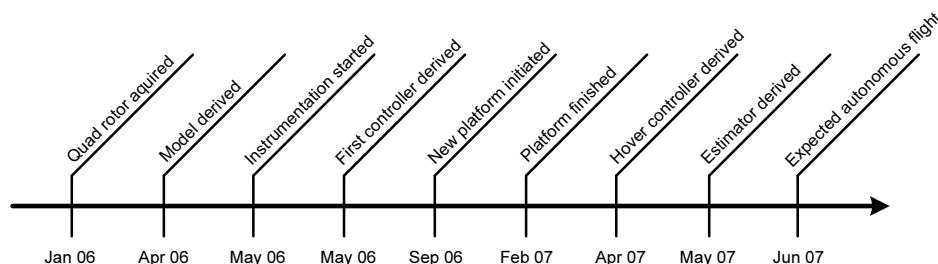


FIGURE 1.1: *Timeline illustrating course of events.*

As mentioned the quad rotor helicopter was acquired late January 2006. At this point, work began from scratch to provide a basic understanding of how the quad rotor helicopter operates, what kind of physics are involved and essentially how to combine this knowledge into a useful model for control purposes. The bold goal at that time, was to be able to hover at the end of the 8<sup>th</sup> semester.

Around the middle of April 2006 a suitable dynamical model was derived of the quad rotor. A model of the motor-gear, rotor, body kinematics and dynamics was derived, altogether constituting the complete quad rotor model. Some model parts were fully verified and some only structurally verified as a full verification of the complete model demands a hover manoeuvre.

Immediately after the last hand was put on the model, instrumentation was initiated (beginning of May 2006), this being the installation of an Inertial Measurement Unit for attitude determination and tachometers at every gear wheel to measure rotor revolutions. The tachometers, which proved to be reliable, are still used today.

At the end of 8<sup>th</sup> semester (late May 2006), the first state feedback controller was derived to drive the quad rotor into a hover state. Due to numerous issues e.g. little knowledge of what was going on in the on-board electronics, sensor saturation and basically a very limited time, the goal for autonomous hover failed.

In the summer of 2006 it was decided for two groups with three members each, to write an extended master's thesis (9<sup>th</sup> – 10<sup>th</sup> semester), ultimately with the goal of achieving autonomous hover for the quad rotor helicopter. The first task for the two groups

was to develop a completely new platform (9<sup>th</sup> semester). This was initiated at the beginning of September 2006 and finished at the beginning of February 2007. The outcome was a completely bottom up built hardware platform containing all sensors needed for a fixed position hover manoeuvre. Of new sensors can be mentioned Global Positioning System (GPS) module, new IMU and Magnetometer. Drivers and maintenance software was written to round off a fully equipt and working platform.

At this point the two groups devided the remaining work of designing an estimator and a controller. The estimator which is provided by [GHB07] and finished May 2007, combines all sensor data and produces quad rotor state estimates. Based upon this interface, an augmented state feedback hover controller was derived in the beginning of April 2007.

The quest for combining the estimator and controller and implementing this constellation on the quad rotor, has due to an excessive amount of work relating to other goals of the project, been postponed till after project delivery. The quad rotor is expected to hover June 2007.

## 1.2 PROJECT OBJECTIVES

As indicated above, this thesis also includes work that is not directly aimed towards achieving hover flight for the quad rotor helicopter. Hence, the aim of this section is to provide a clear comprehension of the objectives for this thesis.

The thesis has two main goals. The first in a non preferred order, is the achievement of a full autonomous hover manoeuvre for the quad rotor helicopter. The second goal revolves around a theoretical application of a Piecewise Affine Hybrid Systems controller defined on convex polytopes. Each of the two goals with their respective objectives are listed in the following.

### AUTONOMOUS HOVER FLIGHT

In order to achieve full autonomous hover three objectives must be met, each of which are described in the following.

#### **Platform**

As touched earlier in section 1.1 a hardware/software platform must be derived in order to provide a computational environment for the estimator and controller.

In order to obtain information about the position and attitude of the quad rotor, adequate sensors must be incorporated in the design. Enough computing power to handle sensor sampling, calculation of state estimates and calculation of control signals, must also be provided. In order to attain full autonomy the platform must be totally wireless, i.e. no strings attached.

### **Estimator**

In order to derive a state feedback controller for the quad rotor, an estimator that fuses sensor data must be developed. As earlier indicated this task is assigned the *Estimator Group*, [GHB07].

### **Controller**

A state feedback controller based on the estimator output, must be derived in order to control the quad rotor helicopter in hover. As this is a model based approach, the derived non-linear model from the 8<sup>th</sup> semester [KSG<sup>+</sup>06], requires an in-depth review and must be extended if necessary. The non-linear model should be linearized in a working point resembling hover and this linear model should provide the basis for the derivation of a linear state feedback controller.

If the three objectives are completed, the only thing left is to implement the estimator/controller constellation on the quad rotor platform. From this it is apparent that the goal towards achieving autonomous hover becomes a joint venture, where each group relies on the other groups work.

## PIECEWISE AFFINE HYBRID SYSTEMS CONTROLLER

As this is a master's thesis a certain academic level is required. The theories applied in the goal towards achieving autonomous hover are in many cases based on knowledge acquired from previous semesters.

This thesis will treat a specific type of Hybrid systems called Piecewise Affine Hybrid Systems, and seek to extend the work presented in the papers *A control problem for affine dynamical systems on a full- dimensional polytope*, [vSH04] and *Reachability and Control Synthesis for Piecewise-Affine Hybrid Systems on Simplices*, [vSHC06], in order to apply control laws from this framework onto the quad rotor helicopter system.

## 1.3 OUTLINE OF THESIS

This thesis is divided into six parts each of which concerns some of the objectives stated in section 1.2.

### **Part I: Platform**

This part concerns the development of the quad rotor hardware/software platform. The first Chapter 2 describes the hardware platform, where it is the intention to provide insight into all aspects of the hardware. Chapter 3 concerns the software developed for the onboard computers. Further elaboration is made on the soft real time target software environment, where the estimator and controller are to be implemented. The last Chapter 4 concerns an acceptance test performed on the platform in order to verify its capability.

### **Part II: Modelling**

This part concerns a résumé of the non-linear model from [KSG<sup>+</sup>06], and documents further extensions of the model. The first Chapter 5 provides a modelling overview, whilst the preceding chapters 6, 7 and 8 handle the Motor-Gear, Rotor and Body model respectively.

### **Part III: Hover Controller**

The Hover Controller part deals with the development of a controller which can stabilize the quad rotor in hover condition. The first Chapter 9 provides an overview of the control strategy. This including a description of the interface towards the estimation group, control analysis and requirement specification. Chapter 10 derives rotor speed controllers, whilst Chapter 11 derives the overall Linear Quadratic (LQ) controller. The part concludes with a verification of the controller in a simulational environment.

### **Part IV: Piecewise Affine Hybrid Systems Controller**

This part documents the work that has been carried out in order to apply theories of PAHS on convex polytopes, to the quad rotor system. The part commences with a specific nomenclature for PAHS. Chapter 12 gives an introduction to the PAHS framework, where theories of are described and problems relating to the application of PAHS on the quad rotor is identified. At the end a feasible method to generate a control law is presented and the application of this on the quad rotor is described in Chapter 13. Chapter 14 serves to conclude upon the results drawn in this part.

### **Part V: Closure**

This part concludes upon the work carried out in this master's thesis. Chapter 15 holds the conclusion and further perspectives are treated in chapter 16.

#### **Part VI: Appendices**

This part contains Appendices A through D, each of which individually elaborates on various subjects throughout the thesis.



# PART I

# PLATFORM

*It was around June 2006 decided by the Draganflyer X-Pro (X-Pro) group to discard the Printed Circuit Board (PCB) that came with the X-Pro, as a complete and thorough knowledge of what was going on in the electronics was necessary. This called for a new PCB, where an extensive research was initiated in order to find the correct sensors and computing power, to both handle sensor sampling, maintenance software and provide an estimator/controller environment.*

*The new platform was finished January 2007 in collaboration with, at that time group 937b. Hence this part serves to document and describe the 9<sup>th</sup> semester work which led to the new quad rotor platform.*

*The part contains three chapters where the first describes the new hardware composition. The second describes the developed software for the main processors and the third concludes with an accepttest performed on the platform.*

# CONTENTS OF PART I

---

<b>2</b>	<b>Hardware architecture</b>	<b>11</b>
2.1	Connection chart . . . . .	11
2.2	Sensor description . . . . .	11
2.2.1	GPS module . . . . .	13
2.2.2	Range finder . . . . .	13
2.2.3	IMU . . . . .	14
2.2.4	Magnetometer . . . . .	14
2.2.5	Rotor speed sensor . . . . .	15
2.3	Actuators . . . . .	15
2.4	Safety equipment . . . . .	15
2.4.1	Radio receiver and safety interrupt module . . . . .	15
2.4.2	Status indicators . . . . .	16
2.4.3	Motor arming switch . . . . .	16
2.5	Computing power . . . . .	16
2.5.1	Robostix . . . . .	17
2.5.2	Gumstix . . . . .	18
2.5.3	Wifistix . . . . .	18
2.6	Depiction of the hardware platform . . . . .	18
<b>3</b>	<b>Software environment</b>	<b>21</b>
3.1	Robostix software . . . . .	21
3.1.1	Main process design . . . . .	22
3.1.2	Remote Control (R/C) driver . . . . .	24
3.2	Gumstix and DHM software . . . . .	27
3.2.1	Gumstix Operating System . . . . .	27
3.2.2	Linux Soft Real Time Target . . . . .	28
3.2.3	Sensor drivers . . . . .	29
3.2.4	Development Host Machine software . . . . .	31

<b>4</b>	<b>Acceptance test</b>	<b>33</b>
4.1	Verification of sensor data . . . . .	33
4.1.1	Robostix packet . . . . .	34
4.1.2	Range finder . . . . .	34
4.1.3	GPS module . . . . .	34

---



# HARDWARE ARCHITECTURE



# 2

*From the original X-Pro construction little remains in the new setup. The mechanical parts are reused, that being the four arms with motor and rotors mounted, and the bottom carbon fiber board of the body. The top board of the body is an all new design which will contain all the electronics needed to build an autonomous quad rotor helicopter.*

The following description concerns the individual hardware parts and the connection of these pieces. First the overall system will be depicted in a connection chart. Later each transducer is listed with their respective features, followed by a short resume of the processors in the setup. Rounding the chapter off, the final hardware platform is depicted.

## 2.1 CONNECTION CHART

Given the project goals, it is possible to derive the needs for different sensor types. Furthermore data logging, safety features and possibility for manual flight are regarded as necessary in a minimum configuration to achieve autonomous hover. In Figure 2.1 all hardware components related to the project are depicted.

As the Figure shows it has been chosen to equip the X-Pro with sensors as GPS for absolute position estimate, Magnetometer for information about heading, range finder to aid the GPS in getting a altitude estimate, IMU for the possibility to propagate position and attitude. Besides all these sensors there are a number of components related to manual flight and other safety features. All together these transducers are connected to the main CPU, some via a slave processor, the Robostix. In the following all transducers, communication blocks and Processing Units are shortly reviewed to make the reader familiar with the physical composition of the platform.

## 2.2 SENSOR DESCRIPTION

All together there are five different sensor types built down on the X-Pro. In the following section these sensors are shortly listed with respect to the manufacturer, weight, power consumption and interface.

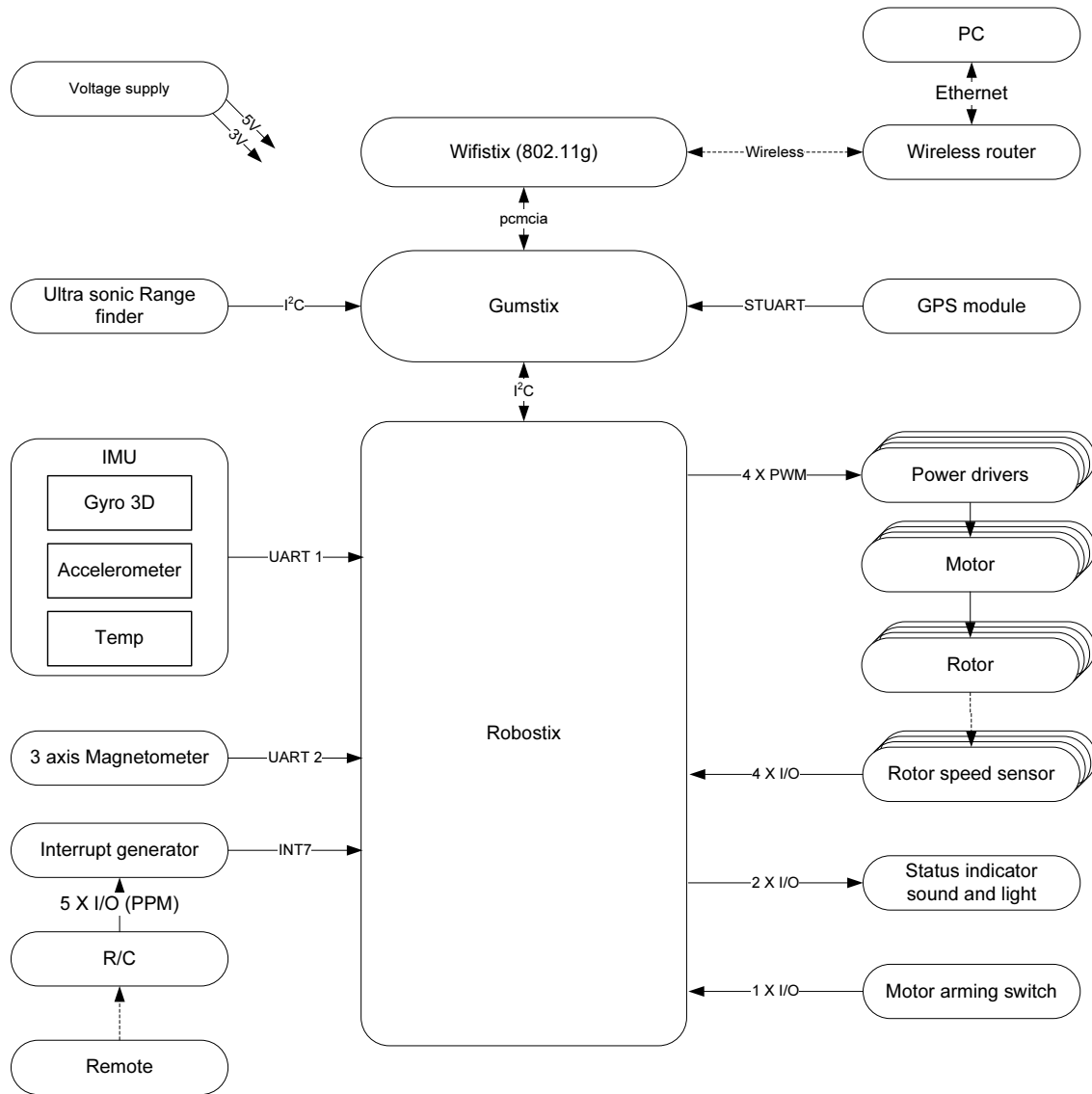


FIGURE 2.1: Hardware communication chart.

### 2.2.1 GPS MODULE

The purpose of the GPS module is to provide a global absolute position estimate. The chosen module is an all in one solution from U-BloX (UBX) listed as the unit SAM-LS that distinguishes itself with a position estimate rate of 4 Hz and low weight of 23 g. The precision of the position estimate is stated from the manufacturer as given below.

- 4Hz position update rate
- 2.5m Circular Error Probability (CEP) (50% of the time)
- 2.0m CEP with Differential Global Positioning System (DGPS) / Satellite Based Augmentation System (SBAS) (50% of the time depending on accuracy of correction data)
- 5.0m Spherical Error Probability (SEP) (50% of the time)
- 3.0m SEP with DGPS / SBAS (50% of the time depending on accuracy of correction data)

The UBX product is based on the ANTARIS Positioning Engine which has a clear defined interface over a serial connection, following the RS-232 standard. The module itself is operated at 3 V and thus also has a limited current draw at maximum 125 mA. Furthermore the module supports features such as DGPS and SBAS which the module can use to refine the position estimate. The datasheet can be found on the CD-ROM: [/datasheets/SAM-LS\\_Data\\_sheet\(GPS.G3-SA-03002\)-2.pdf](#) and [/datasheets/ANTARIS\\_Protocol\\_Specification\(GPS.G3-X-03002\).chm](#).

### 2.2.2 RANGE FINDER

The range finder is a sensor that in principle works as a sonar. The chosen sensor is a Devantech SRF08 and is a popular choice for a range finder. With this sensor it is chosen to estimate the distance to the ground, hence only one unit is needed. The range unit is capable of measuring distance between 3 cm to 6 m, but is limited to two meters to increase the possible sampling rate. The unit has a simple interface defined on the Inter-Integrated Circuit (I<sup>2</sup>C) bus standard, over which both initialization and ranging can be performed. More information is available on the CD-ROM [datasheets/srf08tech.pdf](#)

### 2.2.3 IMU

The chosen O-Navi Falcon MX IMU provides angular rates and linear accelerations. These sensor types are influenced by temperature, but the unit comes with a fully temperature compensated output. In Table 2.1 the measuring specification is shown.

	Resolution	Range
Acceleration	4 mg	$\pm 10g$
Angular rate	$0.30^\circ/\text{sec}$	$\pm 200^\circ/\text{sec}$

TABLE 2.1: *The performance specification for O-Navi Falcon MX IMU.*

Inside the IMU both the angular rate and the accelerometer measurement is filtered using a 5<sup>th</sup> Order Elliptic Low Pass Filter with a bandwidth of 100 Hz. The interface to this unit is defined over a serial connection following the standard of RS-232. The sampling rate provided by the IMU can be user selected to 1, 10, 20, 50, 75 & 100 Hz, where the 100 Hz is chosen in the implementation. All this is contained in a unit, weighing only 11.5 g with a maximum power consumption, according to the data sheet, of  $5V \cdot 89mA = 445mW$ . For more information turn to the CD-ROM in [/datasheets/Falcon\\_Extended\\_Manual.pdf](/datasheets/Falcon_Extended_Manual.pdf).

### 2.2.4 MAGNETOMETER

The magnetometer provides the direction of the magnetic field of the Earth. In it self this information does not give a compass direction, but combined with other sensors it will be possible to estimate the attitude of the X-Pro. The chosen magnetometer is of the type Micromag3, and is a 3-axis magnetic field sensing module. The supply voltage is 5 V and draws a negligible current compared to the IMU or the GPS. The interface to the sensor is done serial via the SPI protocol. The specification of this particular sensor is listed in Table 2.2. Further information can be found on the CD-ROM in [/datasheets/MicroMag3\\_Data\\_Sheet.pdf](/datasheets/MicroMag3_Data_Sheet.pdf)

	Resolution	Field measurement range
Magnetometer	Resolution: $0.015 \mu T$	$\pm 1100 \mu T$

TABLE 2.2: *The performance specification for Micromag3 Magnetometer.*



### 2.2.5 ROTOR SPEED SENSOR

The tachometers placed under each of the four gear wheels are build from a hall effect sensor that detects a passing magnet that is mounted on the gear wheel. When the magnet passes the hall sensor the signal out changes from 5V to 0V. This has proved to be a simple and reliable sensor for estimating the angular velocity of the rotor blades. Only one magnet is placed on the gear wheel which lead to a position dependent speed sensor. By measuring the time between a passing magnet it is possible to estimate the mean angular velocity over one round. This also means that there is a limit to the speed of the motor dynamic for which the tachometers still can follow.

## 2.3 ACTUATORS

It has been chosen to build the X-Pro up around the original brushed DC motors. All the original power electronics has been replaced with new power drives. The chosen design is to drive the four motors by four Pulse Width Modulation (PWM) signals, and it should be the task of the power drives to drive the motors as efficiently as possible. The power drive circuit is build up round a MOSFET transistor, with a turn off snubber dimensioned for the 300 Hz switching frequency. The PWM signal to voltage map is non-linear and must be taken into account when implementing a controller on the X-Pro. The data sheet for the motors can be found on the CD-ROM in `/datasheets/rs_545sh.pdf`.

## 2.4 SAFETY EQUIPMENT

Not directly related to autonomous flight are a number of components that all serve to avoid that the X-Pro inflicts damage on its surroundings or on it self. These components are shortly reviewed in the following, and basically ensures that a human pilot can take control of the vehicle at any given instant.

### 2.4.1 RADIO RECEIVER AND SAFETY INTERRUPT MODULE

A 6 channel FM receiver, from the manufacturer Futaba (model R136F) is thus fitted to on the X-Pro to provide just that functionality. The original Futaba transmitter is used which provide all calibrating and steering features necessary for manual flight. The purpose of

the safety interrupt module is to provide the possibility of changing between two different modes during flight using the R/C. This can be done using the receiver's 5th channel, the E switch on the transmitter. The idea is that the module generates an interrupt to the Robostix if it is needed to take the X-Pro out of an autonomous control that has become hazardous, or to give back the control to the autonomous controller. This has shown to come in handy in the integration process to ensure the security of X-pro and surrounding personnel and hardware.

### 2.4.2 STATUS INDICATORS

Having different modes calls for status indicators indicating which state the X-Pro is in right at the present time. Both a sound and light is chosen as indicators. A 82 dB buzzer is fitted for urgent notification to the backup pilot, and a large size Light Emitting Diode (LED) for showing the mode of the X-Pro. The precise use of these will be clarified in the following software section. A last safety feature is a physical motor switch which effectively can disarm the X-Pro regardless of whatever the transmitter might send.

### 2.4.3 MOTOR ARMING SWITCH

When working on and near the X-Pro a physical switch is added to the structure. The purpose is to ensure the person working on the X-Pro on the ground, that the rotors start spinning if the remote is accidentally flipped on or another remote working in the same frequency band comes in range. Although these occurrences are highly unlikely, it can have damaging effects thus the problem is handled by relative easy means.

## 2.5 COMPUTING POWER

Just as important as all the sensors is the presence of some processing unit that can gather and calculate the needed control signals. As mentioned in Section 2.1 on page 11 the chosen processor design is provided by the firm Gumstix [Gumb], which is a family of small computer related units that each are approximately the size of a packet of chewing gum, thus the name. This product has been chosen since it provides a flexible and computational powerful block of low weight. The following three modules weigh together 56 g including antenna for wireless communication. All three units are built down on the X-Pro thus forming the central computing and communication unit. These latter men-

tioned units are by the manufacturer called: The Robostix, which is a relative slow slave processor built down on a PCB. The I/O pins of the processor are exposed together with communication lines from the Gumstix, which is the Central Processing Unit (CPU). The final unit is the Wifistix which provides wireless connection to the DHM, which is the PC that handles data logging. In the following a listing of the performance specifications and other features related to these, to provide the reader insight in the capabilities of this platform.

### 2.5.1 ROBOSTIX

This board is built up around a 16MHz Atmega128 processor. The board is specially made for external connection thus the ports of the Atmega are exposed on pins for fast connection. The Atmega128's task is to sample the IMU, magnetometer and the rotor speed sensors. Besides this it should set PWM signals for the four motor systems, handle status indicators and lastly interface to the radio receiver. To fulfil these tasks the following peripheral features are used.

#	Type	Usage
4	PWM	Power drives
1	UART	IMU
1	SPI	Magnetometer
1	Interrupt	Radio receiver
4	Interrupt	Rotor speed sensors
2	GPIO	Status indicators
1	GPIO	Motor disable switch
1	I <sup>2</sup> C	Communication with Gumstix

TABLE 2.3: *The use of peripheral features on the Robostix's Atmega128 processor.*

The Robostix relays sensor values via an I<sup>2</sup>C connection, but in this configuration the Robostix is acting as the slave unit and can thus not initiate a sensor relay on its own, thus it has to wait for the Gumstix to request packets. More on this topic is given in the software section, C.6 on page 185.

### 2.5.2 GUMSTIX

The Gumstix features a 400 MHz Intel XScale processor of the type PXA255. It has 16 MB flash memory and 64 MB of SDRAM. Amongst the physical connectors are two Hirose connectors which provide easy connection via a 60-pin Hirose to the Robostix and on the other side of the Gumstix connection to the Wifistix is made through a 92-pin Hirose. As mentioned the connection to the slave processor and the range finder is done via the I<sup>2</sup>C connection, leaving only the GPS sensor to be directly interfaced to the Gumstix. The GPS sensor occupies the only available serial connection on the Gumstix at a rate of 57600 baud. For development purposes it is possible to connect to the Gumstix via a dedicated RS-232 serial console, but for higher communication speed the wireless communication is added to the system, in the presence of the Wifistix.

### 2.5.3 WIFISTIX

This wireless module is a fully configurable wireless board, following the 802.11(g) standard. Which means that the bandwidth can be up to 54 Mb/s. This connection makes it possible to follow the performance of the system very closely while flying. The choice of wireless ethernet connection though poses a limitation in the range of reception, which can be as short as 10 m around the omni-directional aerial antenna, but in these first steps of building a hover controller it should be possible to stay inside this range of the access point.

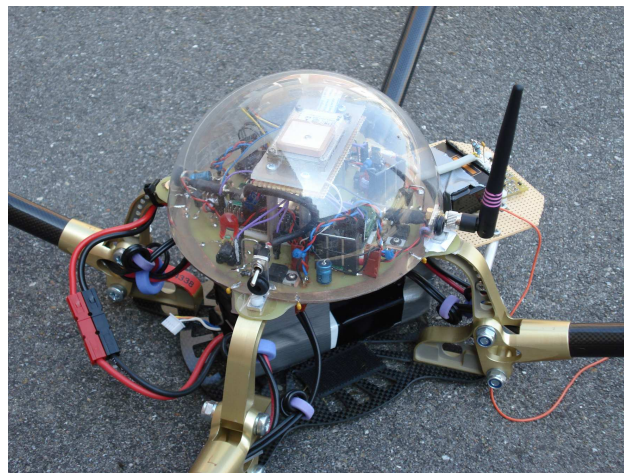
## 2.6 DEPICTION OF THE HARDWARE PLATFORM

To give a more spatial perception of the structure, photos of the X-Pro are shown in this section. All the mentioned hardware are mounted under consideration of the general structure and special needs of the individual sensor. Observing Picture 2.1 on the facing page the X-Pro is shown from the front, seen by the black wifi antenna is facing backwards. The four strings binding the four rotor arms together are mounted to reduce the vibration from the working rotors. Between the back and right rotor arm a small plateau is built for the range finder. This is necessary due to the wide spreading of the sound waves from the range finder can not be obstructed by the body structure. The radio receiver is also placed on this plateau to make place for other hardware under the transparent dome.



**PICTURE 2.1:** *The final hardware platform fully equipped with sensors, total weight is 2.6 kg.*

Going closer, Picture 2.2 shows a close up of the centre body part. Near the left arm, in the bottom of the picture the arming switch is placed, that has overall control of the four rotors, with this flipped down no control signals can be send to the power drives. To the left of this switch is the red status indicator LED, which purpose will be described later on. The wires to each motor and the signal from the tachometers run side by side inside each rotor arm. Due to the large fluctuating current towards the motors inflict noise on the tachometer signals. This problem has been solved by adding ferrite beads on both outgoing and incoming signal, which absorbs the energy in the high frequency noise.



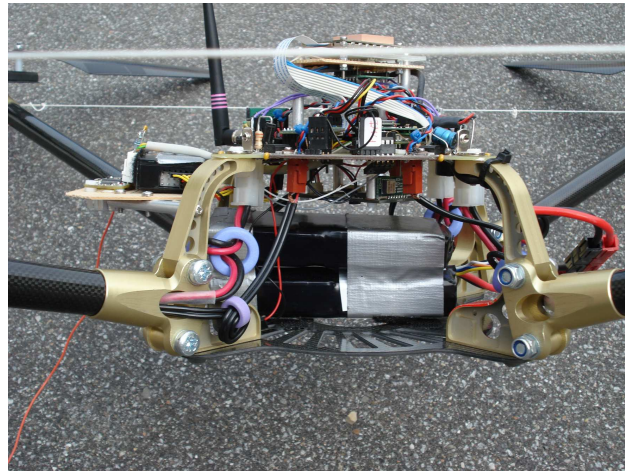
**PICTURE 2.2:** *Close up on the body of the X-Pro, here seen from the left.*

Picture 2.3 is seen from the side with the protective dome removed. In the top of the

## 2.6. DEPICTION OF THE HARDWARE PLATFORM

---

picture the GPS module is placed on yet another plateau to make sure that the sensor has a clear view of the sky above it. Underneath the GPS module is the magnetometer mounted relatively far away from the magnetic noise sources. Underneath the main PCB the IMU is placed, such that the gyros are close to the centre of mass. Lastly the battery is placed on top of the bottom plate.



**PICTURE 2.3:** *Close up on the body of the X-Pro, with the protection cap removed.*



# SOFTWARE ENVIRONMENT

*As mentioned in the previous chapter all peripheral components, depicted in Figure 2.1 on page 12, are gathered and processed by the Robostix and Gumstix. This calls for the development of software for each of the mentioned processors. Hence this chapter will revolve around the description of Robostix and Gumstix software.*

First the Robostix is treated which has the main task of forwarding sensor data to the Gumstix. The low level code is written in C and cross compiled to form a hex file which is loaded onto the Atmega128. Second the Gumstix software is treated which is written in high level C code. This software is also cross compiled to fit the system specific architecture of the Gumstix, namely the Intel Xscale PXA255 processor. Next the Development Host Machine software is described which is basically defined by a Linux Soft Real Time Target application in MATLAB<sup>®</sup>, Simulink<sup>®</sup>. It is in this environment the controller will be derived and also implemented. Finally an accepttest will conclude upon the new quad rotor platform, providing an image of what the platform is capable of.

## 3.1 ROBOSTIX SOFTWARE

This section deals with the software design for the Robostix. In order to interface to the sensors and collect data, low level drivers are written. Most drivers for the sensors are designed as Finite State Automata (FSA)s, and fault detection is built in to an extent considered appropriate. Each driver provides an Application Program Interface (API) used by a main process which is designed to gather, concatenate and transmit data packets from the Robostix to the Gumstix through the I<sup>2</sup>C bus in an efficient and reliable way.

This section will handle some of the highlights of the Robostix design, this being the main process design and the R/C driver. The timer assignments and rest of the drivers together with the robostix/gumstix communication are handled in Appendix C on page 171. The main process handles the following six components.

- Inertial Measurement Unit
- Magnetometer
- Tachometer
- R/C receiver
- Motor drivers

- Gumstix communication

The interaction between the listed components and the main process is illustrated in Figure 3.1.

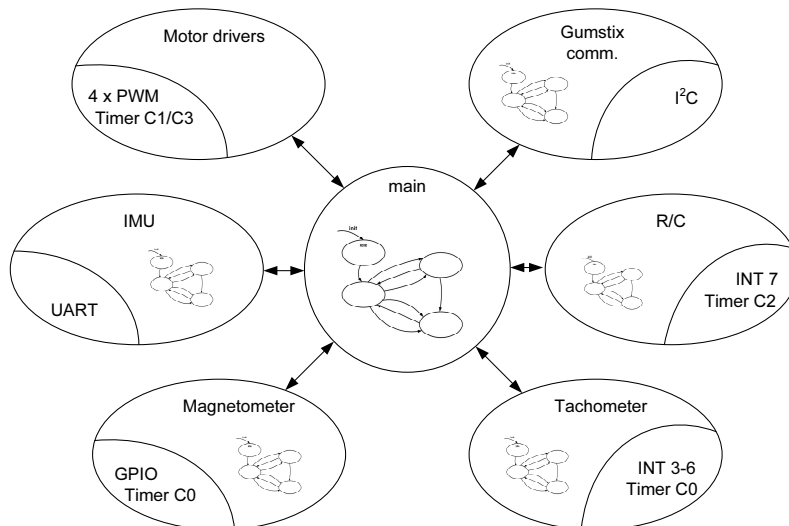


FIGURE 3.1: The main process interacting with peripheral components. Along with each component, hardware interface and timer is shown, however timer only if assigned.

The following section will concern the design of the main process.

### 3.1.1 MAIN PROCESS DESIGN

The main process, which in turn is a single function, initializes the Robostix and the connected sensors at system startup and enters an `idle` state. The reason for entering the `idle` state is to ensure a safe startup, where the rotors are inactive and the R/C transmitter is turned on, before any controlling can take place. In this way, the X-Pro will not automatically take off at power-up. Besides the `idle` state there are three active states the X-Pro can be controlled from:

1. The `human pilot` state, where the X-Pro is controlled from the R/C transmitter, is the state that every flight must be started from. It is always possible for the human pilot to take control from any state, by use of a safety switch on the R/C transmitter, in case of an emergency. Hence a switch activation will always result in an event leading to the `human pilot` state.



2. The `Gumstix pilot` state is where the control signals are calculated by the Gumstix, based upon sensor measurement. It can be switched between the `human pilot` and the `Gumstix pilot` as desired during flight. If a fault is detected on the R/C connection, the Gumstix takes over control, as it is presumed to have the best controller and state estimates.
3. The `Robostix pilot` state has the main purpose of being the last resort for controlling. In case of Gumstix faults, it serves mainly as an indicator for the `human pilot` to take over by indicating faults using a buzzer and a flash LED. Meanwhile it uses some conservative controlling, which should serve as the system “parachute” to prevent major damages.

These before mentioned states has the purpose of ensuring a stable system with known behavior. To furthermore ensure correct navigation between the states a supervising system which is based on a FSA is implemented. The FSA as illustrated in Figure 3.2.

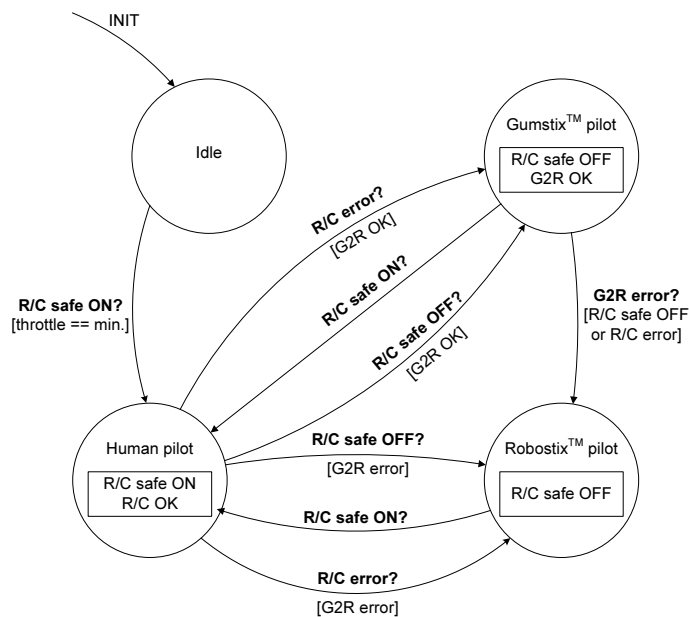


FIGURE 3.2: The FSA for the main function.

The FSA bases its transitions on status messages from the communication lines (Gumstix-to-Robostix connection (G2R) and R/C), hence it must therefore be possible to detect errors in these.

**Idle:** At startup the main function ensures that the system initializes and enters the `idle` state. The system stands still in `idle` until another state is requested.

**Human pilot:** When R/C safe is switched ON and it has been assured that throttle is adjusted to minimum, this state can be reached. Otherwise the supervisor makes sure the system does not change from idle state.

The `Human pilot` state can also be reached from the `Gumstix pilot` state or the `Robostix pilot` state as well, as described in the following.

**Gumstix pilot:** When the G2R is OK, the X-Pro can be set to be controlled by the Gumstix (`Gumstix pilot` state). This is done manually by switching OFF the R/C safe or automatically if the R/C receives error signals.

The FSA returns from the `Gumstix pilot` state to `Human pilot` state, if R/C safe is switched ON.

The FSA switches from the (`Gumstix pilot` state to the `Robostix pilot` state, if there occurs an G2R error meanwhile the R/C safe is switched OFF.

**Robostix pilot:** When the G2R is not OK, the X-Pro can be set to be controlled by the Robostix (`Robostix pilot` state).

This can take place when R/C safe OFF or automatically if the R/C transmits error signals to the Robostix. The buzzer and indicator LED must be turned on in this state, indicating towards the pilot that something is wrong. The FSA returns from the `Robostix pilot` state to `Human pilot` state, if the R/C safe is switched ON.

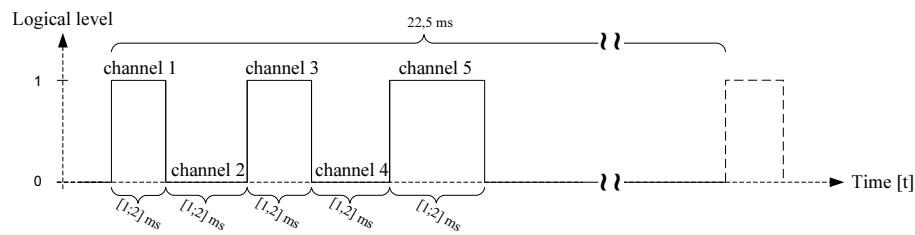
The following section contains documentation of the R/C driver design.

#### 3.1.2 R/C DRIVER

As mentioned earlier in Section 2.4 on page 15 it was chosen to implement a safety feature in the sense that a human pilot should be able to take the control if the designed hover controller, driven from the Gumstix, fails to prevail. This calls for a R/C interface and in order to control the X-Pro using this, the Robostix needs to get the information from the R/C receiver module and together with the IMU generate the appropriate PWM-signals.

The R/C-module is connected to `INT7` on the Robostix through an OR-gate. The OR-gate uses the fact that although there are five channels, which are following each other in time. So by concatenating channel 1, 3 and 5 all information from the five channels can be represented in one serial stream, thus saving interrupt pins in the design.

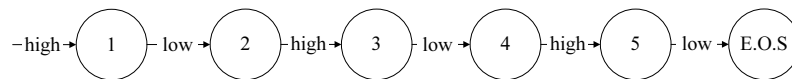
In the combined signal each rising and falling edge of the signal generates an interrupt to the Robostix and provides the possibility of determining each channel length. An example of the signal from the OR-gate is illustrated in Figure 3.3.



**FIGURE 3.3:** The signal lead into *INT7*. The five channels of the R/C receiver can vary individually between  $[1; 2]$  ms and a total R/C receiver package is 22, 5 ms.

The convenience and cleverness of this solution is that a minimum of interrupts is used on the Robostix.

This design is the set of for the Discrete Event System (DES) shown in Figure 3.4.



**FIGURE 3.4:** The DES system when receiving the R/C-signal. E.O.S is end of signal.

The DES changes state every time the transition is fulfilled. Based on this the design of the R/C driver can be derived, using a location observer to handle if errors should emerge.

### Overall design

The FSA is designed in such a way, that the input has to be correct, as described in Figure 3.4. Also if the time between two edges is outside the interval  $[1; 2]$  ms, the sample will be discarded and a error variable will be increased.

Figure 3.5(a) illustrates the FSA-driver for the R/C-module. When the Robostix gets an *INT7* it checks whether the input is correct or not, by checking the input port *INT7* to validate whether it is high or low when needed. The driver also consider whether the signal is within the a given time of two edges. If this is the case, the FSA switches state and await interrupts to gather the rest of the R/C package.

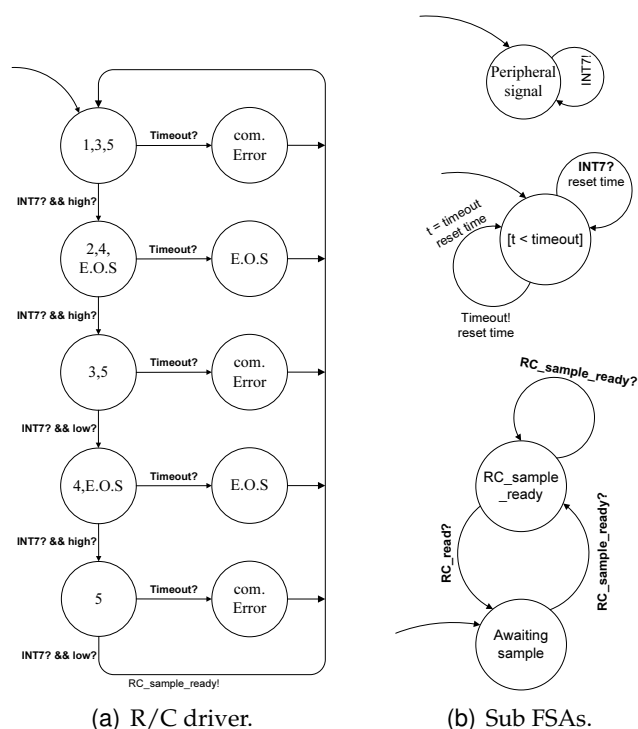


FIGURE 3.5: (a) Shows the complete driver design using a location observer, extended with error handling. (b) The FSA which is either dependent of or the interface to (a).

There are three possible places where a low-to-high interrupt can be started, namely channel 1, 3 or 5. This is illustrated by the first circle, which contains the states 1, 3 and 5. If no timeout occur then the next circle contains state 2, 4 and E.O.S, etcetera. The state names `com.error` and `E.O.S`. will be further described in the following.

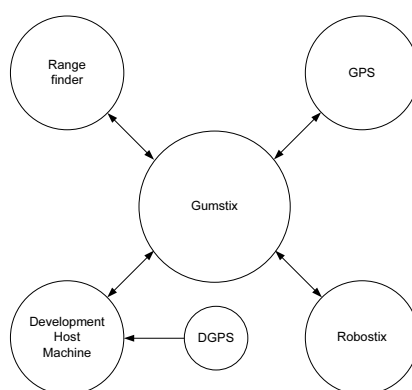
- **com.error:** If a timeout occur when being in either state 1, 3 or 5 the only transition is to the `com.error` state. Since this means that the signal has been high longer than the defined timeout, the only solution for this is a broken wire or a broken R/C-module
- **E.O.S.:** This is end of signal and there by also a timeout, because the next channel does not appear within the defined time.

Figure 3.5(b) shows the small FSA driver uses in order to collect a R/C package. The top FSA is the one generating the transition event. The one in the middle is defining whether timeout between transitions has occurred and the bottom one is the interface to the Robostix main.

This concludes the R/C driver design. The Robostix/Gumstix communication interface can be found in Appendix C on page 171. Next the Gumstix and DHM software will be handled.

## 3.2 GUMSTIX AND DHM SOFTWARE

As written in Section 2.5 on page 16 the Gumstix is the main computer on the X-Pro. It is the one running the estimator and controller, and also relaying sensor data to the DHM. Figure 3.6 shows the peripheral devices connected to the Gumstix.



**FIGURE 3.6:** *The peripheral devices connected to the Gumstix. The data flow to and from the Gumstix is donated by an arrow.*

As can be seen on the Figure the Gumstix is connected to the Robostix as depicted in Figure 2.1 on page 12. The range finder is also connected to the Gumstix on the same communication media as the Robostix. Finally the DHM is the slave for receiving data from the Gumstix and host for relaying the DGPS package to the Gumstix, which are used for correcting the GPS signal.

In the following a description of the Operating System (OS) on the Gumstix, Linux Soft Real Time Target (LNX), the drivers to interface with the sensors and the Robostix.

### 3.2.1 GUMSTIX OPERATING SYSTEM

The OS on the Gumstix is contained in the Buildroot provided by the Gumstix, Inc., [Guma]. The important parts of the Buildroot package will further be described below.

**Buildroot:** The Gumstix is equipped with a modified version of BusyBox, which is a tiny version of Linux. Using the menuconfig option gives the ability to configure

the OS so it can fulfill the requirements one might have. Gumstix, Inc. have modified the BusyBox such that, the hardware under their brand can be connected without adding any third part software, the drivers only has to be enabled via the menuconfig option when configuring the Buildroot.

**Toolchain:** The cross compiler which makes it possible to compile source code on any computer running a Linux distribution. Thereby already developed software only needs to be recompiling in order to run on the Gumstix. The cross compiler is a modified version of the gcc compiler. Thereby the software and drivers can be developed in C and compiled on a more powerful computer than the Gumstix.

### 3.2.2 LINUX SOFT REAL TIME TARGET

In this section the LNX for MATLAB<sup>®</sup> is described. The LNX developed by [Bha], is equivalent xPC from Mathworks. The operational principle of the LNX and how to setup the environment will be further described in the following.

**Operational principle:** The similarities of LNX and xPC are that the Simulink<sup>®</sup> model is compiled in MATLAB<sup>®</sup> using the chosen mex compiler. The difference is that the LNX generates an executable file which is to be executed on the target machine, whereas the xPC target compiles the Simulink<sup>®</sup> model into a bootable OS. This is when the real difference is stated, the xPC runs "hard" real time, whereas the LNX, as the name states it, runs soft real time. Because of the LNX runs on the OS of Gumstix the LNX is changing the scheduler so it gets highest priority. Then the LNX is running it uses a POSIX real time clock to generate periodic signals, corresponding to the overall samplings frequency. To gain a higher resolution of the scheduler, the scheduling frequency have been changed to the maximum value, on the Gumstix the maximum scheduler frequency is 1000 Hz, the default value is 100 Hz. This introduces a overhead when increasing the number of process switches, but is a necessary when high accuracy of sampling is needed.

**Setup:** The setup of the LNX is more or less the same as xPC. The LNX is installed as described in the `readme.txt` which can be found on the CD-ROM: `/matlab/lrx`. The `lrx` Makefile has been changed in order to use the correct cross compiler, namely the one described above. The chosen method is selected in the Simulink<sup>®</sup> model, and the interfaces to the sensors can developed using a s-function.

Table 3.1 on the next page shows the advantages and disadvantages when using LNX instead of implementing it all in C.

Advantage	Disadvantage
Fast development	Overhead
Intuitive structure	OS scheduler
Real time scopes	

TABLE 3.1: *Advantages and disadvantages using LNX instead of implementing the controller and estimator by hand.*

One of the advantages of the LNX is that the development of as an example a controller is fast, counter to if it was to be implemented in C. Due to the MATLAB<sup>®</sup> implementation and the mex compiling the executable application containing the controller generates a overhead of code, meaning that the real time is harder to maintain. The real time demand is also harder to maintain if the scheduling frequency is set too low and thereby the change of tasks in the kernel takes longer, resulting in more overhead.

Since that the LNX is equivalent to MATLAB<sup>®</sup> xPC target, the structure of the Simulink<sup>®</sup> model is intuitive, if the X-Pro model with the controller was to be implemented in C. The last advantage is the real time scopes in the Simulink<sup>®</sup> model in MATLAB<sup>®</sup>, and thereby able to see the sensors values real time.

### 3.2.3 SENSOR DRIVERS

To interface the sensors attached to the Gumstix drivers needs to be developed. As depicted in Figure 2.1 on page 12 the sensors that interfaces to the Gumstix is the GPS and the range finder. The Robostix driver and range finder are gathered in the same driver because the interface specification to there are the same. Only the protocol to communicate with these individually are different. The first driver handled is the driver for the GPS module.

#### GPS

The GPS module is connected to the Gumstix through a serial interface, and fully software controlled thus the interface is also wide but intuitive. Therefore it is important to determine which packages are received from the GPS module, thus the communication protocol is described. To handle DGPS a drivers has been developed in [BBG<sup>+</sup>06, pp 83.] and will not be further describe here.

**Protocol specification:**

The key features of the protocol are the following, [Ubl]:

- It is compact, 8 bit binary data is used.
- Checksum protected, using a low overhead checksum algorithm.
- Modular, using a 2-stage message identifier (Class- and message ID).

The UBX package is configured as illustrated in the following Figure 3.7.

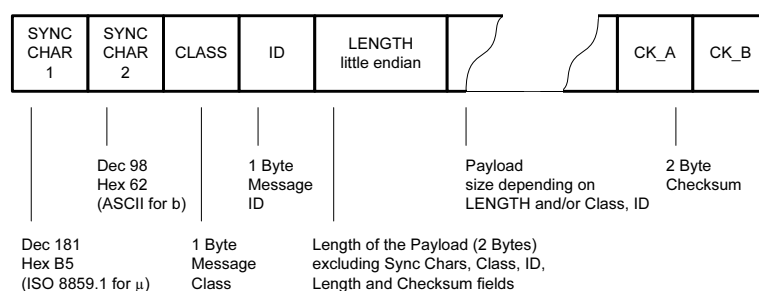


FIGURE 3.7: UBX binary protocol, [Ubl].

Every message starts with two synchronizing bytes. One byte follows containing the class which defines the basic subset of the message. Another byte ID defines more specifically what is actually contained in the package. The length part consists of two bytes which represents the length of the payload only. The payload is a variable length field. The message ends with a 2 byte checksum CK\_A and CK\_B. The 8-bit Fletcher checksum algorithm can be found in Appendix C.6.1, Equation (C.10).

The GPS module must be configured in accordance with the specific goals for the project. As mentioned earlier it is possible to configure everything from the serial port setup to which packages to receive from the module, either by polling or periodic output and guide the way the position is calculated. These various options can be found in the protocol specification [Ubl].

The GPS module has been configured to return NAV\_SOL and NAV\_DOP. NAV\_SOL stands for Navigation Solution Information and contains all the information of the position of the GPS module. NAV\_DOP stands for Dilution of precision and contain information of how good the position estimate is.

With the above the driver to the GPS module has been developed. The source code to interface the GPS module and the DGPS software can be found on the CD-ROM in: /source/gps/\*.



### **Range finder**

The range finder is connected to the I<sup>2</sup>C bus on Gumstix as described in Section 2.2.2 on page 13. To communicate over the I<sup>2</sup>C bus the API developed by [Gumb] is used. The API provides some commands for receiving and sending data to and from the connected devices. The range finder is a slave unit, meaning that the Gumstix has to send a command to get the sensor to start a ranging. After a predefined time period the Gumstix again send a command to collect the data. The predefined time period is defined by the maximum range, if a smaller maximum ranging is wanted the time period is minimized and vice versa. Therefore the driver to the range finder has been developed to include the functionality to start a ranging, collect the data, change the maximum range.

### **Robostix**

The contents of the data from the Robostix is described earlier and will not be stated again. As stated before the Robostix and Gumstix are connected through the I<sup>2</sup>C bus. Due to the data flow from the IMU to the Robostix, the Gumstix samples the Robostix at the same rate, namely 100 Hz. The sampling frequency is important to minimize the jitter because the sensor data on the Robostix is not timestamped, whereas the Gumstix timestamps every package when they are received.

## **3.2.4 DEVELOPMENT HOST MACHINE SOFTWARE**

All the software from the Gumstix is also available as a stand alone application. There all the sensor data is relayed using a TCP connection from the Gumstix to the DHM. The relaying of the DGPS signal is also integrated in this application. The software is implemented in order to get sensor data and to see if it was capable of implementation. The LNX is used when implementing the hover controller and estimator when this is to be tested later on.





# ACCEPTANCE TEST

The purpose of this acceptance test is to give an idea of what the developed X-Pro platform is capable of. Normally an acceptance test is written with respect to a previously written specification of requirements. However in this specific case the latter does not exist due to the fact that various requirements were developed during the 9th. semester and were never quite clear from the beginning. Hence this acceptance test will revolve around important issues from a user perspective point of view. The results of the acceptance test are described in this chapter and are based on the acceptance test specification Appendix D on page 191.

Previously in [BBG<sup>+</sup>06] several module tests have been performed on both the hardware and software of the X-Pro. However this test will particularly cover how well the modules work together, especially in the X-Pro environment. Firstly tests are performed in the **non rotary movement** state which is a state where all software / hardware is integrated and powered up, but without the stressfull and noisy environment that comes with the **rotary movement**. When this is achieved the system is tested in full **rotary movement** pushing the system to its limits.

The acceptance test specification is built around the following main structure

1. What to test ?
2. Necessary conditions
3. Expected results
4. How to test ?
5. Test results

The issue covered in this acceptance test is **Verification of sensor data**, thus this is important considering autonomous flight.

## 4.1 VERIFICATION OF SENSOR DATA

It is imperative that sensor data at all times and in all flight states are received on the Gumstix correctly and that the estimator and controller can rely on this.

Within the Robostix packet lies sensor data from the magnetometer, the tachometers and the IMU. Directly channeled to the Gumstix are the range finder and GPS module. The following subsections will conclude on the acceptance test with respect to the mentioned sensors.

### 4.1.1 ROBOSTIX PACKET

The acceptance test of the robostix packet contains verification of the various contained sensor data in the form of sanity check. As the Robostix transmits packets at 100 Hz it is verified if the packets are received at this rate at the Gumstix and the DHM. The actual jitter level is furthermore concluded upon.

All data contained in the robostix was sanity checked and thereby verified. The frequency was determined to 100.007 Hz which is acceptable and only a small negligible jitter level was detected.

### 4.1.2 RANGE FINDER

The acceptance test of the range finder contains verification of the sensor data. The range finder is sampled at 50 Hz and this is to be verified. The range limit is set to 99 cm, and this is also to be tested.

The range finder is able to measure the range up to 99 cm. The resulting sampling frequency was determined to 50.0002 Hz, and thereby the sampling rate is fulfilled. The jitter was considered acceptable.

### 4.1.3 GPS MODULE

The acceptance test of the GPS module contains packet verification, issues concerning frequency and jitter of sampling and furthermore if the DGPS functionality works as it should. All results stated in the following applies to both **non rotary movement** and **rotary movement**.

#### **Packet verification**

Although the UBX protocol specifies a large amount of packages that can be requested only few are considered relevant. The packages are **NAV\_SOL**, **NAV\_DOP**.

All mentioned packets were received successfully and sanity checked with respect to its contents. However it seems as the module dropped a packet set, exactly when 3D fix was acquired. This incidence is believed to be module specific more than a communication line failure as the probability of packet loss right at that instant would be extremely low.

**Frequency and jitter**

Apart from the single outlier mentioned above the packets are received with a 250 ms interval i.e. 4Hz as expected. Moreover the jitter level is low and more importantly constant.

**DGPS**

The GPS module receives Radio Technical Commission for Maritime services (RTCM) packets mostly within a second as it should, however a fair amount of jitter is still apparent. At two striking intervals a packet is as much as five seconds delayed. This is however believed to be due to propagation delay on the network line from the Suldrup antenna to AAU. Furthermore the bandwidth used is not dedicated to the RTCM reception, so a profound use of bandwidth from others at that instant could prove to have been the decisive factor.



# **PART II**

# **MODELLING**

*This part deals with the derived X-Pro model from [KSG<sup>+</sup>06], and aims to provide and understanding of how a model of a quad rotor helicopter can be derived. Besides from a model resumé, the following chapters also include actual model augmentation aswell as augmented module verification. Four chapters reside in this part and the first provides a modelling overview, whilst the following handle the Motor-Gear, Rotor and Body model.*

# CONTENTS OF PART II

---

<b>5</b>	<b>Modelling overview</b>	<b>39</b>
<b>6</b>	<b>Motor-Gear model</b>	<b>41</b>
6.1	Complete motor model . . . . .	41
6.2	Gearing . . . . .	42
6.3	Verification of Motor-Gear model . . . . .	45
6.4	Conclusion . . . . .	45
<b>7</b>	<b>Rotor model</b>	<b>47</b>
7.1	Forces and Torques . . . . .	47
7.2	Linearization of induced velocity . . . . .	51
7.3	Verification of rotor model . . . . .	51
7.3.1	Linearization . . . . .	51
7.3.2	Angular velocity . . . . .	52
7.4	Conclusion . . . . .	53
<b>8</b>	<b>Body model</b>	<b>55</b>
8.1	Kinematics . . . . .	55
8.2	Dynamics . . . . .	56
8.2.1	Forces . . . . .	56
8.2.2	Moments . . . . .	58
8.3	The complete body model . . . . .	58
8.4	Verification of body model . . . . .	60
8.5	Converting to Quaternions . . . . .	62
8.6	Conclusion . . . . .	63

---





# MODELLING OVERVIEW

This chapter will provide an understanding of how the non-linear model from [KSG<sup>+</sup>06], is constructed and how the model parts interact. Furthermore it will discuss simplifications in the model and why these are acceptable. For closing, issues concerning model augmentation and verification will be dealt with.

The system can be divided into three subsystems as depicted in Figure 5.1.

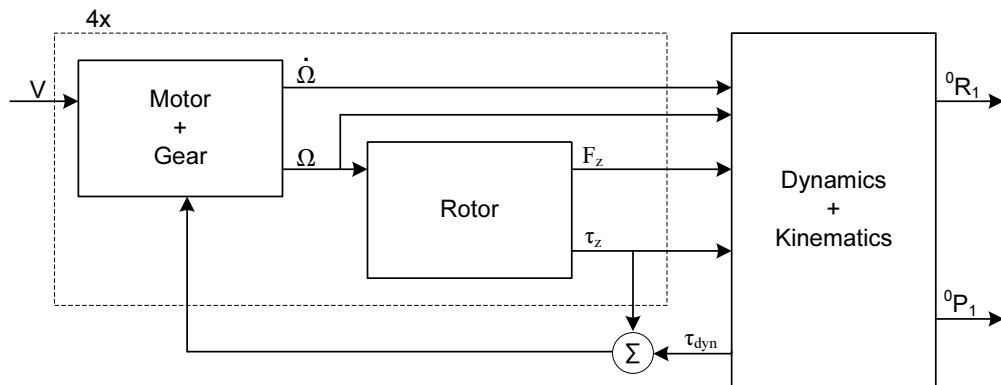


FIGURE 5.1: Modelling overview of the X-Pro divided into three subsystems, motor-gear, rotor and body, [KSG<sup>+</sup>06, p. 69].

As can be observed from the Figure, the controllable inputs to the X-Pro are the four motor voltages  $V$ . The outputs from the model are the X-Pro attitude  ${}^0R_1$  and position  ${}^0P_1$ . There are three subsystems: motor-gear, body and rotor. The interfaces between these will be described in the following.

The angular velocity,  $\Omega$ , produced by the motor and reduced by the gear is lead to both the rotor and body block. The angular acceleration  $\dot{\Omega}$  is lead to the body block in order to calculate the body moments which eventually leads to attitude. It should be noted that angle of attack and linear velocity of the body, [KSG<sup>+</sup>06, p. 19], which specifies angle of relative wind and disymmetric lift respectively, has been omitted as rotor inputs. This is a natural and acceptable simplification as the main objective is to operate in a hover state.

The rotor output is a combination of force  $F_z$  and torque  $\tau_z$ . The generated rotor force,  $F_z$ , is lead to the body block and consists only of the force component in the  $z$  axis direction. The force contributions along the  $x$  and  $y$  axis ideally neutralize each other in hover and the resulting  $F_z$  vector is according to the latter considered an acceptable simplification. The generated rotor torque,  $\tau_z$ , is lead not only to the body block, but also

---

back to the motor. The torque is directed about the  $z$  axis.

The four mentioned inputs to the body block,  $\dot{\Omega}$ ,  $\Omega$ ,  $F_z$  and  $\tau_z$  are necessary in order to calculate the rotational position  ${}^0_1R$  of the X-Pro relative to the universal frame  $\{0\}$ , and the translational position  ${}^0P_1$  from frame  $\{1\}$  relative to frame  $\{0\}$ . These designated outputs from the body, functions as the X-Pro system outputs aswell.

Last but not least should be stated that it is not only the rotor torque which is fed back to the motor, but also the torque produced by the X-Pro dynamics. These two torque contributions are summarized to comprise the total load torque seen by the motor.

For closing, it will prove valuable to discuss possible model augmentation and verification. It is desired to be able to guarantee a certain angular velocity of the gear, thereby rotor, as this maps directly to lift. From [KSG<sup>+</sup>06, p. 105] it was shown that the angular velocity of the rotor deviated from the model with a small gain factor. The gain factor was later identified to lie within the motor-gear model. In [KSG<sup>+</sup>06] both motor and motor-gear were verified through current comparisons. If the verification domain was augmented to contain an angular velocity verification for the motor-gear and motor-gear-rotor, it will be possible to guarantee a certain angular velocity.

The body model is according to Figure 5.1 on the preceding page based on the Direct Cosine Matrix (DCM) matrix. With implementational aspects in mind, integration of quaternions instead of DCM in the body model will lighten the computational burden and reduce quantization issues to a minimum.

The before stated issues will be integrated in the forth coming chapters and serve as new results to the already developed model. The following chapter will handle the motor-gear part.



# MOTOR-GEAR MODEL

# 6

This chapter concerns the modelling of the motor-gear parts mounted on the four carbon fiber arms of the X-Pro. The motors are of the brushed permanent magnet Direct Current type, which are found in many industrial robotic applications [Cra05, p. 278]. For specifications on the motor, turn to the Datasheet which can be found on the CD-ROM in /Datasheets/rs\_545sh.pdf.

In order to obtain a fairly accurate model, the motor is tested separately in the laboratory, completely isolated from the gear and rotor. The test records can be located in Appendix B.1, [KSG<sup>+</sup>06, pp. 139]. These records along with the Datasheet of the motor serves as basis for the motor modelling according to [AP05, pp. 8].

The model is divided into an electrical and mechanical part, which are dealt with separately. These parts are then combined into the complete model for the motor, which eventually is tested and verified. The motor parts and their combining are described in Appendix A, [KSG<sup>+</sup>06, pp. 133]. Consequently this chapter will not concern the outlining of the model but instead give its emphasis on the complete resulting motor model. Furthermore the final part of this chapter will integrate the gear into the motor model and verify the motor-gear model with respect to its angular velocity output.

## 6.1 COMPLETE MOTOR MODEL

The electrical and mechanical part of the motor are combined into the complete motor model according to Appendix A, [KSG<sup>+</sup>06, pp. 133]. The result of this is depicted in Figure 6.1.

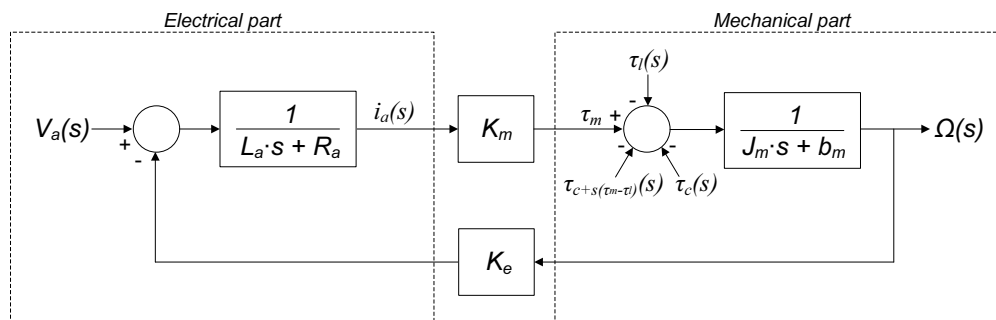


FIGURE 6.1: Block diagram of combined electrical and mechanical part.

Worth noticing from the Figure is that the viscous friction is integrated into the mechani-

cal system due to its linearity. Stiction which is only active at zero velocity and coulomb friction which is an offset friction, are both added conveniently in Simulink<sup>®</sup>. Refer to the simulink file `/matlab/model/motor/MotorModel.mdl` located on the CD-ROM for a depiction of this. At this point the torque load remains an unknown torque produced by the rotor and body dynamics.

The various parameters and constants devised from the Datasheet and the test journal in Appendix B.1, [KSG<sup>+</sup>06, pp. 139], are listed in Table 6.1 and can be reproduced through the m-file `matlab/model/motor/MotorExperiment.m` located on the CD-ROM.

Parameter	Description	Value	Unit
$L_a$	Armature inductance	0.2811	mH
$R_a$	Armature resistance	0.2094	$\Omega$
$K_m$	Electric constant	0.0046	$\frac{N \cdot m}{A}$
$K_e$	Torque constant	0.0046	$\frac{N \cdot m}{A}$
$J_m$	Inertia of motor	$2.6200 \cdot 10^{-6}$	$\frac{\dot{\omega}}{\tau}$
$b_m$	Viscous friction	$9.9438 \cdot 10^{-7}$	$\frac{N \cdot m}{rad/sec}$
$\tau_{s,m}$	Stiction	0.0071	$N \cdot m$
$\tau_{c,m}$	Coulomb friction	0.0029	$N \cdot m$

TABLE 6.1: Motor parameters and constants.

Verification of the motors have been carried out by applying measured inputs, from the motor tests, to the model and comparing system current with model current. Two input scenarios have been used, a classical step input and a more dynamic input. The verification document can be located at [KSG<sup>+</sup>06, p. 22-23].

The next component, namely the gear combining motor with rotor can now be described.

## 6.2 GEARING

As indicated by Figure 5.1 on page 39, the motor is connected through a gear to the inertial load produced by the rotor. The gearing which is driven by a belt, can be seen as a system combining transmission with speed-reduction. Transmission in the sense that the motion is transmitted from actuator to joint, and speed-reduction due to the larger radius at the joint.

The gear ratio  $\eta$ , of this gear type is calculated by Equation (6.1), where  $n_{gi}$  is the

number of teeth of the input actuator pulley and  $n_{go}$  is the number of teeth of the output joint pulley:

$$\eta = \frac{n_{go}}{n_{gi}} \tag{6.1}$$

The gear ratio describes the speed-reducing and torque increasing effects of the gear pair. When observing the X-Pro, the joint gear is larger than the motor-gear, hence  $\eta > 1$  and therefore a reduction of speed and an increase in torque is apparent. These relationships are given by Equation (6.2) and (6.3), [Cra05, p. 246].

$$\Omega_{go} = \frac{1}{\eta} \cdot \Omega_{gi} \tag{6.2}$$

$$\tau_{go} = \eta \cdot \tau_{gi} \tag{6.3}$$

where  $\omega_{gi}$  and  $\omega_{go}$  are input and output angular velocities, and  $\tau_{gi}$  and  $\tau_{go}$  are the input and output torques.

When modelling a gearing system of the mentioned type, which increases the torque to a desired level, one also has to recon the added disadvantages. Especially friction and flexibility are introduced into the system. These phenomenons along with various delimitations of the gear system are described in [KSG<sup>+</sup>06, p. 24-25]. A block diagram of the motor and gear model is shown in Figure 6.2.

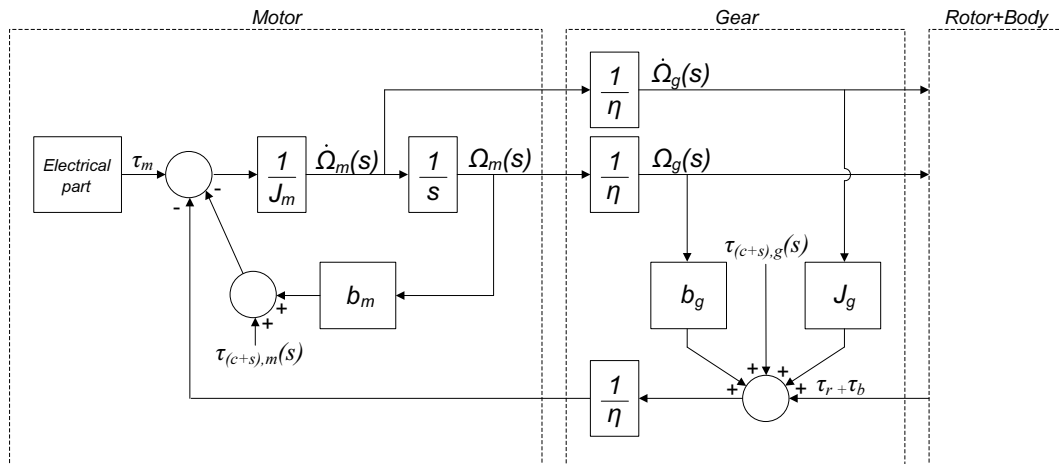


FIGURE 6.2: Block diagram of the motor-gear model. For clarity reasons the feedback from the mechanical part of the motor to the electrical part is excluded.

As can be seen from Figure 6.2, the electrical part of the motor is encapsulated in one box and the feedback from the mechanical part of the motor, excluded. This is done to

simplify the figure and only emphasize the important parts, namely the connection between motor and gear. It can be observed that in order to get from motor to gear or vice versa a multiplication of the gear ratio  $1/\eta$  must be carried out complementing the Equations (6.2) and (6.3). Through manipulation one can arrive at the following block diagram shown in Figure 6.3. This constellation enables a parameter identification similar

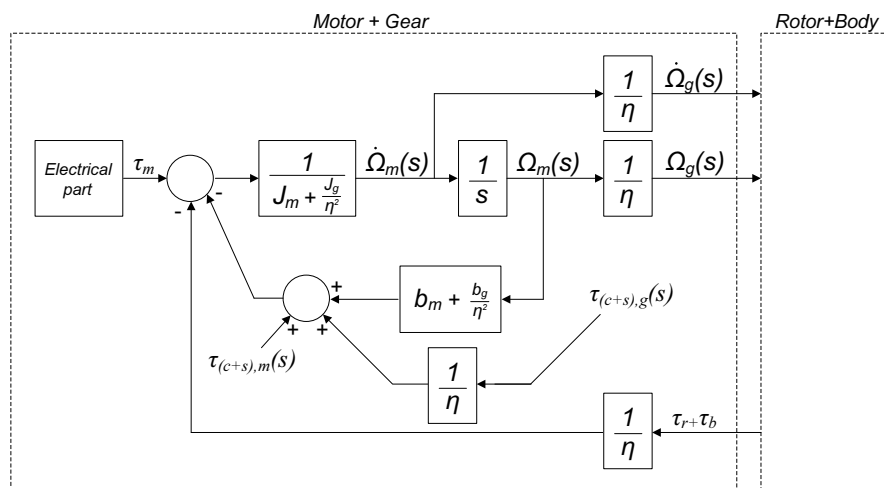


FIGURE 6.3: Block diagram of gear integrated into the motor model. For clarity reasons the feedback from the mechanical part of the motor to the electrical part is excluded.

to the one carried out for the single motor. The Simulink<sup>®</sup> model is located on the CD-ROM at `matlab/model/motor/MotorGearModel.mdl`. The various parameters and constants devised from the Datasheet and the test journal in Appendix B.2, [KSG<sup>+</sup>06, p. 143-144], are listed in Table 6.2 on the facing page and can be reproduced through the m-file `matlab/model/motor/MotorGear Experiment.m` located on the CD-ROM.

As Table 6.2 indicates it is the total friction contributions which are listed. With respect to Figure 6.3,  $b_{tot,i}$  comprises  $(b_m + \frac{b_g}{\eta^2})$  and  $\tau_{ctot}$  includes  $(\tau_{c,m} + \frac{1}{\eta}\tau_{c,g})$ . The stiction  $\tau_{Stot}$  which is the same for all motors contains  $(\tau_{s,m} + \frac{1}{\eta}\tau_{s,g})$ .

Verification of the motor-gear systems have been carried out in the same manner as for the single motor. The verification document can be located at [KSG<sup>+</sup>06, p. 25-26]. As indicated earlier the motor and motor-gear systems have previously been verified by comparing system current with model current. The next section augments the verification domain by examining the actual model output, angular velocity  $\Omega_g$ .

Parameter	Description	Value	Unit
$\eta$	Gear ratio	10	-
$J_{tot}$	Motor-gear inertia	$2.7035 \cdot 10^{-6}$	$\frac{\dot{\omega}}{\tau}$
$\tau_{s,tot}$	Total stiction	0.0120	$N \cdot m$
$b_{tot,f}$	Viscous friction, front motor	$2.1176 \cdot 10^{-6}$	$\frac{N \cdot m}{rad/sec}$
$\tau_{c_{tot},f}$	Coulomb friction, front motor	0.0076	$N \cdot m$
$b_{tot,l}$	Viscous friction, left motor	$2.1176 \cdot 10^{-6}$	$\frac{N \cdot m}{rad/sec}$
$\tau_{c_{tot},l}$	Coulomb friction, left motor	0.0056	$N \cdot m$
$b_{tot,b}$	Viscous friction, back motor	$9.7948 \cdot 10^{-6}$	$\frac{N \cdot m}{rad/sec}$
$\tau_{c_{tot},b}$	Coulomb friction, back motor	0.0208	$N \cdot m$
$b_{tot,r}$	Viscous friction, right motor	$9.7948 \cdot 10^{-6}$	$\frac{N \cdot m}{rad/sec}$
$\tau_{c_{tot},r}$	Coulomb friction, right motor	0.0158	$N \cdot m$

TABLE 6.2: Motor-gear parameters and constants.

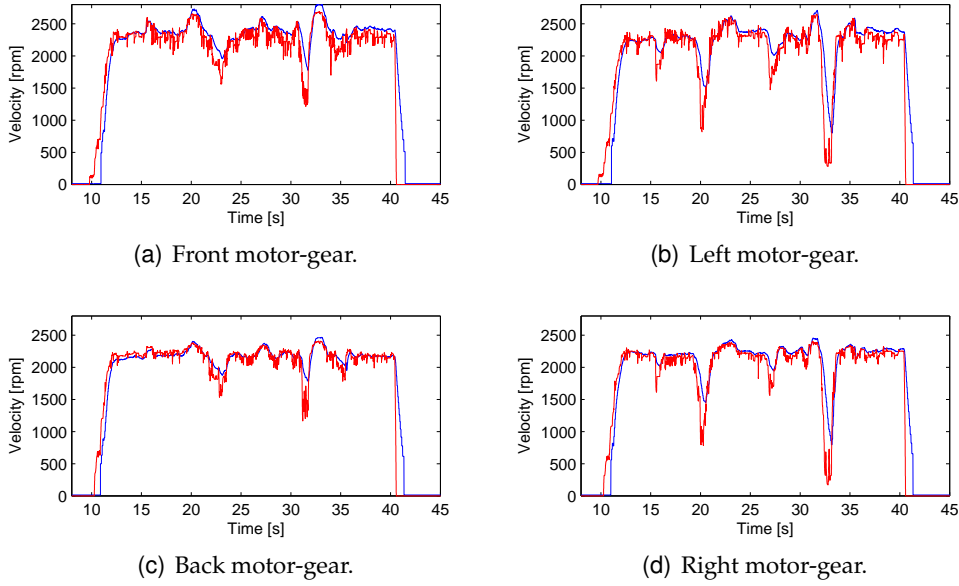
### 6.3 VERIFICATION OF MOTOR-GEAR MODEL

Table 6.2 indicates profound deviations in the coulomb and viscous friction of the four motor-gear systems. The parameters have been adapted to fit each motor-gear system individually, but can and will however deviate over time as a function of wear. The result can be observed in Figure 6.4.

The Figure documents a decisive resemblance between the model outputs and the actual motor-gear velocity. Around 32s an emphatic drop in velocity is observed in all four motors along with a certain model deviation. This can however be explained by the following. The X-Pro power drive has been designed as a one way drive, which indicates that the angular velocity of the rotor only can be controlled in the velocity increasing direction. Hence, when a sudden deceleration occurs, the decreasing of rotor velocity solely depends on the viscous friction in the motor-gear. It is the result of this that constitutes the deviation observed in the Figure. This is noted and considered an acceptable deviation. It is furthermore noted that the significance of this phenomenon will decrease when the rotors are added, increasing the friction and thereby balancing out the differences in acceleration and deceleration.

### 6.4 CONCLUSION

This chapter concerns the modelling of the four motor-gear parts of the X-Pro. When observing Figure 6.1 on page 41 the following resulting equations (6.4) and (6.5)



**FIGURE 6.4:** Verification of the four motor-gear velocities. The blue graphs are measured output, whilst the red is model output.

can be derived.

$$i_a(s) = \frac{V_a(s) - K_e \cdot \Omega_m(s)}{L_a \cdot s + R_a} \quad (6.4)$$

$$\Omega_m(s) = \frac{K_m \cdot i_a(s) - \tau_f(s) - \tau_l(s)}{J_m \cdot s} \quad (6.5)$$

The resulting equation for  $\Omega_g$  can according to Figure 6.3 page 44 be derived in (6.6).

$$\Omega_g(s) = \frac{K_m \cdot i_a(s) - \tau_{f_{tot}}(s) - \tau_l(s)}{\eta \cdot (J_m + \frac{J_g}{\eta^2}) \cdot s} \quad (6.6)$$

with the total frictional torque defined as in (6.7)

$$\tau_{f_{tot}}(s) = \tau_{(c+s),m}(s) + \frac{1}{\eta} \cdot \tau_{(c+s),g}(s) + (b_m + \frac{b_g}{\eta^2}) \cdot \Omega_m(s) \quad (6.7)$$

The motor-gear verification has been augmented to contain a verification of the actual model output, angular velocity of the gear. This has been done because it is desired to be able to guarantee a certain velocity more than a specific current level. Figure 6.4 on page 46 verifies that the developed model with the parameters listed in Table 6.2 on page 45 can guarantee a certain velocity matching the velocity of the real motor-gear system. Equations (6.4)-(6.7) along with the latter concludes this chapter.





# ROTOR MODEL

*This chapter concerns the modelling of the rotor blade dynamics. A basic knowledge of how a rotor is provided the ability to produce lift, is necessary to model the blades correctly. Blade element theory should furtherly be applied to explain how elementary forces are exposed to a rotor blade, based on its layout and air velocity exposed to it. Knowledge about the previous issues can be acquired at [KSG<sup>+</sup>06, p. 29-36] and it is through this, equations for the forces and torques provided by the rotor can be initiated. The final part of the chapter will concern a linearization of the model in the hover state and a rotor model verification.*

The rotor movement in the X-Pro environment is generally affected by three inputs. First and most intuitive is the angular velocity generated by the motor. However a component involving the X-Pros actual angular velocity would have to be added or subtracted to this input. The linear velocity as well plays an important role as it would result in dissymmetric lift. Another input one must consider is the angle of attack which basically specifies the angle of relative wind, thereby affecting the rotor.

However as the X-Pro flight is restricted to hover, the only remaining input is the angular velocity,  $\Omega$ , produced by the motor. This allows the following Input/Output (I/O) relation in Figure 7.1.

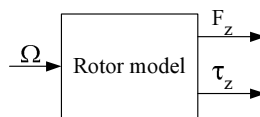


FIGURE 7.1: Illustration of the I/O of the linearized rotor model.

The Figure shows two output vectors, the lift force  $F$  and the torque  $\tau$ . In the following section, the equations comprised by these outputs will be derived.

## 7.1 FORCES AND TORQUES

Each rotor generates forces acting outside the origin and produces torques acting about the origin. The forces  ${}^rF_x$ ,  ${}^rF_y$  and  ${}^rF_z$  are parallel to their corresponding axes whereas the torques are as illustrated in Figure 7.2 on the following page.

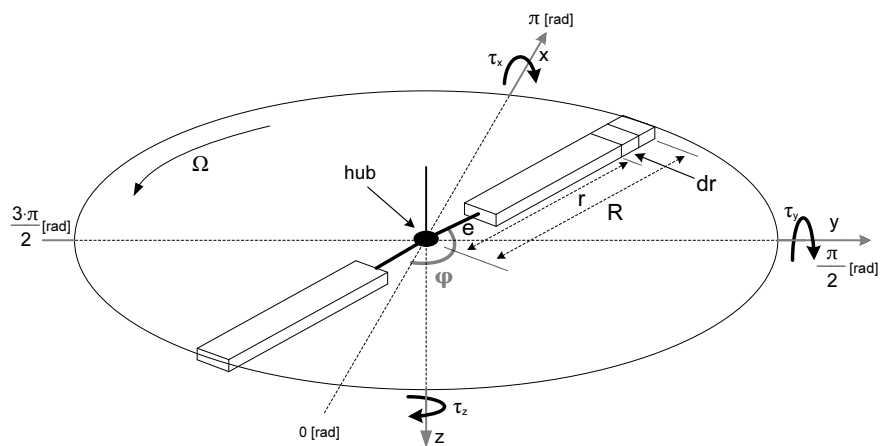


FIGURE 7.2: Illustration of torques around the  $x$ -,  $y$ - and  $z$ -axes.

Further a cross-section of the rotor-blade is shown in Figure 7.3 [Pro89, pp. 157-159].

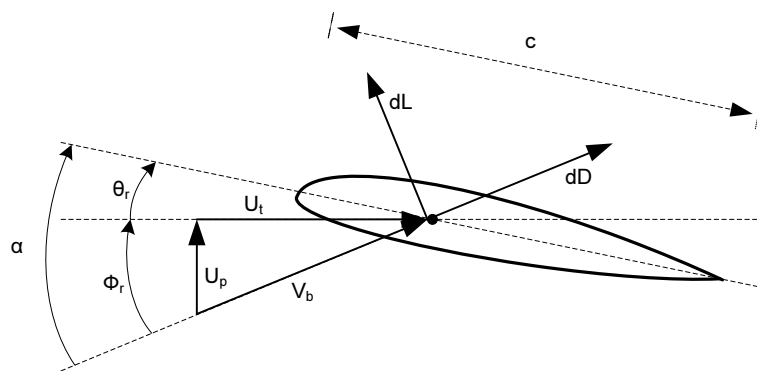


FIGURE 7.3: Rotor blade section flow conditions in flight.

As the figure shows, there are two elementary forces that depend on the velocity components,  $U_t$  and  $U_p$ , that are exposed to the rotor blade. These are the lift,  $dL$ , and the drag,  $dD$ , force. The angle of attack,  $\alpha$ , is assembled by two angles, the blade pitch  $\theta_r$  and the inflow angle  $\phi_r$ . Finally  $V_b$  is the blade velocity and  $c$  is the chord length.

The forces generated and the torques produced by the rotors depend on the form of the rotor blade. Its form has influence on its potential to produce lift, therefore the rotor blade has been divided into simplified forms, which are shown in Figure 7.4 on the facing page.

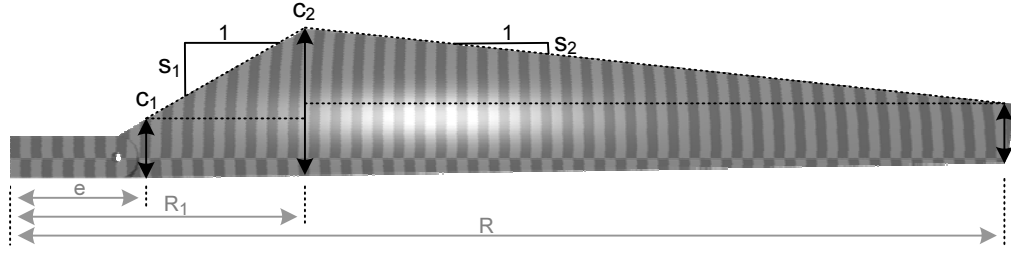


FIGURE 7.4: Rotor blade.

The form of the rotor blade is used when deriving the rotor model. A mathematical representation of the rotor blade is shown in Equation (7.1), see Appendix C [KSG<sup>+</sup>06, p. 145] for further details.

$$e \rightarrow R_1 : c(r) = c_1 + s_1 \cdot r \quad (7.1a)$$

$$R_1 \rightarrow R : c(r) = c_2 + s_2 \cdot r \quad (7.1b)$$

Where  $r$  is the radius to the blade section  $dr$ , see Figure 7.2 on the preceding page.

When Equation (7.1) is inserted in the rotor model, Equation (7.2) and (7.3) are the final result based upon the X-Pro's rotor blades. These equations has been derived in [KSG<sup>+</sup>06, p. 36-42]. Equation (7.2) shows the forces parallel to the x-, y- and z-axes.

$$\begin{aligned} {}^r F_x = & \frac{b}{2\pi} \frac{1}{2} \rho \int_0^{2\pi} \left[ \int_e^{R_1} U_t^2 \cdot (c_1 + s_1 r) \cdot (\phi_r \cdot a \cdot (\theta_r + \frac{U_p}{U_t}) - C_D) dr \right. \\ & \left. + \int_{R_1}^R U_t^2 \cdot (c_2 + s_2 r) \cdot (\phi_r \cdot a \cdot (\theta_r + \frac{U_p}{U_t}) - C_D) dr \right] \cdot \sin(\varphi) d\varphi \end{aligned} \quad (7.2a)$$

$$\begin{aligned} {}^r F_y = & \frac{b}{2\pi} \frac{1}{2} \rho \int_0^{2\pi} \left[ \int_e^{R_1} U_t^2 \cdot (c_1 + s_1 r) \cdot (-\phi_r \cdot a \cdot (\theta_r + \frac{U_p}{U_t}) + C_D) dr \right. \\ & \left. + \int_{R_1}^R U_t^2 \cdot (c_2 + s_2 r) \cdot (-\phi_r \cdot a \cdot (\theta_r + \frac{U_p}{U_t}) + C_D) dr \right] \cdot \cos(\varphi) d\varphi \end{aligned} \quad (7.2b)$$

$$\begin{aligned} {}^r F_z = & -\frac{b}{2\pi} \frac{1}{2} \rho \cdot a \cdot \int_0^{2\pi} \left[ \int_e^{R_1} U_t^2 \cdot (c_1 + s_1 r) \cdot (\theta_r + \frac{U_p}{U_t}) dr \right. \\ & \left. + \int_{R_1}^R U_t^2 \cdot (c_2 + s_2 r) \cdot (\theta_r + \frac{U_p}{U_t}) dr \right] d\varphi \end{aligned} \quad (7.2c)$$

The equations in (7.3) describes the torques around the corresponding axes, which

also is shown in Figure 7.2 on page 48.

$${}^r\tau_x = \frac{b}{2\pi} \frac{1}{2} \cdot \rho \cdot a \cdot \int_0^{2\pi} \left[ \int_e^{R_1} r \cdot U_t^2 \cdot (c_1 + s_1 r) \cdot \left(\theta_r + \frac{U_p}{U_t}\right) dr + \int_{R_1}^R r \cdot U_t^2 \cdot (c_2 + s_2 r) \cdot \left(\theta_r + \frac{U_p}{U_t}\right) dr \right] \cdot \sin(\varphi) d\varphi \quad (7.3a)$$

$${}^r\tau_y = -\frac{b}{2\pi} \frac{1}{2} \cdot \rho \cdot a \cdot \int_0^{2\pi} \left[ \int_e^{R_1} r \cdot U_t^2 \cdot (c_1 + s_1 r) \cdot \left(\theta_r + \frac{U_p}{U_t}\right) \cdot dr + \int_{R_1}^R r \cdot U_t^2 \cdot (c_2 + s_2 r) \cdot \left(\theta_r + \frac{U_p}{U_t}\right) \cdot dr \right] \cdot \cos(\varphi) d\varphi \quad (7.3b)$$

$${}^r\tau_z = \frac{b}{2\pi} \frac{1}{2} \cdot \rho \int_0^{2\pi} \left[ \int_e^{R_1} r \cdot U_t^2 \cdot (c_1 + s_1 r) \cdot \left(\phi_r \cdot a \cdot \left(\theta_r + \frac{U_p}{U_t}\right) - C_D\right) dr + \int_{R_1}^R r \cdot U_t^2 \cdot (c_2 + s_2 r) \cdot \left(\phi_r \cdot a \cdot \left(\theta_r + \frac{U_p}{U_t}\right) - C_D\right) dr \right] d\varphi \quad (7.3c)$$

The unknown constants in the equations above are listed in Table 7.1.

Parameter	Description	Value	Unit
${}^rF_x$	Output from rotor model	-	N
${}^rF_y$	Output from rotor model	-	N
${}^rF_z$	Output from rotor model	-	N
${}^r\tau_x$	Output from rotor model	-	Nm
${}^r\tau_y$	Output from rotor model	-	Nm
${}^r\tau_z$	Output from rotor model	-	Nm
$\Omega$	Input from motor-gear	-	rad/s
$b$	Numbers of blades on the rotor	2	-
$\rho$	Density of the air	1.293	kg/m <sup>3</sup>
$e$	Disk width from hub to rotor blade	0.034	m
$v_i$	Induced velocity through rotor disk	3.7078	m/s
$c_1$	Cross-section 1 of rotor blade	0.015	m
$c_2$	Cross-section 2 of rotor blade	0.042	m
$s_1$	Slope constant 1 for rotor blade	0.56	m
$s_2$	Slope constant 2 for rotor blade	-0.125	m
$\theta_r$	Blade pitch angle	0.44	rad/s
$R_1$	Disk width from hub to middle of rotor disk	0.082	m
$R$	Disk width from hub to end of rotor disk	0.26	m
$C_D$	The drag constant	-0.1426	-
$a$	The rotor blades lift slope	2.2815	-

TABLE 7.1: Parameters of Equation (7.2) and (7.3).

With the above equations the rotor model is derived and furtherly linearized in the following section.

## 7.2 LINEARIZATION OF INDUCED VELOCITY

The rotor model has been linearized in hover, with respect to the induced velocity  $v_i$ , see [KSG<sup>+</sup>06, p. 42]. When linearizing there are two unknown parameters, namely  $a$  and  $C_D$ . These are estimated in Appendix C, [KSG<sup>+</sup>06, pp. 145], by using the results from the test journal in Appendix D, [KSG<sup>+</sup>06, pp. 155]. The linearization of Equation (7.2) (7.3) gives that  ${}^r F_x$ ,  ${}^r F_y$ ,  ${}^r \tau_x$  and  ${}^r \tau_y$  cancel out. This results in that only  ${}^r F_z$  and  ${}^r \tau_z$  contributes to the lift and drag, respectively. Finally these have been simplified in Appendix C.2, [KSG<sup>+</sup>06, p. 148], and C.3, [KSG<sup>+</sup>06, p. 151], and their final form is shown here:

$${}^r F_z = a_{F_z} \Omega^2 + b_{F_z} \Omega \quad (7.4a)$$

$${}^r \tau_z = a_{\tau_z} \Omega^2 + b_{\tau_z} \Omega + c_{\tau_z} \quad (7.4b)$$

## 7.3 VERIFICATION OF ROTOR MODEL

To verify that the linearization is correct a structural analysis of Equation (7.4) is made. Further a verification of the rotor velocity is performed augmenting the verification domain.

### 7.3.1 LINEARIZATION

This section contains the results of structural analysis of Equation (7.4). The structural analysis can be found in [KSG<sup>+</sup>06, p. 45-47].

Figure 7.5 on the following page shows the verification of the linearization of  ${}^r F_z$ .

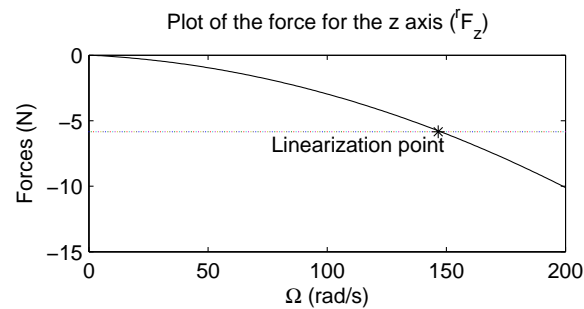


FIGURE 7.5: Illustration of the force acting parallel to the z axis.

The \* shows the linearization point and the line is the plotted version of Equation (7.4a). As can be seen the line crosses the \* which verifies the correctness of the linearization.

Figure 7.6 shows the verification of linearization of  ${}^r\tau_z$

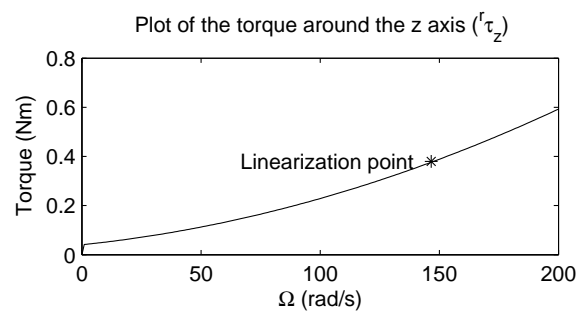


FIGURE 7.6: Illustration of the torque revolving around the z axis.

The \* shows the linearization point and the line is the plotted version of Equation (7.4b). As can be seen the line crosses the \* which verifies the correctness of the linearization.

### 7.3.2 ANGULAR VELOCITY

The rotor model has been inserted into the Simulink<sup>®</sup> version of the motor gear. The combined model has been tested to see whether the angular velocity can be verified. The Simulink<sup>®</sup> model can be found on the CD-ROM at `/matlab/model/motor/MotorGearRotorModel.mdl` and the corresponding MATLAB<sup>®</sup> script file is `/matlab/model/motor/MotorGearRotorExperiment.m`. Equation (7.4b) is inserted into the motor gear model as a MATLAB<sup>®</sup> function, and is seen as a load torque for the motor to overcome.

The modeled block that is to be verified has torque and lift force out, the verification

of the lift force is though not verified here directly. From [KSG<sup>+</sup>06, pp. 155] a test is performed to estimate the lift as a function of the angular velocity of the rotor. This test is tough also dependent on the density of the air at the test instance. A change in density will map linear to the lift and drag.

$$\rho = \frac{p}{R \cdot T} \quad (7.5)$$

$$R_{dry,air} = 287.05 J/kg \cdot K \quad (7.6)$$

In the formula the  $\rho = 1.293g/L$  was used this can of cause change according to the temperature and the air pressure. The extremes in the air pressure over Denmark are  $94.39 kPa - 106.25 kPa$  set in year 1907 January and February respectively. The chosen temperature extremes can be set to  $5^{\circ}C - 30^{\circ}C$ . Worst case the figures of  $\rho$  can then be calculated:

$$\rho_{min} = \frac{p_{max}}{R \cdot T_{min}} = 0.9698g/L \quad (7.7)$$

$$\rho_{max} = \frac{p_{min}}{R \cdot T_{max}} = 1.3307g/L \quad (7.8)$$

These values of  $\rho$  will influence the mapping of angular rotor speed to lift (and drag) with 25% at low density and 2.9% at high air density. These relative small deviations are given in weather conditions that does not allow flying, thus the influence of  $\rho$  is taken as a constant.

Given the air density is constant a verification of the angular velocity of the rotor will provide the basis of the verification of the lift force.

Figure 7.7 on the following page shows the four rotor speeds, when applying measured duty cycles and feeding these through a map that calculates the corresponding voltage.

The dynamics of the motor-gear-rotor model reflects the real system. As can be seen on the figure all the measured velocities fits the motor-gear-rotor model. The deviation in the beginning of the simulation is caused by, that the tachometer driver does not give a velocity before the rotors exceed 8 Hz i.e. 480 rpm.

## 7.4 CONCLUSION

This chapter deals with the derivation of the rotor model. Through a linearization in hover, the model has been simplified to Equation (7.4). Equation (7.4b) has then been inserted into the motor-gear model to verify the angular velocity.

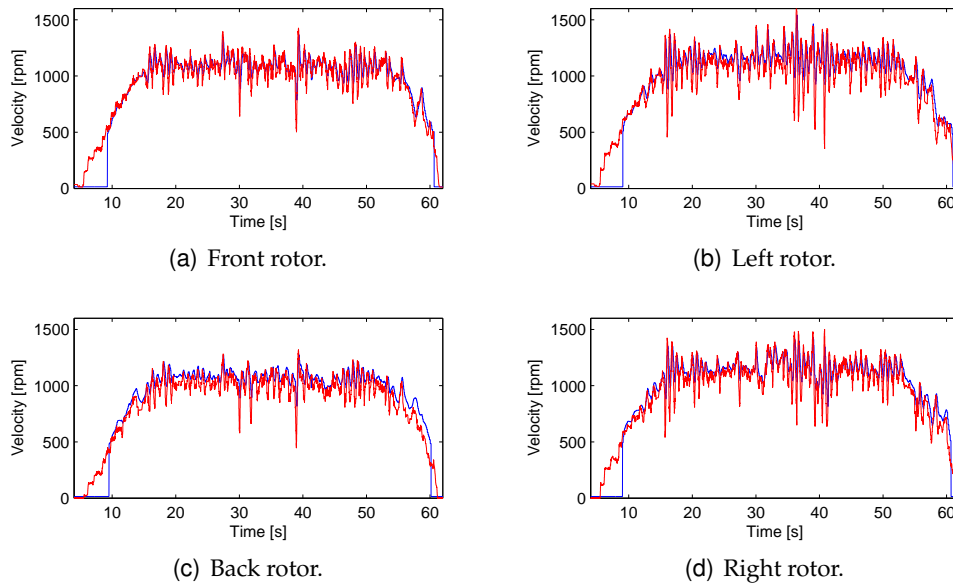


FIGURE 7.7: Verification of the four rotor velocities. The blue graphs are measured output, whilst the red is model output.

Equation (7.4a) calculates the force acting on the z-axis and again here:

$${}^r F_z = a_{F_z} \Omega^2 + b_{F_z} \Omega \quad (7.9)$$

Equation (7.4b) calculates the torque acting around the z-axis and again here:

$${}^r \tau_z = a_{\tau_z} \Omega^2 + b_{\tau_z} \Omega + c_{\tau_z} \quad (7.10)$$

The values for the constants Equation (7.9) and (7.10) can be found in Table 7.2.

Parameter	Description	Value	Unit
$a_{F_z}$	—	$-6.6399 \cdot 10^{-5} \pi$	$N \cdot s^2 / \text{rad}^2$
$b_{F_z}$	—	$-0.0029 \pi$	$\frac{N \cdot s}{\text{rad}}$
$a_{\tau_z}$	—	$8.7922 \cdot 10^{-6}$	$Nm \cdot s^2 / \text{rad}^2$
$b_{\tau_z}$	—	0.0010	$Nm \cdot s / \text{rad}$
$c_{\tau_z}$	—	0.0414	$Nm$

TABLE 7.2: Rotor parameters.

With Figure 7.1 on page 47 in mind the goal for the rotor model have been achieved, due to the fact that the force and torque in the z-axis have been derived.





# BODY MODEL

*This chapter concerns the modelling of the X-Pro body from [KSG<sup>+</sup>06, p. 50-61], with focus on the behavior when exposed to internal and external forces. The subjects dealt with in this chapter are the kinematics and dynamics. These are combined to comprise the complete body model. After the model is structurally verified, quaternions are due to implementational aspects integrated to replace the DCM.*

Since the X-Pro consists of several, individual moving parts affecting its movement, the vehicle is divided into five components when modelling its kinematics and dynamics:

**The four rotors** each rotates at an individually controllable speed. Because of this, the rotors affect the X-Pro differently, which is why they are considered as individual objects.

**The base** of the X-Pro also contributes to its movement since it has a mass and an inertia. The base can translate and rotate at a velocity individually from the rotors, by which it is considered a separate mechanical object.

Common for all of these five components are:

- They each represent a mass, which is affected by gravity. The masses must be accelerated to translate the component.
- They each represent an inertia, which must be angularly accelerated to rotate the component.
- Rotating objects produce precession, which must also be taken into account.
- All of the components are connected to each other, and thereby affect each other.

With the X-Pro's body divided into five parts the modelling of the kinematics can begin. This will be further described in the following section.

## 8.1 KINEMATICS

As described above the X-Pro is divided into five components, the base and four rotors. To describe the kinematics of the X-Pro, coordinate systems are introduced and defined in each of the five components and additionally one to describe the universe in which the X-Pro operates. This is depicted in Figure 8.1 on page 57.

Each coordinate system is referred to as a frame where every point can be described with a vector e.g. the vector  ${}^1\vec{P}_{C_1}$  describes the point (CM,1), the CM of the base, within frame {1}. The universal coordinate system named frame {0} can be placed freely but it is needed as a reference to describe the motion of the X-Pro. When the universal coordinate system has been defined, the placement of the X-Pro can be described. To describe how the base of the X-Pro is oriented, an additional coordinate system {1} is introduced in the CM of the X-Pro (CM,tot).

There are two main subjects when describing the kinematics of the X-Pro. These are linear and rotational velocity, and linear and rotational acceleration. The equations are described in [KSG<sup>+</sup>06, p. 52-55] and used in the modelling of the dynamics.

## 8.2 DYNAMICS

To be able to model the motion of the X-Pro, some basic formulas from physics are necessary. These formulas are primarily divided into two parts, forces and moments, which will be described coarsely in the following. The full dynamics outline can be located in [KSG<sup>+</sup>06, p. 55-61].

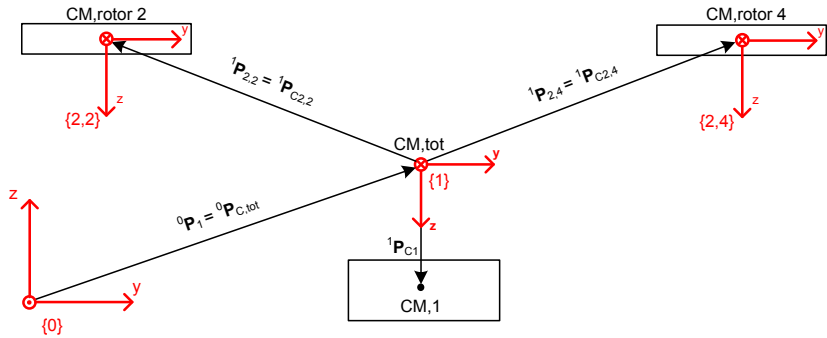
### 8.2.1 FORCES

For the X-Pro, the forces regarded are:

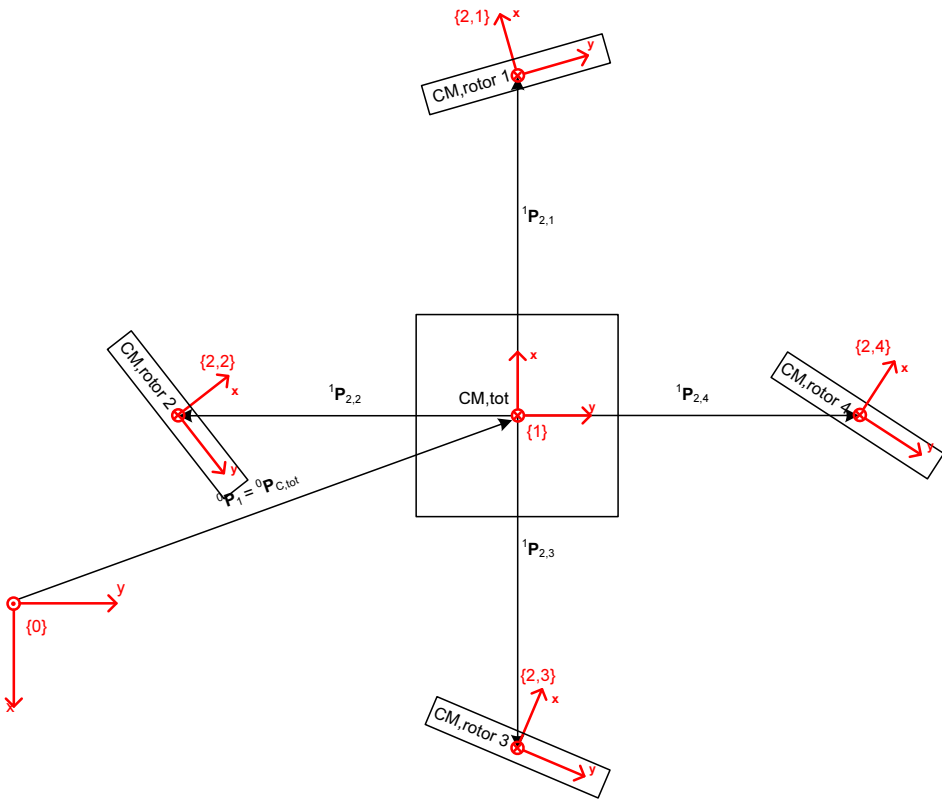
**Gravity** is ranked as a force of large influence on the X-Pro dynamics. It attacks in the CM of the X-Pro, and will always point towards the ground. Mathematically, it's direction is fixed to the orientation of the universal frame {0}.

**Rotor lift** essentially makes the X-Pro fly by surpassing gravity. The rotor lift attacks the X-Pro at the rotor hubs, and can advantageously be expressed in coordinates of the X-Pro's own frame {1}. It should be noted, that the lift of the rotors can be larger at ground level due to the increased pressure below the blades.

**Normal force** is only relevant in the case where the X-Pro is situated on the ground. In this case, forces such as friction with the ground could also be relevant. The normal forces and frictions would attack at the points of contact between the ground and the X-Pro, and can thereby be difficult to model. A model of the geometric shape of the X-Pro would be necessary to get an accurate result. Because of this, these forces are disregarded.



(a) The X-Pro seen from behind.



(b) The X-Pro from above.

**FIGURE 8.1:** Illustrations of the X-Pro from above and from behind, showing the five components of the X-Pro relative to its coordinate system {1}, placed in the CM of the entire X-Pro. The mass of the base does not translate or rotate relative to frame {1}, and is consequently described by the vector  ${}^1\vec{P}_{C_1}$ . The rotors rotate relative to frame {1}, and have consequently been assigned their own frames {2, j}.

**Air resistance** is also a force affecting the X-Pro, and is dependent on the relative wind, the surface of the X-Pro relative to the direction of the wind and pressure [SB00, p. 202]. This force is also disregarded, since it is considered highly complex to model and difficult to measure. Also, the design of the X-Pro is very slim, by which the air resistance would only be relevant at high velocities of relative wind.

## 8.2.2 MOMENTS

The external moments which act on the X-Pro body are:

**Rotor drag** from each of the rotors is an external moment, and originates from the rotors as seen in Equation (7.4b). It will normally be balanced out, since one half of the rotors rotate in the opposite direction of the other half. But this is only the case, when the rotors rotate with even speeds, which is not always the case. The moment from the rotor drag is fixed with the orientation of the X-Pro.

**Rotor lift:** If a force  $\vec{F}_j$  affects a rigid body  $i$  in a point  $\vec{P}_j$  distant from the CM of the body, the force will also produce a moment  $\vec{\tau}_j$ , which can be described as [SB00, p. 333]:

$$\vec{\tau}_j = \vec{P}_j \times \vec{F}_j \quad (8.1)$$

Thereby the rotor lift creates a moment according to (8.1), since the lift forces do not attack in the CM of the X-Pro.

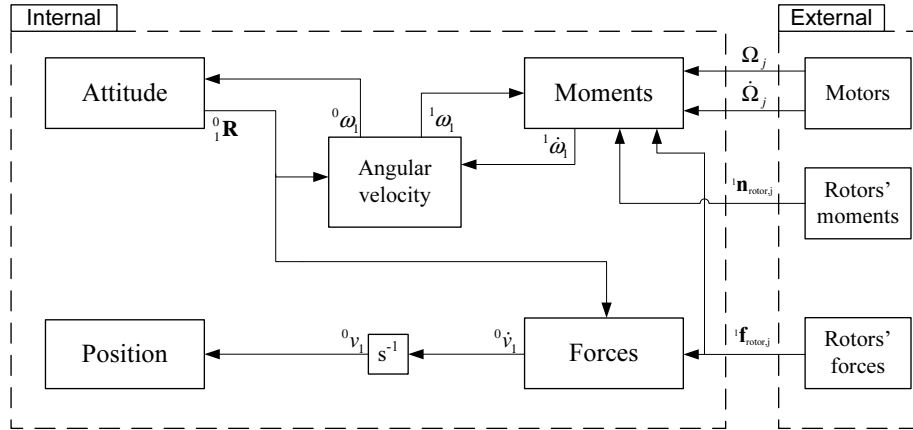
**Normal force** will create a moment, if the force does not attack in the CM of the X-Pro. The normal force has earlier been delimited, by which it is also disregarded here.

**Air resistance** will also create a moment, dampening the rotation of the X-Pro. This moment originates in the air resistance forces, acting on the individual surfaces of the X-Pro components, and has been disregarded. Consequently, the generated moment is also disregarded.

## 8.3 THE COMPLETE BODY MODEL

In the previous sections the kinematics and dynamics have been treated. These will be combined into the resulting equations representing attitude and position for the X-Pro.

Figure 8.2 on the facing page illustrates the assembly of the body model.



**FIGURE 8.2:** Block diagram of the body model where the internal block contains moments and forces generated by the X-Pro and the attitude and position which can be calculated on behalf of the inputs from the external block.

**Moments** are calculated based on the angular velocity of the X-Pro, and on the external influences of rotors and motors. The blocks **Motors**, **Rotor moments** and **Rotor forces** will not be furtherly described here. Equation (8.2) describes the relation between these factors and the angular acceleration of the X-Pro.

$${}^1\dot{\omega}_1 = {}^1I_{tot}^{-1} \left\{ \begin{array}{l} \sum_{j=1}^4 \left[ {}^1\vec{n}_{rotor,j} + {}^1\vec{P}_{2,j} \times {}^1\vec{f}_{rotor,j} \right] - {}^1\vec{\omega}_1 \times \left( {}^1I_{tot} {}^1\vec{\omega}_1 \right) \dots \\ - {}^1 \left( C_2 \tilde{I}_2 \right) {}^1\ddot{\Omega}_{2,tot} - {}^1\vec{\omega}_1 \times \left( {}^1 \left( C_2 \tilde{I}_2 \right) {}^1\vec{\Omega}_{2,tot} \right) \end{array} \right\} \quad (8.2)$$

The angular acceleration is expressed in the X-Pro's own coordinate system. If sensors for measuring angular acceleration are to be mounted on the X-Pro, their axes about which they measure will constantly be aligned with the X-Pro, and so their measurements will also be expressed in coordinates of the X-Pro.

**Attitude** calculates the orientation of the X-Pro, based on its angular velocity relative to the universal frame. The block is based on [Cra05, p. 142], which defines that:

$$\dot{R} = \lim_{\Delta t \rightarrow 0} \frac{R(t + \Delta t) - R(t)}{\Delta t} \quad (8.3)$$

In this equation, the elements of  $\dot{R}$  are calculated independently from other elements. This also applies to the individual elements of  $R(t)$  and  $\dot{R}(t)$ . In [Cra05, p.

143], the following is also defined:

$$\dot{R}R^{-1} = \begin{bmatrix} 0 & -\Omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix} = S(\vec{\Omega}) \quad (8.4)$$

$$\dot{R} = S(\vec{\Omega})R \quad (8.5)$$

Based on these equations, the attitude of the X-Pro is derived to be:

$${}^0_1\dot{R} = S({}^0\vec{\omega}_1) {}^0_1R \quad (8.6)$$

**Forces** sums the external forces acting on the X-Pro and calculates the linear acceleration in CM:

$${}^1\dot{v}_1 = \frac{\sum_{j=1}^4 [{}^1\vec{f}_{rotor,j}]}{m_{tot}} + {}^1_0R {}^0\vec{g} \quad (8.7)$$

The forces from the rotors are received externally from the **Rotor forces** block, which will not be furtherly described here. The gravity is calculated internally in the **Forces** block, by which the attitude of the X-Pro is needed, represented in  ${}^0_1R$ . The linear acceleration is expressed in the coordinates of the X-Pro's own frame  $\{1\}$ , which coincides with what would be measured, if linear accelerometers were to be mounted in the CM of the X-Pro. The linear acceleration seen from the universal frame is stated as follows.

$${}^0\dot{v}_1 = {}^0_1R \frac{\sum_{j=1}^4 [{}^1\vec{f}_{rotor,j}]}{m_{tot}} + {}^0\vec{g} \quad (8.8)$$

**Position** calculates the position of the X-Pro relative to the universal frame, based on the attitude and the linear velocity. The linear velocity is expressed in coordinates of the X-Pro's own frame. Because of this, the position of the X-Pro can be expressed as:

$${}^0\dot{P}_1 = {}^0\vec{v}_1 \quad (8.9)$$

## 8.4 VERIFICATION OF BODY MODEL

This section contains the verification of the body model of the X-Pro. The verification is merely a structural test which should investigate the overall correctness of the body

model. The test has been carried out in Appendix A on page 159. The test is based on coarse considerations regarding how the X-Pro should rotate and translate, due to certain influences such as the lift and moment acting on each rotor. The test is divided into four test cases with the following results:

**Test case 1: Throttle** In this test, the lift forces of the X-Pro  ${}^1f_{rotor,i}$  are adjusted equally, and the sum of them is set to values both above and beneath the force exerted on the X-Pro by gravity.

The test showed, that the X-Pro body model accelerated upwards for high throttle and downwards for low throttle, which was expected. The test is concluded to be successful.

**Test case 2: Pitch** The sum of lift forces was set to equalize the force of gravity in hover. The forces on the left and right arm were each set to a fourth of this sum, while the force on the front was slightly higher and on the back was slightly lower. This should represent a positive pitch. The same test was conducted with slightly higher lift on the back arm and slightly lower on the front, representing a negative pitch.

The test for positive pitch resulted in the model to accelerate positively around its pitch axis  $y$ , and to accelerate linearly backwards and slightly downwards for positive pitch, which was expected. Also, the model rotated a bit around its other axes, which is considered a result of the asymmetric inertia tensor of the entire X-Pro, due to the battery being rotated 45 degrees from the symmetric axes. This causes the model to slowly accelerate around its other axis and eventually go into an unstable spin, if the simulation had continued. Also the gyro forces produced by the rotating discs can have affected the model to rotate about other axes. The test for negative pitch showed the same as for positive pitch, except that the X-Pro accelerated negatively around its pitch axis, and accelerated linearly forwards instead. Since some of the results were expected and the other were plausible, the test is considered successful.

**Test case 3: Roll** As in test case 2, the sum of lift forces was initially balancing the force of gravity. In the test, the front and back lift forces were a fourth of the gravity, while the left lift force was slightly higher and the right lift force was slightly lower. This would compare to a positive roll. The same test was conducted with a negative roll.

As expected, the model rotated positively around its roll axis  $x$  and accelerated to its right with a slight descend. As with test case 2, the model rotated slightly

around it's other axis, presumably for the same reasons. Due to this, the test is considered successful.

**Test case 4: Yaw** In the last test case, the sum of lift forces were neutralizing the force exerted by gravity. The moment acting on the front and back arm was slightly increased, while it was slightly decreased on the left and right arm. This would represent positive yaw. Also a test for negative yaw was conducted.

In this test case, the model accelerated about it's yaw axis  $z$  as expected. Opposite to test cases 2 and 3, the model did not accelerate towards an unstable spin. Since the battery, which presumably caused the uneven rotation, is rotated about the  $z$ -axis, it will not cause angular acceleration in other directions than about the  $z$ -axis itself. On grounds of this, the test is considered successful.

## 8.5 CONVERTING TO QUATERNIONS

As indicated earlier the body model kinematics uses the DCM to rotate between frames. This solution has the advantage of being intuitive. Amongst the disadvantages are the six constraints that are needed to make sure the basis vectors in the DCM are orthogonal on each other. These constraints makes the DCM difficult to maintain, and the using of these rotations become unnecessary inefficient with respect to the number of calculations that must be performed. On the other hand the quaternions have only one constraint, which is to keep  $|q| = 1$ . This is where the quaternion has an advantage over the DCM. Amongst the disadvantages is that a quaternion is a four element vector, it can only be represented in a four dimensional space, but with the constraint, the quaternion is a point on a 3-sphere in  $R^4$ . A normal 2-sphere, consists of all points equally distanced to origo in  $R^3$ . So a 3-sphere is all points equally distanced to origo in  $R^4$ . As often in mathematics it is possible to understand and calculate three dimensional problems, but the underlying mathematics is extendable to four or more dimensions. With this understanding it is possible to grasp that the rotation by a quaternion will only rotate the vector and not change the length of the vector. In this project where the aim is to implement the controller and estimator on the Gumstix, which has limited calculating resources, it will be advantageous to choose the quaternions to represent the relations between frames.

In Appendix B on page 167 a mathematical description of the quaternions is presented, which include the definition of: quaternion, quaternion multiplication, rotation by quaternion and time derivative of a quaternion. Using these basic building blocks it



is possible to redefine the model equations so rotation is done by quaternions instead of DCM.

The new stated equations are given in the following:

**Translatory velocity** The rotor forces are described in the body frame {1} and have to be rotated to the reference frame {0}, the gravitation vector  ${}^0\vec{g}$  is already described in the reference frame thus no rotation is needed. Here the force vector is padded with zero, and treated as a quaternion, thus quaternion multiplication is used in the calculation, as defined in Equation B.9 on page 168, and where  ${}^0_1q^*$  is the complex conjugated quaternion to  ${}^0_1q$ . The Equation 8.10 will replace equation 8.8.

$$\dot{v}_1 = {}^0_1q \left[ \begin{array}{c} \sum_{n=1}^4 \frac{{}^1\vec{f}_{lift,n}}{m_{tot}} \\ 0 \end{array} \right] {}^0_1q^* + {}^0\vec{g} \quad (8.10)$$

**Attitude** The time derivative of the attitude is stated as in 8.11, where  ${}^1\vec{\omega}_1$  is the instantaneous angular velocity of the body frame seen in the body frame. Again here the quaternion multiplication is used. The Equation will replace equation 8.5.

$${}^0_1\dot{q} = \frac{1}{2} {}^0_1q \left[ \begin{array}{c} {}^1\vec{\omega}_1 \\ 0 \end{array} \right] \quad (8.11)$$

These where the listing of the needed reformulations performed on the body model.

## 8.6 CONCLUSION

In this chapter, a body model of the X-Pro has been presented, based on Newtonian mechanics. A similar model based on Lagrangian mechanics has been described in [TM04]. The two models are expressed in different coordinate systems and with different symbols, but are otherwise mathematically identical regarding the contents of the equations. This implies that the model presented is correct.

To furtherly verify the correctness of the body model, a rough test has been carried out and documented in Appendix A on page 159. Since it has not been considered possible to verify the body model in a test isolated from the motor and rotor models, the conducted test only seeks to investigate the correctness of the body model structure. The verification showed what was expected from an intuitive point of view. The expected movement and

the movement in the model were in the same direction and have the X-Pro orientated in the same way, by which the structural test of the model is assumed to be successful. Consequently, the model has proved to be correct structurally.

After the structural verification, the DCM to quaternion conversion was carried out, mainly to minimize the attitude calculations online. The model with quaternions proved to have the same behavior as with DCM, by which the DCM to quaternion conversion is verified.

# **PART III**

# **HOVER CONTROLLER**

*This part deals with the development of a controller which can stabilize the X-Pro in a hover condition. The first chapter describes the control strategy, which comprises a cascade coupling of four Proportional Integral (PI) controller and a LQ- controller. The controller heavily depends on the estimator which has the task of providing accurate estimates of the system states. Hence, the first chapter will define the interface between the estimator and controller to provide a common ground to start from. In the second chapter rotor speed controllers are designed in order to guarantee certain rotor speeds and the chapter hereafter revolves around the design of the overall controller. Finally the controller is verified and concluded upon.*

# CONTENTS OF PART III

---

<b>9</b>	<b>Control strategy</b>	<b>69</b>
9.1	Interface towards estimation group . . . . .	69
9.1.1	Statevector . . . . .	69
9.1.2	State estimate delay . . . . .	70
9.1.3	Area of validity of state estimates . . . . .	71
9.2	Control analysis . . . . .	72
9.3	Controller requirement specification . . . . .	73
9.3.1	Rotor speed controllers . . . . .	73
9.3.2	LQ- controller . . . . .	74
<b>10</b>	<b>Rotor speed controller</b>	<b>75</b>
10.1	Deriving controller equations . . . . .	75
10.2	Deriving controller parameters . . . . .	77
10.3	Conclusion . . . . .	80
<b>11</b>	<b>Linear Quadratic controller</b>	<b>81</b>
11.1	System equations . . . . .	81
11.2	Linearization . . . . .	82
11.2.1	Linearization of Translatory velocity . . . . .	83
11.2.2	Linearization of the attitude . . . . .	86
11.3	State Space form . . . . .	88
11.3.1	Operating point . . . . .	89
11.3.2	Controllability . . . . .	90
11.3.3	Discretization . . . . .	90
11.4	Deriving controller parameters . . . . .	91
11.4.1	Integral action . . . . .	91
11.4.2	Weight matrices . . . . .	93
11.5	Verification . . . . .	94
11.5.1	Disturbance issues . . . . .	94
11.5.2	Simulation . . . . .	94

11.6 Conclusion . . . . . 100

---





# CONTROL STRATEGY

*This chapter serves to provide an overview of the control strategy used to stabilize the X-Pro in a hover condition. The first section describes the controller/estimator interface which in turn stipulates certain requirements for the controller design. The next section aims to clarify how these requirements should be maintained and further to describe the applied control strategy. In order*

to attain autonomous behavior for the X-Pro, a controller must be devised. A common way of controlling a Multiple Input Multiple Output (MIMO) system is by the application of state feedback. This results in a control law which must have state estimates as input. Hence an estimator that converts sensor information into accurate estimates of the states is called for. The following Figure 9.1 illustrates the idea.

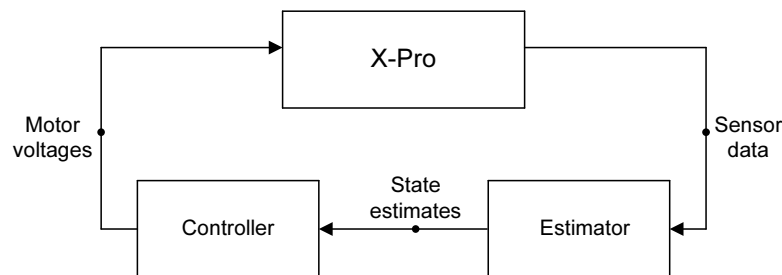


FIGURE 9.1: The main control idea stabilizing the X-Pro in hover.

As mentioned earlier on page 3 in the introduction the estimator is developed by group [GHB07]. This highly motivates a clear description of the controller/estimator interface, which is carried out in the following section.

## 9.1 INTERFACE TOWARDS ESTIMATION GROUP

The parting into two groups calls for a clean interface, which mainly concerns the state vector that is estimated, but also the delay of the state estimate is of concern. These topics are dealt with in the following.

### 9.1.1 STATEVECTOR

The estimation group provides an estimate of the state vector as stated in 9.1.

$$\vec{x} = \begin{bmatrix} \textit{position} \\ \textit{velocity} \\ \textit{small signal attitude} \\ \textit{angular velocity} \\ \textit{rotor speed} \end{bmatrix} = \begin{bmatrix} {}^0\vec{P}_1 \\ {}^0\vec{v}_1 \\ {}^0_1\tilde{q}_{1:3} \\ {}^1\vec{\omega}_1 \\ {}^1\vec{\Omega}_2 \end{bmatrix} \quad (9.1)$$

Where the individual parts are defined as in 9.2 to 9.6.

$${}^0\vec{P}_1 = \begin{bmatrix} {}^0P_{x1} & {}^0P_{y1} & {}^0P_{z1} \end{bmatrix}^T \quad (9.2)$$

$${}^0\vec{v}_1 = \begin{bmatrix} {}^0v_{x1} & {}^0v_{y1} & {}^0v_{z1} \end{bmatrix}^T \quad (9.3)$$

$${}^0_1\tilde{q}_{1:3} = \begin{bmatrix} {}^0_1\tilde{q}_1 & {}^0_1\tilde{q}_2 & {}^0_1\tilde{q}_3 \end{bmatrix}^T \quad (9.4)$$

$${}^1\vec{\omega}_1 = \begin{bmatrix} {}^1\omega_{1,x} & {}^1\omega_{1,y} & {}^1\omega_{1,z} \end{bmatrix}^T \quad (9.5)$$

$${}^1\vec{\Omega}_2 = \begin{bmatrix} {}^1\Omega_{2,1} & {}^1\Omega_{2,2} & {}^1\Omega_{2,3} & {}^1\Omega_{2,4} \end{bmatrix}^T \quad (9.6)$$

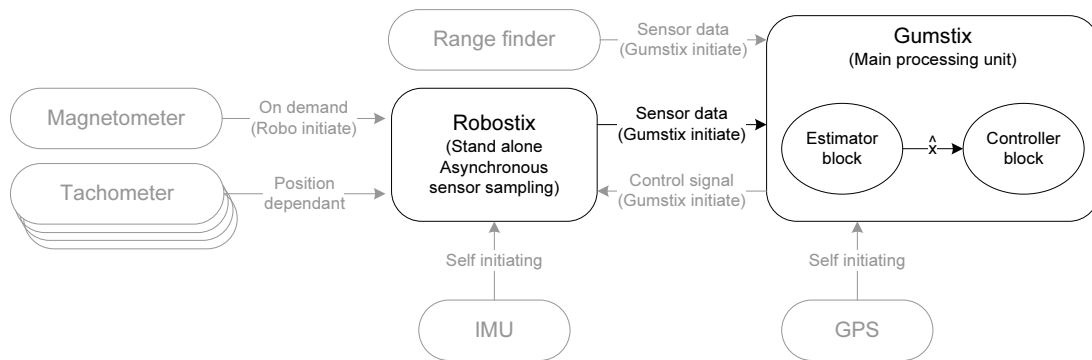
Here it is important to notice that both the angular velocity of the body and the angular rotor speed are given in body frame,  $\{1\}$ , but opposed the position and the transversal velocity which is seen from the earth frame,  $\{0\}$ . The attitude of the body is represented by the first three elements of the small signal quaternion. The fourth element  ${}^0_1\tilde{q}_4$  can always be reconstructed from the other three, and is thus left out. This leaves the problem of normalizing the quaternion to the estimation group. The subscript on the rotor speeds, e.g. denoted by  ${}^1\Omega_{2,3}$ , refer to the frame that is placed in the center of the back rotor, as defined in Figure 8.1 on page 57.

### 9.1.2 STATE ESTIMATE DELAY

Due to the hardware setup, it is unavoidable that the state estimates will be delayed. In Figure 9.2 the setup is depicted.

The platform is built in a way that a fixed delay period cannot be guaranteed, nor can the delay be kept small due to hardware issues. The problem is rooted in the way the Gumstix get sensor data over the I<sup>2</sup>C bus from the Robostix. On the Robostix the sampling of sensors are done regardless of when the Gumstix choses to request data from the Robostix. This results in a worst case delay of one sample period on the sensor data itself. On top of this, an estimator will have some convergence time which in some sense can be seen as a further delay on the state estimates. Communication from the robostix





**FIGURE 9.2:** The setup of data gathering to the Gumstix, notice that the sampling on the robostix is done asynchronous with respect to the Gumstix. this is the main cause of delay in the setup.

to the Gumstix over the I<sup>2</sup>C bus is driven at 100 Hz, thus the estimation loop is driven at the same rate. The overall delay of the state variables are expected to be maximum 20 ms, thus the developed controller should be tolerant to a delay of this magnitude. Taking these measures will help to avoid instability when integrating the two parts into the final system.

### 9.1.3 AREA OF VALIDITY OF STATE ESTIMATES

Ideally the state estimates would reflect the state variables regardless of the value of the states, but due to nonlinearities in the dynamics of the X-Pro there will be a trade off between an accurate state estimate and a wide validity range, when the estimator is built on a linearized version of the X-Pro. With this argumentation, working areas on the state variables are defined, each stating which values a given state variable can obtain, such that the state estimate can be used for control. It is important to notice that a controller can shift a state variable to a point where it surpasses the validity area, thus resulting in possible erroneous state estimates, probably causing instability. In the following listing, the constraints are given on all states.

**Position** The position does not affect the linearization as long as the X-Pro is out of the ground effect area which as a rule of thumb should be two times the rotor radius [KSG<sup>+</sup>06], thus it is not needed to setup further boundaries.

**Velocity** Due to unmodelled behavior such as the build up of pressure under the rotors, even the nonlinear model can only poorly estimate the lift when the X-Pro has transversal movement. Furthermore since the eight rotor blades rotate at a high

velocity and have the potential of causing harm to its surroundings it will not be advisable to allow high translatory velocities, making it possible for humans to intervene before an impact. Assuming the X-Pro is stationed one meter above ground, and assuming that a human being has a reaction time of one second, gives the working area for the velocity at  $\pm 0.5 \text{ m/s}$  in all directions, also making it possible for the a pilot to intervene before impact occurs.

**Small signal attitude** The attitude state is nonlinear in its basic form, which at least dictates the need for a determination of a working area. The yaw rotation is contained in the  $\tilde{q}_3$  element and turning away from the linearization point will cause the position in the x-y plane to be propagated wrongly by the linear model. The fault becomes even bigger on pitch and roll movement since the linear model can not describe that the X-Pro gains a velocity towards the ground when the attitude is changed away from the horizontal plane. A constraint of  $30^\circ$  is chosen as the deviation from the horizontal plane, and also a constraint of  $30^\circ$  is chosen on the z axis in the earth frame,  $\{0\}$ .

**Angular velocity** The IMU amongst others can estimate the angular rate, and the constraint is derived from the specifications for this unit. The limit for this sensor is  $200^\circ/\text{s}$ . Exceeding the boundaries for this unit which can have devastating consequences. A safety margin is chosen leaving the limit at  $180^\circ/\text{s} \Rightarrow \pi \text{ rad/s}$  on all axes of the body frame,  $\{1\}$ .

**Rotor speed** The design of the tachometers causes limits on the measuring area. The lower limit is set to  $8 \text{ Hz} \approx 50 \text{ rad/s}$  and the upper limit is set on the basis of what is physically possible, this being set to  $2000 \text{ rpm} \approx 210 \text{ rad/s}$ . In practice this means that only the lower limit should be actively avoided in the controller design.

## 9.2 CONTROL ANALYSIS

With the controller/estimator interface defined the controller structure can be determined. It is decided to design four PI rotor speed controllers there main functionality will be to suppress parameter differences in the motor-gear-rotor systems. This means that a uniform input to all four controllers will result in an equal rotational speed and thereby also lift. The reference to the speed controllers are generated by a LQ controller, which takes the before mentioned state estimates as input. This controller type has proved to be a good method for handling MIMO systems. The cascade coupling of the two controller

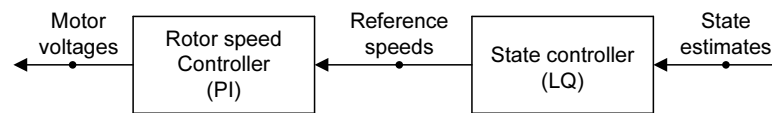


FIGURE 9.3: The hover controller structure showing a cascade coupling between four PI controllers and a LQ.

types is illustrated through the following block diagram in Figure 9.3. An advantage would be to operate the rotor speed controllers at a frequency of at least one decade higher than that of the LQ controller, as this would endure a decoupled controller design process. This would require that the closed loop eigenvalue of the rotor speed controller and thereby the natural frequency for the rotor speed controllers has to be one decade faster than the LQ-controller. However as state estimates, thereby also rotor speed estimates, arrive at 100 Hz the control frequency of the state feedback controller would be limited to a maximum of 10 Hz. As this is regarded too low, the option of decoupled design is regarded non admissible. Although the design of the rotor speed controllers will have to be incorporated in the LQ design, they will still be designed separately as the control problem at hand will prove more feasible. The following section will specify certain performance requirements for the controllers.

## 9.3 CONTROLLER REQUIREMENT SPECIFICATION

This section serves to define the specific performance requirements for the controllers. These will in turn be based on what kind of performance is wanted for the X-Pro at the end of the day. Clearly a relatively slow manoeuvring i.e. transient response both in translational and rotational sense would be preferred as personnel surrounding the vehicle in this way would feel more comfortable. This would however decrease the controllers ability to reject disturbances and as a certain wind quantity is likely to be present on the day of flight, a compromise of this issue should be found. The requirements of the four PI controllers will be defined in the following.

### 9.3.1 ROTOR SPEED CONTROLLERS

What is wanted by adding PI controllers on the motor-gear systems is not to pressure the natural frequency upwards, but to ensure that parameter differences in the motor-gear system are suppressed. The dynamic of the motors, without controllers, initially contain

the fastest poles of the modelled system thus it should be possible to control the movement of the system with the actuators without demanding a faster dynamic response of the motors. The motor and controller will comprise of a second order system and by choosing maximal damping, that is  $\zeta = 1$  this system will behave mostly as a first order system with no overshoot, and with the advantages that the included controller parameters can be used to fine tune the dynamic response of each motor system. The choice of the  $\zeta = 1$  will be the predominant constraint and secondary the natural frequency,  $\omega_n$ , should be fitted within the saturations bounds of the motor voltages, ensuring as fast a rise time  $t_r$  as possible. Specifically the saturating motor voltage is 14.8 V thus this is the boundary that should be avoided in the design.

### 9.3.2 LQ- CONTROLLER

The LQ controller takes state estimates as input and provides reference speeds for the rotor speed controllers as output. The working area of the states as defined in Section 9.1.3 on page 71, states the following.

$${}^0\vec{P}_1 : \text{No ground effect} \quad (9.7a)$$

$${}^0\vec{v}_1 : \pm 0.5 \text{ m/s} \quad (9.7b)$$

$${}^0_1\tilde{q} : \pm 30^\circ \quad (9.7c)$$

$${}^1\vec{\omega}_1 : \pm \pi \text{ rad/s} \quad (9.7d)$$

$${}^1\vec{\Omega}_2 : \text{min } 8 \text{ Hz} \approx 50 \text{ rad/s} \quad (9.7e)$$

The controller must be designed such that none of the above state boundaries are exceeded, when the system is excited in any manner. Furthermore it must converge to its reference value with respect to position, and heading without steady state error.

# ROTOR SPEED CONTROLLER



*This chapter describes the derivation of four PI controllers which are to be implemented on each motor-gear-rotor system to guarantee a certain rotor speed with a uniform input. Firstly the motor-gear-rotor Ordinary Differential Equations (ODE) will be derived and after that, control parameters designed. Lastly the controllers will be verified and concluded upon.*

The motors do not have same parameters although they come from the same manufacturer and the same production team. There are differences in the meshing (thereby total friction) of the four motor-gear systems and the balancing of the rotor blades, just to name a few dissimilarities. In the current setup it is desired that the rotors will run at the same speeds when they are given the same input, hence the features of feedback are used to suppress the differences in the before mentioned parameters.

## 10.1 DERIVING CONTROLLER EQUATIONS

The motor equations stated in 6.4 and 6.5 on page 46 are the foundation for the derivation of the rotor speed controller. They are restated in Equations (10.1) and (10.2).

$$i_a(s) = \frac{V_a(s) - K_e \cdot \Omega_m(s)}{L_a \cdot s + R_a} \quad (10.1)$$

$$\Omega_m(s) = \frac{K_m \cdot i_a(s) - \tau_f(s) - \tau_l(s)}{J_m \cdot s} \quad (10.2)$$

As can be seen the model consists of two parts, a mechanical and an electrical. The natural frequency for both parts respectively can be derived according to the following equations.

$$\frac{1}{L_a \cdot s + R_a} \Rightarrow \omega_{n,elec} = \frac{1}{L_a/R_a} \approx 745 \text{ Hz} \quad (10.3)$$

$$\frac{1}{J_m \cdot s + b_m} \Rightarrow \omega_{n,mech} = \frac{1}{J_m/b_m} \approx 0.38 \text{ Hz} \quad (10.4)$$

The equations state that the natural frequency for the electrical part is more than three decades over the natural frequency of the mechanical part. According to Nyquist-Shannon sampling theorem a signal must be sampled with at least two times the maximum occurring frequency to be able to observe. As the Robostix packets arrive at 100 Hz the dynamics of the electrical part is regarded too fast to observe. The latter together with the fact

that the motors are driven by a nonlinear PWM stage where the current behaves in an irregular manner, indicates that the dynamics of the electrical part with good reason can be left out for simplification, ( $s = 0$ ).

Inserting the new  $i_a(s)$  in Equation 10.2 and disregarding stiction which only has an effect outside the working area ( $\Omega_m = 0$ ), defines the single differential equation that describes the motor dynamics. The differential equation is reformulated as in Equation (10.5) to be seen from the rotor ( $\Omega_r = \Omega_m/\eta$ ), since the defined state vector dictates so, but also as the rotor speeds are directly observable and wanted as a control parameter.

$$\begin{aligned}\dot{\Omega}_r &= \frac{K_m \left( \frac{V_a - K_e \Omega_r \eta}{R_a} \right)}{J_{tot}} - \frac{b_{tot} \Omega_r}{J_{tot}} - \frac{\tau_c}{J_{tot} \eta^2} - \frac{\tau_l}{J_{tot} \eta^2} \\ &= \frac{K_m V_a}{R_a J_{tot} \eta} - \frac{K_m K_e \Omega_r}{R_a J_{tot}} - \frac{b_{tot} \Omega_r}{J_{tot}} - \frac{\tau_c}{J_{tot} \eta^2} - \frac{\tau_l}{J_{tot} \eta^2}\end{aligned}\quad (10.5)$$

Where  $\tau_c$  is the coulumb friction in the motor/gear part and  $\tau_l$  is the torque that the rotor blades project back on the gear wheel. In the forthcoming equations the working point and small signal value,  $\Omega_r = \bar{\Omega}_r + \tilde{\Omega}_r$ , are substituted into the equations.

As mentioned the torque produced by the rotor constitutes  $\tau_l$ . It is stated in 7.10 on page 54, and restated in Equation 10.6.

$$\tau_l \Rightarrow {}^r \tau_z = a_{\tau_z} (\bar{\Omega}_r + \tilde{\Omega}_r)^2 + b_{\tau_z} (\bar{\Omega}_r + \tilde{\Omega}_r) + c_{\tau_z} \quad (10.6)$$

Before inserting the rotor dynamics, we notice the nonlinear term in  $(\bar{\Omega}_r + \tilde{\Omega}_r)^2$  and linearize the expression.

$$a_{\tau_z} (\bar{\Omega}_r + \tilde{\Omega}_r)^2 = a_{\tau_z} \bar{\Omega}_r^2 + a_{\tau_z} \tilde{\Omega}_r^2 + 2 a_{\tau_z} \bar{\Omega}_r \tilde{\Omega}_r \approx a_{\tau_z} \bar{\Omega}_r^2 + 2 a_{\tau_z} \bar{\Omega}_r \tilde{\Omega}_r \quad (10.7)$$

Equation (10.6) and (10.7) constitutes the linearized rotor equation (10.8).

$${}^r \tau_z = a_{\tau_z} (\bar{\Omega}_r^2 + 2 \bar{\Omega}_r \tilde{\Omega}_r) + b_{\tau_z} (\bar{\Omega}_r + \tilde{\Omega}_r) + c_{\tau_z} \quad (10.8)$$

Inserting Equation (10.8) into (10.5) with  $\Omega_r = \bar{\Omega}_r + \tilde{\Omega}_r$ , reveal the following.

$$\begin{aligned}\dot{\Omega}_r &= \frac{K_m V_a}{R_a J_{tot} \eta} - \frac{K_m K_e \bar{\Omega}_r}{R_a J_{tot}} - \frac{K_m K_e \tilde{\Omega}_r}{R_a J_{tot}} - \frac{b_{tot} \bar{\Omega}_r}{J_{tot}} - \frac{b_{tot} \tilde{\Omega}_r}{J_{tot}} - \frac{\tau_c}{J_{tot} \eta^2} - \\ &\quad \frac{a_{\tau_z} \bar{\Omega}_r^2}{J_{tot} \eta^2} - \frac{2 a_{\tau_z} \bar{\Omega}_r \tilde{\Omega}_r}{J_{tot} \eta^2} - \frac{b_{\tau_z} \bar{\Omega}_r}{J_{tot} \eta^2} - \frac{b_{\tau_z} \tilde{\Omega}_r}{J_{tot} \eta^2} - \frac{c_{\tau_z}}{J_{tot} \eta^2}\end{aligned}\quad (10.9)$$

The controllable input to the motor is the voltage,  $V_a$ . On this a PI controller is added, as stated in the frequency domain Equation 10.10 with  $\Omega_{ref} = \bar{\Omega}_{ref} + \tilde{\Omega}_{ref}$ .

$$\begin{aligned} V_a &= \left( k_p + \frac{k_i}{s} \right) (\Omega_{ref} - \Omega_r) \\ &= k_p (\bar{\Omega}_{ref} + \tilde{\Omega}_{ref} - \bar{\Omega}_r - \tilde{\Omega}_r) + \frac{k_i}{s} (\bar{\Omega}_{ref} + \tilde{\Omega}_{ref} - \bar{\Omega}_r - \tilde{\Omega}_r) \end{aligned} \quad (10.10)$$

The proportional term does not include any dynamics and is inserted in Equation 10.9. The remaining term describes the voltage correction from the integrator part triggered by the error of the rotor speed, thus denoted  $V_{errI}$ . The two ODE which describes the rotor speed with integrated controller are now shown in Equation 10.11 and 10.12, given that  $\bar{\Omega}_{ref} = \bar{\Omega}_r$ .

$$\begin{aligned} \dot{\Omega}_r &= \left( -\frac{K_m}{R_a J_{tot} \eta} k_p - \frac{K_m K_e}{R_a J_{tot}} - \frac{b_{tot}}{J_{tot}} - \frac{2a_{\tau_z} \bar{\Omega}_r}{J_{tot} \eta^2} - \frac{b_{\tau_z}}{J_{tot} \eta^2} \right) \tilde{\Omega}_r + \\ &\quad \left( \frac{K_m}{R_a J_{tot} \eta} k_p \right) \tilde{\Omega}_{ref} + \left( \frac{K_m}{R_a J_{tot} \eta} \right) V_{errI} + \\ &\quad \left( -\frac{K_m K_e \bar{\Omega}_r}{R_a J_{tot}} - \frac{b_{tot} \bar{\Omega}_r}{J_{tot}} - \frac{a_{\tau_z} \bar{\Omega}_r^2}{J_{tot} \eta^2} - \frac{b_{\tau_z} \bar{\Omega}_r}{J_{tot} \eta^2} - \frac{c_{\tau_z}}{J_{tot} \eta^2} - \frac{\tau_c}{J_{tot} \eta^2} \right) \end{aligned} \quad (10.11)$$

$$\dot{V}_{errI} = k_i \tilde{\Omega}_{ref} - k_i \tilde{\Omega}_r \quad (10.12)$$

With the rotor equations defined, the controller parameters can be derived.

## 10.2 DERIVING CONTROLLER PARAMETERS

In order to derive the controller parameters, the following simplifications are made.

$$\gamma = \frac{K_m}{R_a J_{tot} \eta} \quad (10.13)$$

$$\beta = -\frac{K_m K_e}{R_a J_{tot}} - \frac{b_{tot}}{J_{tot}} - \frac{2a_{\tau_z} \bar{\Omega}_r}{J_{tot} \eta^2} - \frac{b_{\tau_z}}{J_{tot} \eta^2} \quad (10.14)$$

$$\delta = -\frac{K_m K_e \bar{\Omega}_r}{R_a J_{tot}} - \frac{b_{tot} \bar{\Omega}_r}{J_{tot}} - \frac{a_{\tau_z} \bar{\Omega}_r^2}{J_{tot} \eta^2} - \frac{b_{\tau_z} \bar{\Omega}_r}{J_{tot} \eta^2} - \frac{c_{\tau_z}}{J_{tot} \eta^2} - \frac{\tau_c}{J_{tot} \eta^2} \quad (10.15)$$

leading to the simplified equations.

$$\dot{\Omega}_r = (-\gamma k_p + \beta) \tilde{\Omega}_r + \gamma k_p \tilde{\Omega}_{ref} + \gamma V_{errI} + \delta \quad (10.16)$$

$$\dot{V}_{errI} = k_i \tilde{\Omega}_{ref} - k_i \tilde{\Omega}_r \quad (10.17)$$

The aim in this section is to find  $k_p$  and  $k_i$  for the controller such that the rotor quickly settles at the reference velocity without having overshoot according to the requirement

specification seen in Section 9.3.1. One of the problems that does not make this work straightforward is the constraints on the motor voltage. This saturation will make the system strongly nonlinear, if working near and over the saturation limit. The following derivation of  $k_p$  and  $k_i$  only holds if these saturations can be avoided.

Having the two ODE listing the full system, we form the transfer function,  $\tilde{\Omega}_r/\tilde{\Omega}_{ref}$  from Equation (10.16).

$$\begin{aligned}\frac{\tilde{\Omega}_r(s)}{\tilde{\Omega}_{ref}(s)} &= \frac{\gamma k_p s + \gamma k_i}{s^2 + (\gamma k_p - \beta) s + \gamma k_i} \\ &= \frac{\frac{k_p}{k_i} s + 1}{\frac{s^2}{\gamma k_i} + \frac{(\gamma k_p - \beta)}{\gamma k_i} s + 1}\end{aligned}\quad (10.18)$$

From the literature [FENP02, p. 151], we have the following standard form.

$$H(s) = \frac{\left(\frac{s}{\alpha\zeta\omega_n}\right) + 1}{\left(\frac{s}{\omega_n}\right)^2 + 2\zeta\left(\frac{s}{\omega_n}\right) + 1}\quad (10.19)$$

Where  $\omega_n$  is the undamped natural frequency,  $\zeta$  is the damping ratio and  $\alpha$  is related to the locations of the zero with respect to the poles. With values of  $\alpha = 1$  the zero will have substantial effect on the rest of the system, with  $\alpha = 100$  the effect of the zero can be neglected. By comparing terms in 10.18 with 10.19, it is possible to link the controller parameters with  $\zeta$  and  $\omega_n$  through the following.

$$k_i = \frac{\omega_n^2}{\gamma}\quad (10.20)$$

$$k_p = \sqrt{\frac{4\zeta^2\gamma k_i + \beta^2}{\gamma^2}} = \sqrt{\frac{4\zeta^2\omega_n^2 + \beta^2}{\gamma^2}}\quad (10.21)$$

Stated in Section 9.3.1 comes the constraint that entails  $\zeta = 1$ . Having obtained this it is wanted to have as fast a  $\omega_n$  as possible. The limitation on this parameter is the saturation on the motor voltage. This nonlinear behavior is most easily modeled in Simulink<sup>®</sup>. Figure 10.1 shows the setup of the saturation. The limits are set to 0 V and 14.8 V which are realistic limits for the motor voltage.

From this system it is possible to plot the step response and observe which natural frequency is possible without noticeable saturation. In Figure 10.2 the step response at  $\omega_n = 3.5 \cdot 2\pi \text{ rad/s}$  and  $\zeta = 1$  is plotted.



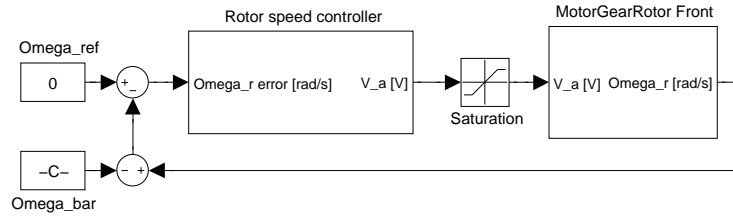


FIGURE 10.1: The setup for testing step response vs. controller parameters. In this example for the front

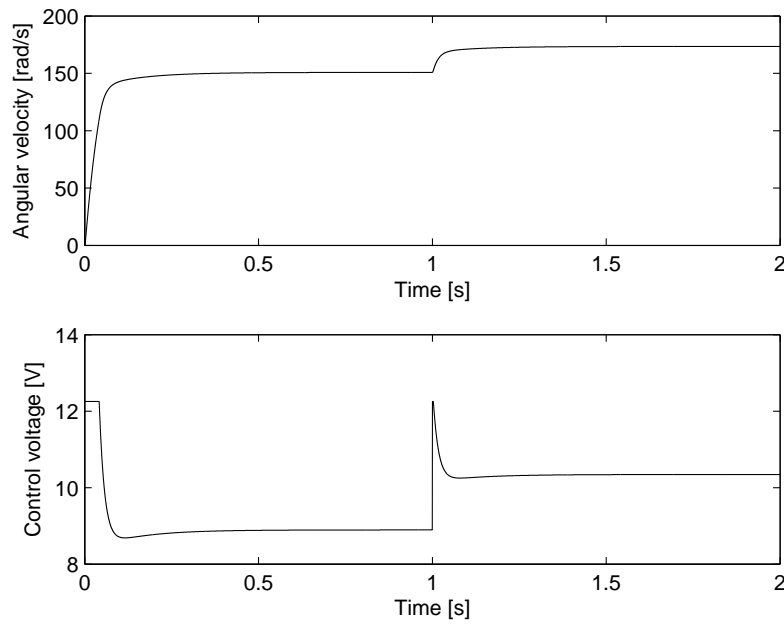


FIGURE 10.2: Simulated step responses from zero to full hover speed, and a step from hover to (hover + 15%). The motor voltage saturates accordingly for 41.7 ms and 2.7 ms.

Two steps are performed as going from zero to the hover speed, and from there to 15% over the hover speed. The first step is not realistic in the sense that the rotor blades will collapse when it is applied, however it gives an idea of the amount of saturation that occurs and the overall performance. The second step is more realistic however still powerful, which only saturates the motor voltage for 2.7 ms, thereby no noticeable effect on the overshoot on the angular velocity.

## 10.3 CONCLUSION

This chapter has derived speed controllers for the motor-gear-rotor system, in order to guarantee a certain rotor speed at all times, thereby suppressing parameter inaccuracies in the friction and internal motor parameters. Purely by adding the PI controllers on each of the four systems the steady state angular velocity is ensured. The proportional and the integral gain has been designed to make the system fast without pressuring the poles out on the imaginary axis. Furthermore according to Figure 10.2 the response shows no overshoot and together with the before mentioned issues the controller fulfills the requirements specified in section 9.3.1 on page 73. Though having a rough idea on the controller parameters, leaves still the assignment to fit all four systems to have the same dynamic response. This will be done when implemented together with the LQ controller, which is handled in the next chapter, and estimator on the soft real time target environment.



# LINEAR QUADRATIC CONTROLLER

*This chapter concerns the LQ controller design. The purpose of the controller is, together with the speed controllers, to stabilize the X-Pro in a hover condition. The chapter will firstly specify the system equations based on the model and the speed controllers from the previous chapter. These equations are then linearized, which serves as basis for the LQ controller. The controller will then be designed and verified through simulation when applied to the non-linear model.*

## 11.1 SYSTEM EQUATIONS

This section deals with identifying the equations which needs to be linearized in order to get the system represented on state space form. The non-linear system can be represented as shown in Equation (11.1).

$$\dot{x} = f(x, u) \quad (11.1a)$$

$$y = g(x, u) \quad (11.1b)$$

where  $x$  is the state,  $u$  is the input and  $y$  is the output vector.

The equations describing the non-linear model and the rotor speed controller from the previous chapter, are stated in Equation (11.2), thereby also the derivative of the state vector. The equations are the result of the modelling and the derived rotor speed controller.

$${}^0\dot{P}_1 = {}^0\vec{v}_1 \quad (11.2a)$$

$${}^0\dot{v}_1 = {}^0_1q \begin{bmatrix} \sum_{n=1}^4 \frac{{}^1\vec{f}_{lift,n}}{m_{tot}} \\ 0 \end{bmatrix} {}^0_1q^* + {}^0\vec{g} \quad (11.2b)$$

$${}^0_1\dot{q} = \frac{1}{2} {}^0_1q \begin{bmatrix} {}^1\vec{\omega}_1 \\ 0 \end{bmatrix} \quad (11.2c)$$

$${}^1\dot{\vec{\omega}}_1 = {}^1I_{tot}^{-1} \left\{ \begin{array}{l} \sum_{j=1}^4 [{}^1\vec{n}_{rotor,j} + {}^1\vec{P}_{2,j} \times {}^1\vec{f}_{rotor,j}] - {}^1\vec{\omega}_1 \times ({}^1I_{tot} {}^1\vec{\omega}_1) \dots \\ - {}^1(C_2 \tilde{I}_2) {}^1\dot{\vec{\Omega}}_{2,tot} - {}^1\vec{\omega}_1 \times ({}^1(C_2 \tilde{I}_2) {}^1\vec{\Omega}_{2,tot}) \end{array} \right\} \quad (11.2d)$$

$$\begin{aligned} \dot{\tilde{\Omega}}_{r,l} = & \left( -\frac{K_m}{R_a J_{tot} \eta} k_{p,l} - \frac{K_m K_e}{R_a J_{tot}} - \frac{b_{tot,l}}{J_{tot}} - \frac{2a_{\tau_z} \bar{\Omega}_r}{J_{tot} \eta^2} - \frac{b_{\tau_z}}{J_{tot} \eta^2} \right) \tilde{\Omega}_{r,l} + \\ & \left( \frac{K_m}{R_a J_{tot} \eta} k_{p,l} \right) \tilde{\Omega}_{ref,l} + \left( \frac{K_m}{R_a J_{tot} \eta} \right) V_{errI,l} + \\ & \left( -\frac{K_m K_e \bar{\Omega}_r}{R_a J_{tot}} - \frac{b_{tot,l} \bar{\Omega}_r}{J_{tot}} - \frac{a_{\tau_z} \bar{\Omega}_r^2}{J_{tot} \eta^2} - \frac{b_{\tau_z} \bar{\Omega}_r}{J_{tot} \eta^2} - \frac{c_{\tau_z}}{J_{tot} \eta^2} - \frac{\tau_{c,l}}{J_{tot} \eta^2} \right) \end{aligned} \quad (11.2e)$$

$$\dot{V}_{errI,l} = k_{i,l} \tilde{\Omega}_{ref,l} - k_i \tilde{\Omega}_{r,l} \quad (11.2f)$$

for  $l = \{1, 2, 3, 4\}$  representing the four motor-gear-rotor systems.

When the left hand side of the above equations are compared to the derivative of the state vector stated in Equation (9.1) on page 70, the only difference is the  $\dot{V}_{errI,(1:4)}$  from each rotor speed controller. This increases the order of the system by four. Thereby the new state vector is:

$$x_{20 \times 1} = \left[ {}^0\vec{P}_1^T \quad {}^0\vec{v}_1^T \quad {}^0q_{1:3}^T \quad {}^1\vec{\omega}_1^T \quad \vec{\Omega}_r^T \quad \vec{V}_{errI}^T \right]^T \quad (11.3)$$

With the state vector in place the input and output vector needs to be chosen, in order to get the linearized system on state space form. The input vector is chosen due to interface between the LQ and rotor speed controller to the rotor speed reference, as depicted on Figure 9.3 on page 73.

$$u_{4 \times 1} = [\Omega_{ref,1} \quad \Omega_{ref,2} \quad \Omega_{ref,3} \quad \Omega_{ref,4}]^T \quad (11.4)$$

$$y = I_{16 \times 20} x_{20 \times 1} \quad (11.5)$$

With Equation (11.2) and the input, state and output vector defined the linearization of can be carried out, and thereby the basis for the controller have been created.

## 11.2 LINEARIZATION

The quaternions are complex to linearized therefore this is carried out by hand, in co-operation with [GHB07]. The equations containing quaternions are  $\vec{v}_1$  and  ${}^0q_{1:3}$ , Equation (11.2b) and (11.2c) respectively. The Jacobian method, truncated to first order and evaluated in a working point, are applied to the remaining equations of Equation (11.2), resulting in the complete linearized system.

### 11.2.1 LINEARIZATION OF TRANSLATORY VELOCITY

The linearization of the translatory velocity is in principle a task of linearization of the rotation of all the lift forces, so they depend linearly of  $\tilde{\Omega}_r$  and  ${}^0_1\tilde{q}$ . The starting point is stated in (11.6).

$$\dot{\vec{v}}_1 = {}^0_1q \begin{bmatrix} \sum_{n=1}^4 \frac{{}^1\vec{f}_{lift,n}(\Omega_r)}{m_{tot}} \\ 0 \end{bmatrix} {}^0_1q^* + {}^0\vec{g} \quad (11.6)$$

The single quaternion rotation is replaced by two new rotations, a fixed and a small deviation from this. From now on it should be clear which quaternion is used, thus  $q = {}^0_1q$  to make the equations more readable. We expand 11.6 by (11.7), to arrive at (11.8)

$$q = \bar{q} \tilde{q} \quad (11.7)$$

$$\dot{\vec{v}}_1 = \bar{q} \tilde{q} \begin{bmatrix} \sum_{n=1}^4 \frac{{}^1\vec{f}_{lift,n}(\tilde{\Omega}_r)}{m_{tot}} \\ 0 \end{bmatrix} \tilde{q}^* \bar{q}^* + {}^0\vec{g} \quad (11.8)$$

The lift force consist of four nonlinear functions, in  $\tilde{\Omega}_r$ , the linearization yields the following for the front rotor, but will be equivalent for the other three.

$$\begin{aligned} f_{lift,F} &= a {}^1\Omega_{2,F}^2 + b {}^1\Omega_{2,F} \\ &= a \left( \bar{\Omega}_r + {}^1\tilde{\Omega}_{2,F} \right)^2 + b \left( \bar{\Omega}_r + {}^1\tilde{\Omega}_{2,F} \right) \\ &= a \left( \bar{\Omega}_r^2 + {}^1\tilde{\Omega}_{2,F}^2 + 2\bar{\Omega}_r {}^1\tilde{\Omega}_{2,F} \right) + b\bar{\Omega}_r + b {}^1\tilde{\Omega}_{2,F} \\ &= (a\bar{\Omega}_r^2 + b\bar{\Omega}_r) + (2a\bar{\Omega}_r + b) {}^1\tilde{\Omega}_{2,F} \end{aligned} \quad (11.9)$$

Substituting Equation (11.9) back into Equation (11.8) yields Equation 11.10.

$$\begin{aligned}
 \dot{v}_1 &= \bar{q} \tilde{q} \begin{bmatrix} 0 \\ 0 \\ \sum_{n=1}^4 \frac{(2a\bar{\Omega} + b) {}^1\tilde{\Omega}_{2,n}}{m_{tot}} \\ 0 \end{bmatrix} \tilde{q}^* \bar{q}^* + \bar{q} \tilde{q} \begin{bmatrix} 0 \\ 0 \\ \sum_{n=1}^4 \frac{a\bar{\Omega}^2 + b\bar{\Omega}}{m_{tot}} \\ 0 \end{bmatrix} \tilde{q}^* \bar{q}^* + {}^0\vec{g} \\
 &= \frac{2a\bar{\Omega} + b}{m_{tot}} \underbrace{\bar{q} \tilde{q} \begin{bmatrix} 0 \\ 0 \\ \sum_{n=1}^4 {}^1\tilde{\Omega}_{2,n} \\ 0 \end{bmatrix} \tilde{q}^* \bar{q}^*}_{\star} + \frac{4a\bar{\Omega}^2 + b\bar{\Omega}}{m_{tot}} \underbrace{\bar{q} \tilde{q} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tilde{q}^* \bar{q}^*}_{\Delta} + {}^0\vec{g} \quad (11.10)
 \end{aligned}$$

The problem is now boiled down to linearizing the two terms  $\Delta$  and  $\star$ . Starting by examining  $\Delta$  will yield the following.

As hover is the working point, thus  $\bar{q}$  is well defined and stated in Equation (11.11)

$$\bar{q} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T \quad (11.11)$$

This quaternion will rotate a vector 180 degrees around the y-axis given in frame {1}. By knowing this it is easy to replace the rotation with the corresponding rotation matrix.

$$\Delta = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{pmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ \tilde{q} \\ \tilde{q}^* \end{pmatrix} \quad (11.12)$$

Equation (11.12) only makes sense if the result of the quaternion rotation is later on treated as a vector, and this can be done without loss of information since the fourth element will remain zero. What remains is a small rotation by  $\tilde{q}$ , in order to linearize this rotation one needs to write the quaternion multiplication outline. The following template for the multiplication is used.

$$pq = \begin{bmatrix} p_{1:3} \times q_{1:3} + p_{1:3} q_4 + q_{1:3} p_4 \\ -p_{1:3}^T q_{1:3} + p_4 q_4 \end{bmatrix} \quad (11.13)$$

Using this definition on the inner term of Equation (11.12), yields Equation (11.14).

$$\begin{aligned}
 \tilde{q} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tilde{q}^* &= \tilde{q} \left[ \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \times \tilde{q}_{1:3}^* + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tilde{q}_4^* \right. \\
 &\quad \left. - \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tilde{q}_{1:3}^* \right] \\
 &= \tilde{q} \begin{bmatrix} \tilde{q}_2 \\ -\tilde{q}_1 \\ \tilde{q}_4 \\ \tilde{q}_3 \end{bmatrix} = \begin{bmatrix} \tilde{q}_2 \\ -\tilde{q}_1 \\ \tilde{q}_4 \\ -\tilde{q}_{1:3}^T \begin{bmatrix} \tilde{q}_2 & -\tilde{q}_1 & \tilde{q}_4 \end{bmatrix}^T + \tilde{q}_4 \tilde{q}_3 \end{bmatrix} \quad (11.14)
 \end{aligned}$$

Using small signal approximation on Equation (11.14), and knowing that:

$$\lim_{\theta \rightarrow 0} \begin{bmatrix} q_{1:3} \\ q_4 \end{bmatrix} = \lim_{\theta \rightarrow 0} \begin{bmatrix} \hat{e} \sin(\frac{\theta}{2}) \\ \cos(\frac{\theta}{2}) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (11.15)$$

We can linearize Equation (11.14) as shown in Equation (11.16).

$$\tilde{q} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tilde{q}^* \approx \begin{bmatrix} \begin{bmatrix} \tilde{q}_2 \\ -\tilde{q}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} \tilde{q}_2 \\ -\tilde{q}_1 \\ 1 \end{bmatrix} \\ 0 \end{bmatrix} = \begin{bmatrix} 2\tilde{q}_2 \\ -2\tilde{q}_1 \\ 1 \\ 0 \end{bmatrix} \quad (11.16)$$

Turning to the second part of Equation (11.10) denoted by  $\star$ , we only need to realize the following case, again shown for the front motor:

$$\tilde{q} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tilde{q}^* = \tilde{q} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tilde{q}^* \stackrel{1}{\tilde{\Omega}}_{2,F} = \begin{bmatrix} 2\tilde{q}_2 \\ -2\tilde{q}_1 \\ 1 \\ 0 \end{bmatrix} \stackrel{1}{\tilde{\Omega}}_{2,F} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \stackrel{1}{\tilde{\Omega}}_{2,F} \quad (11.17)$$

Combining both linearized terms yield Equation (11.18)

$$\dot{v}_1 = \underbrace{\frac{2a\bar{\Omega} + b}{m_{tot}} \begin{bmatrix} 0 \\ 0 \\ -\sum_{n=1}^4 \stackrel{1}{\tilde{\Omega}}_{2,n} \end{bmatrix}}_{\Delta} + \underbrace{\frac{4(a\bar{\Omega}^2 + b\bar{\Omega})}{m_{tot}} \begin{bmatrix} -2\tilde{q}_2 \\ -2\tilde{q}_1 \\ -1 \end{bmatrix}}_{\star} + {}^0\vec{g} \quad (11.18)$$

Looking at Equation (11.18) the structure seems intuitive, the  $\Delta$  term states that the acceleration of the body in the z-axis, if the four rotors change from the hover state. The negative sign in front of the summation will in practice flip because both  $a$  and  $b$  are negative. Thereby an increase in rotor speed will produce a positive acceleration in the z direction. The  $\star$  term includes the functionality that if the X-Pro is not level, the body will accelerate in the xy-plane. Finally the constant z-term of  $\star$  include the term that in hover will balance out the gravity. This can be converted to a constraint is given in Equation (11.20)

$$\frac{4(a\bar{\Omega}^2 + b\bar{\Omega})}{m_{tot}} \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = 0 \quad (11.19)$$

$$\frac{4(a\bar{\Omega}^2 + b\bar{\Omega})}{m_{tot}} = -g \quad (11.20)$$

Choosing  $\bar{\Omega}$  such that the rotor lift cancel out gravity, will simplify the expression further and yield the final linearized equation in Equation 11.21

$$\dot{\vec{v}}_1 = \frac{2a\bar{\Omega} + b}{m_{tot}} \begin{bmatrix} 0 \\ 0 \\ -\sum_{n=1}^4 {}^1\tilde{\Omega}_{2,n} \end{bmatrix} + \frac{4(a\bar{\Omega}^2 + b\bar{\Omega})}{m_{tot}} \begin{bmatrix} -2\tilde{q}_2 \\ -2\tilde{q}_1 \\ 0 \end{bmatrix} \quad (11.21)$$

### 11.2.2 LINEARIZATION OF THE ATTITUDE

The time derivative of the quaternion describing the attitude of the X-Pro is a nonlinear equation. The following section will deal with a linearization of Equation (8.11) on page 63 and is restated in (11.22). For simplicity in the following equations,  $q_\omega$  is defined to be the quaternion interpretation of the angular body velocity. Defining again  $q = {}^0_1 q$ .

$${}^0_1 \dot{q} = \frac{1}{2} {}^0_1 q \begin{bmatrix} {}^1\vec{\omega}_1 \\ 0 \end{bmatrix} = \frac{1}{2} {}^0_1 q(t) q_\omega(t) \quad (11.22)$$

For the problem at hand inspiration is found in [Bha05, pp. 58]. Starting of we define a working attitude and a small deviation from this working attitude. The rules of small signal approximation apply, but working with quaternions one needs to multiply as defined in Equation (11.23) in order to derive the small signal attitude,  $\tilde{q}$ .



$$\begin{aligned}
 q(t) &= \bar{q}(t)\tilde{q}(t) \Leftrightarrow \\
 \tilde{q}(t) &= \bar{q}^*(t)q(t)
 \end{aligned} \tag{11.23}$$

Using the chain rule and Equation (11.22), we can arrive at the derivative of the small signal attitude quaternion, as in Equation (11.24)

$$\begin{aligned}
 \dot{\tilde{q}}(t) &= \dot{\bar{q}}^*(t)q(t) + \bar{q}^*(t)\dot{q}(t) \\
 &= \frac{1}{2} \left( (\bar{q}(t)q_{\bar{\omega}}(t))^* \dot{q}(t) + \bar{q}^*(t)\dot{q}(t)q_{\omega}(t) \right) \\
 &= \frac{1}{2} \left( -q_{\bar{\omega}}(t)\bar{q}^*(t)\dot{q}(t) + \bar{q}^*(t)\dot{q}(t)q_{\omega}(t) \right) \\
 &= \frac{1}{2} \left( -q_{\bar{\omega}}(t)\tilde{q}(t) + \tilde{q}(t)q_{\omega}(t) \right)
 \end{aligned} \tag{11.24}$$

Note that since the scalar part of  $q_{\bar{\omega}}$  is zero, the complex conjugate to this is simply  $-q_{\bar{\omega}}$ . The angular velocity can as well be defined as a working point,  $\bar{\omega}$ , and a small signal,  $\tilde{\omega}$ , as

$$\vec{\omega}(t) = \bar{\omega}(t) + \tilde{\omega}(t) \tag{11.25}$$

Hence  $q_{\omega}$  can be written

$$q_{\bar{\omega}}(t) = \begin{bmatrix} \bar{\omega}(t) + \tilde{\omega}(t) \\ 0 \end{bmatrix} \tag{11.26}$$

$$= q_{\bar{\omega}}(t) + q_{\tilde{\omega}}(t) \tag{11.27}$$

Since the scalar part of both  $q_{\bar{\omega}}$  and  $q_{\tilde{\omega}}$  are zero the elements are associative and Equation (11.24) can be expanded to (11.28)

$$\dot{\tilde{q}}(t) = \frac{1}{2} \left( -q_{\bar{\omega}}(t)\tilde{q}(t) + \tilde{q}(t)q_{\bar{\omega}}(t) + \tilde{q}(t)q_{\tilde{\omega}}(t) \right) \tag{11.28}$$

To further reduce the equation the quaternion is split up into a vector part and scalar part. The last term of Equation (11.28) is still troublesome, thus the following approximation is performed.

$$\tilde{q}(t)q_{\tilde{\omega}}(t) = \begin{bmatrix} \tilde{q}_{1:3}(t) \times \tilde{\omega}(t) + \tilde{q}_4(t)\tilde{\omega}(t) \\ -\tilde{q}_{1:3}^T(t)\tilde{\omega}(t) \end{bmatrix} \tag{11.29}$$

$$\approx q_{\tilde{\omega}}(t) \tag{11.30}$$

The two first terms of Equation (11.28) can likewise be expanded as shown in, Equation (11.31) and Equation (11.32).

$$q_{\bar{\omega}}(t)\tilde{q}(t) = \begin{bmatrix} \bar{\omega}(t) \times \tilde{q}_{1:3}(t) + \bar{\omega}(t)\tilde{q}_4(t) \\ -\bar{\omega}^T(t)\tilde{q}_{1:3}(t) \end{bmatrix} \quad (11.31)$$

$$\tilde{q}(t)q_{\bar{\omega}}(t) = \begin{bmatrix} \tilde{q}_{1:3}(t) \times \bar{\omega}(t) + \tilde{q}_4(t)\bar{\omega}(t) \\ -\tilde{q}_{1:3}^T(t)\bar{\omega}(t) \end{bmatrix} \quad (11.32)$$

Substituting back Equation (11.31), (11.32) and (11.30) into Equation (11.28) yields (11.33)

$$\dot{\tilde{q}}(t) = \begin{bmatrix} -\mathbf{S}(\bar{\omega}(t)) & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 0 \end{bmatrix} \tilde{q}(t) + \frac{1}{2}q_{\bar{\omega}}(t) \quad (11.33)$$

Where  $\mathbf{S}(\bar{\omega}(t))$  is the cross-product matrix function, applied on  $\bar{\omega}(t)$ . And  $\mathbf{0}_{1 \times 3}$  is a zero matrix of the denoted size. Throughout this linearization it has been expected that the working point,  $\bar{\omega}$  could change over time. This would be the case if the control objective was to follow a trajectory, but since the present scope only includes the hover state,  $\bar{\omega}$  will not change over time thus be equal to zero in hover. This eventually means that  $\mathbf{S}(\bar{\omega}(t))\tilde{\omega}$  will be zero and the linearized derivative of the small signal quaternion becomes, for the hover objective.

$$\dot{\tilde{q}}(t) = \frac{1}{2}q_{\bar{\omega}}(t) \quad (11.34)$$

With the quaternions are now linearized and the linearization process of the remaining equations in (11.2) are automated and carried out in MATLAB<sup>®</sup>, see `/matlab/control/LQR/linearization.m`.

## 11.3 STATE SPACE FORM

When putting the linearized equation on state space form, the working point of the system needs to be calculated. This gives that the following criteria has to be fulfilled:

$$\dot{x} = f(x_0, u_0) = \vec{0} \quad (11.35)$$

The  $x_0$  and  $u_0$  that fulfills this equation will be the operating point. The sensor readings for the general case will then be:

$$y_0 = g(x_0, u_0) \quad (11.36)$$

The small-signal values for all states, except the quaternion see Equation (11.38), are then defined as:

$$\tilde{x} = x - x_0, \quad \tilde{u} = u - u_0, \quad \tilde{y} = y - y_0 \quad (11.37)$$

When finding the small-signal value of the quaternion the following must be fulfilled:

$$\tilde{q} = \tilde{q}^* q \quad (11.38)$$

The linearized state space equations for small-signals become:

$$\dot{\tilde{x}} = A \tilde{x} + B \tilde{u} \quad (11.39a)$$

$$\tilde{y} = C \tilde{x} + D \tilde{u} \quad (11.39b)$$

Here  $A$ ,  $B$ ,  $C$  and  $D$  are the coefficient matrices, defined as the Taylor series expansion truncated to first order, also called the Jacobian. When evaluating in the operating point these can be derived as shown in Equation (11.40).

$$\begin{aligned} A &= \left. \frac{\partial f(x,u)}{\partial x} \right|_{x_0, u_0}, & B &= \left. \frac{\partial f(x,u)}{\partial u} \right|_{x_0, u_0} \\ C &= \left. \frac{\partial g(x,u)}{\partial x} \right|_{x_0, u_0}, & D &= \left. \frac{\partial g(x,u)}{\partial u} \right|_{x_0, u_0} \end{aligned} \quad (11.40)$$

The linearized equations are now represented on state space form, and the matrices have the dimensions  $A_{20 \times 20}$ ,  $B_{20 \times 4}$ ,  $C_{16 \times 20}$  and  $D_{20 \times 4}$ . Since the input  $u$  does not directly affect the output  $y$ , the  $D$  matrix will be zero.

### 11.3.1 OPERATING POINT

Hover is achieved with the following operating point:

$${}^0\bar{P}_1 = \vec{0} \quad (11.41a)$$

$$\vec{v}_{1_{op}} = \vec{0} \quad (11.41b)$$

$${}^0_1\bar{q} = [0 \ 1 \ 0 \ 0]^T \quad (11.41c)$$

$$\vec{\omega} = \vec{0} \quad (11.41d)$$

$$\bar{\Omega}_r = [150.78 \ 150.78 \ 150.78 \ 150.78]^T \quad (11.41e)$$

$$\vec{V}_{errI} = [8.81 \ 8.76 \ 9.29 \ 8.92]^T \quad (11.41f)$$

The working point of the rotor velocity is calculated by solving Equation (11.20) with respect to  $\Omega_r$ .

Before discretization of the linearized system, this needs to be tested for controllability, which will be handled in the following section.

### 11.3.2 CONTROLLABILITY

To ensure that the system can be controlled, the controllability is tested. The MATLAB<sup>®</sup> function `ctrb` generates a controllability matrix:

$$\begin{bmatrix} B & AB & \dots & A^{\dim(x)-1}B \end{bmatrix} \quad (11.42)$$

The rank of the controllability matrix should be the same as the length of the state vector  $x$  to achieve full controllability. Due to the high order of the system, using the MATLAB<sup>®</sup> function `rank` on the result of `ctrb` does not work properly.

The MATLAB<sup>®</sup> documentation [Mat07] suggests to use `ctrbf` instead, which returns the system in a stair case form, sorting the controllable and non-controllable states. The function returns a vector  $K$ , which contains the number of controllable states found at each iteration. The sum of elements of  $K$  gives the total number of controllable states. The result of `ctrbf` yielded full rank, which gives full controllability. This result has been double checked with the Popov-Belevich-Hautus (PBH) test, [Tsk07].

### 11.3.3 DISCRETIZATION

Before the controller can be implemented the linearized system must be discretized at a certain sample time  $T_s$ . As mentioned earlier in section 9.2 on page 72 the maximum admissible control frequency is bounded by the rate at which the sensors are sampled, i.e. 100 Hz. As much as a high control frequency is desired, it will eventually be bounded by what the MATLAB<sup>®</sup> target environment allows to pass.

The control frequency is according to the latter chosen to be 25 Hz and the linearized system must therefore also be discretized at 25 Hz, i.e.  $T_s = 0.04$  s. The `c2d` function in MATLAB<sup>®</sup> is used and the Zero Order Hold method applied. The new discretized structure is shown in Equation 11.43.

$$\tilde{x}(k+1) = \Phi_s \tilde{x} + \Gamma u \quad (11.43a)$$

$$\tilde{y}(k) = H_s \tilde{x}_s \quad (11.43b)$$

and the discrete LQ controller is now ready to be derived.

## 11.4 DERIVING CONTROLLER PARAMETERS

Linear Quadratic controllers are applied to MIMO systems in order to gain optimal control. Practically this means finding good candidate values of the feedback gain,  $\mathbf{K}$ , such that it minimizes the performance index, stated in Equation (11.44), [FPW98, p. 364]:

$$\mathcal{J} = \frac{1}{2} \sum_{k=0}^N (\tilde{x}_s^T(k) \mathbf{Q} \tilde{x}_s(k) + \tilde{u}_s^T(k) \mathbf{R} \tilde{u}_s(k)) \quad (11.44)$$

The performance index allows use of two parameters,  $\mathbf{R}$ , an  $m \times m$  control weighting matrix and  $\mathbf{Q}$ , a  $n \times n$  state weighting matrix. The weights  $\mathbf{Q}$  and  $\mathbf{R}$  are used to balance the state and the input respectively. The derived controller must maintain the LQ requirements specified in section 9.3.2 on page 74 and due to the unwanted steady state error in the position and heading, integral action must be introduced.

### 11.4.1 INTEGRAL ACTION

The integral action increases the order of the system, by introducing four integral states on  $P_x, P_y, P_z$  and  $q_3$ . The closed loop system with integral states is shown in Figure 11.1.

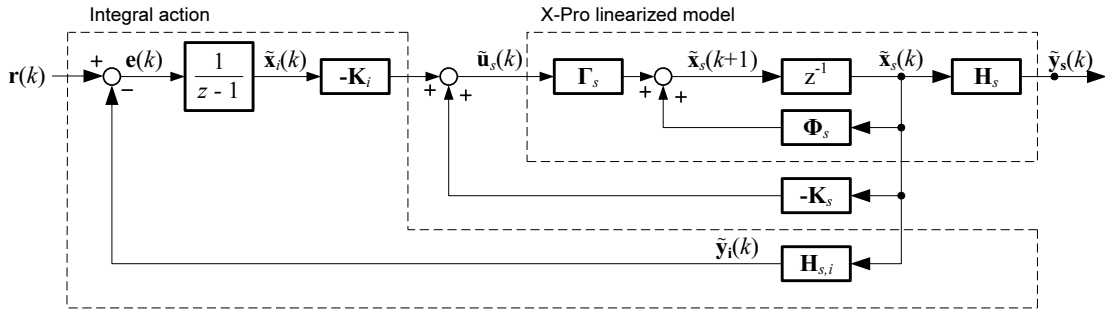


FIGURE 11.1: Closed loop system of the X-Pro with integral states.

The integral states can be expressed by the following, [And07, p. 42]:

$$\tilde{x}_i(k+1) = \tilde{x}_i(k) + e(k) \quad (11.45)$$

where  $e(k) = -\tilde{y}_i(k)$ , because the reference input is zero. Thereby the resulting equation

is:

$$\begin{aligned}\tilde{x}_i(k+1) &= \tilde{x}_i(k) - \tilde{y}(k) \\ &= \tilde{x}_i(k) - \mathbf{H}_{s,i}\tilde{x}_s(k)\end{aligned}\quad (11.46)$$

where  $\mathbf{H}_{s,i}$  represents the rows of  $\mathbf{H}_s$  corresponding to the integral states, due to the fact that not all of the states in  $\tilde{x}_s$  needs integral action. In this case the demands for the controller, states that the X-Pro must hover in a fixed position, therefore the steady state error of the position must be reduced. Further it is not desired to have a steady state error on the third component of the attitude,  $\tilde{q}_3$ . If no integral action is applied to  $\tilde{q}_3$  the heading of the X-Pro could have a steady state error. This gives a problem if the value of  $\tilde{q}_3$  gets too large, the X-Pro would simply crash. The introduced integral states gives that  $\tilde{x}_i = \begin{bmatrix} \tilde{P}_i^T & \tilde{q}_{3,i} \end{bmatrix}^T$ . and the system order is increased by four.

When Equation (11.46) is inserted into (11.43) the system equations are defined by the following:

$$\begin{bmatrix} \tilde{x}_s(k+1) \\ \tilde{x}_i(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_s & \mathbf{0} \\ -\mathbf{H}_{s,i} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \tilde{x}_s(k) \\ \tilde{x}_i(k) \end{bmatrix} + \begin{bmatrix} \mathbf{\Gamma}_s \\ \mathbf{0} \end{bmatrix} \tilde{u}_s(k)\quad (11.47a)$$

$$\tilde{y}(k) = \begin{bmatrix} \mathbf{H}_s & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{x}_s(k) \\ \tilde{x}_i(k) \end{bmatrix}\quad (11.47b)$$

Looking closer at Equation (11.47) the following can be derived:

$$x_i(k) = \begin{bmatrix} -\mathbf{0} & \mathbf{I} \end{bmatrix} \tilde{x}(k) = \mathbf{H}_i \tilde{x}(k)\quad (11.48)$$

$$e(k) = \begin{bmatrix} -\mathbf{H}_{s,i} & \mathbf{0} \end{bmatrix} \tilde{x}(k) = \mathbf{H}_e \tilde{x}(k)\quad (11.49)$$

$\mathbf{H}_i$  and  $\mathbf{H}_e$  are needed when deriving the new performance index, which gives the possibility of weighting the integral states as well.

Due to the difficulty of creating a weight on the state it is easier to weight the state error,  $e(k)$ , Therefore the  $\mathbf{Q}$  matrix needs to be expanded in order to contain the information, this is also the case with the integral state,  $\tilde{x}_i(k)$ . With this the new  $\mathbf{Q}$  matrix is, [And07, p. 47]:

$$\mathbf{Q} = \mathbf{H}_e^T \mathbf{Q}_e \mathbf{H}_e + \mathbf{H}_i^T \mathbf{Q}_i \mathbf{H}_i\quad (11.50)$$

The  $\tilde{x}^T(k)\mathbf{Q}\tilde{x}$  can then be rewritten as:

$$\tilde{x}^T(k)\mathbf{Q}\tilde{x}(k) = \tilde{x}^T(k) (\mathbf{H}_e^T \mathbf{Q}_e \mathbf{H}_e + \mathbf{H}_i^T \mathbf{Q}_i \mathbf{H}_i) \tilde{x}(k)\quad (11.51)$$

$$= \tilde{x}^T(k) (\mathbf{H}_e^T \mathbf{Q}_e \mathbf{H}_e) \tilde{x}(k) + \tilde{x}^T(k) (\mathbf{H}_i^T \mathbf{Q}_i \mathbf{H}_i) \tilde{x}(k)\quad (11.52)$$

$$= (\mathbf{H}_e \tilde{x}(k))^T \mathbf{Q}_e (\mathbf{H}_e \tilde{x}(k)) + (\mathbf{H}_i \tilde{x}(k))^T \mathbf{Q}_i (\mathbf{H}_i \tilde{x}(k))\quad (11.53)$$

$$= e^T(k)\mathbf{Q}_e e(k) + \tilde{x}_i(k)^T \mathbf{Q}_i \tilde{x}_i(k)\quad (11.54)$$

With the new definition of  $\tilde{x}^T(k)\mathbf{Q}\tilde{x}(k)$  the performance index becomes:

$$\mathcal{J} = \frac{1}{2} \sum_{k=0}^N (e^T(k)\mathbf{Q}_e e(k) + \tilde{x}_i^T(k)\mathbf{Q}_i \tilde{x}_i(k) + \tilde{u}_s^T(k)\mathbf{R}\tilde{u}_s(k)) \quad (11.55)$$

The control law of Equation (11.47a) becomes:

$$\tilde{u}_s(k) = - \begin{bmatrix} \mathbf{K}_s & \mathbf{K}_i \end{bmatrix} \begin{bmatrix} \tilde{x}_s(k) \\ \tilde{x}_i(k) \end{bmatrix} \quad (11.56)$$

With the control law and the augmented system defined the weight matrices can now to be defined.

## 11.4.2 WEIGHT MATRICES

To find  $\mathbf{Q}$  and  $\mathbf{R}$  Brysons rule is used, [FENP02, p. 537].

$$\mathbf{Q}_{jj} = \frac{1}{\max(\tilde{x}_j^2(k))} \quad (11.57)$$

$$\mathbf{R}_{jj} = \frac{1}{\max(\tilde{u}_j^2(k))} \quad (11.58)$$

The Weight matrices are formed on the basis of the controller requirement specifications in Section 9.3.2. The specific weights are listed in Table 11.1.

$\mathbf{Q}_e$				$\mathbf{Q}_i$		$\mathbf{R}$	
State	Max error	State	Max error	State	Max	State	Max
$P_x$	0.2	$\omega_y$	$\pi$	$P_{x,i}$	1	$\Omega_{ref,F}$	1
$P_y$	0.2	$\omega_z$	1	$P_{y,i}$	1	$\Omega_{ref,L}$	1
$P_z$	0.2	$\Omega_{r,F}$	500	$P_{z,i}$	1	$\Omega_{ref,B}$	1
$v_x$	0.01	$\Omega_{r,L}$	500	$q_{3,i}$	1	$\Omega_{ref,R}$	1
$v_y$	0.01	$\Omega_{r,B}$	500				
$v_z$	0.01	$\Omega_{r,R}$	500				
$q_1$	0.012	$V_{errI,F}$	100				
$q_2$	0.012	$V_{errI,L}$	100				
$q_3$	0.012	$V_{errI,B}$	100				
$\omega_x$	$\pi$	$V_{errI,R}$	100				

TABLE 11.1: The weight of the state error, integral state and the input.

The controller gain  $\mathbf{K}_s$  is calculated using the MATLAB<sup>®</sup> function `dlqr`. The input arguments to the function are: the discrete system equations, the weight matrices  $\mathbf{Q}$  and  $\mathbf{R}$ .

## 11.5 VERIFICATION

This section intends to document that the designed controller is able to stabilize the X-Pro in a hover condition. First relevant disturbance issues are handled and then the verification is carried out in a Simulink<sup>®</sup> environment where the nonlinear model was implemented.

### 11.5.1 DISTURBANCE ISSUES

When simulating the derived controller on the non-linear model in the Simulink<sup>®</sup> environment, it will prove valuable to incorporate disturbances as this will provide the most realistic environment for the test of the controller. These will be in the form of a plausible state delay from the estimator and a delay when the control signals are sent down through the system to the actuators. It is due to the I<sup>2</sup>C bus configuration, realistic with a sensor sample delay of 10 ms. Another 10 ms is added for the time needed for the estimator to calculate state estimates. This results in an inserted 20 ms state delay between the non-linear model and the controller. For the controller to calculate control signals, 10 ms is accounted for. As the I<sup>2</sup>C is also used for the downstream, a total of 20 ms is inserted between the controller output and the non-linear model.

### 11.5.2 SIMULATION

This section serves to verify by simulation that the controller can stabilize the quad rotor even when exposed to the disturbances mentioned in the previous section. The LQ controller must furthermore maintain the requirements set in Section 9.3.2 on page 74. First the closed loop system is examined through a case study which puts emphasis on a limited time window of the simulation. This is done in order to verify the system by checking for realistic behavior and intuitive response. After this each requirement from the before mentioned specification is handled.

In order to excite the system properly a reference signal in the  $x, y$  and  $z$  position is given. The signal is constituted by  $x : 0 \rightarrow 0.5$  m,  $y : 0 \rightarrow 0.5$  m and  $z : 1 \rightarrow 2$  m, and



steps alternately in all three of the directions before it applies a concurrent step driving the X-Pro back to its initial state.

The Figures 11.2 on page 97, 11.3 on page 98 and 11.4 on page 99 are generated from the latter mentioned reference. Figure 11.2 shows the position in the left column and attitude on the right column,- Figure 11.3 shows the motor voltages on the left column and the reference speeds on the right column, whilst Figure 11.4 shows linear velocity in the right column and angular velocity on the right column.

### Case study

The first step scenario at  $t = 1$  s is reviewed in the following. A step is performed in  $z$  from 1 m - 2 m which can be observed through Figure 11.2(e). From the left column of Figure 11.3 it is seen that each motor receives the same voltage which is correct as the X-Pro is supposed to perform a vertical displacement. However due to differences in the four motor-gear systems the rotors settle at different speeds with the same voltage which is exemplified through small deviations in both attitude, right column of Figure 11.2, and thereby also linear displacement in  $x$  and  $y$ , left column of Figure 11.2. The designed controller provides the reference speeds seen in the right column of Figure 11.3. These ensure that both position and attitude converges to their respective reference.

Other time windows can be treated with equally interesting details of realistic behavior however the latter is considered adequate.

### Position and Attitude

As no ground effect is modelled in the nonlinear Simulink<sup>®</sup> model the controller is tested in a no ground effect position, thereby meeting this requirement.

The maximum angular excitation in attitude with respect to the performed manoeuvre can be calculated through the following.

$$\text{Positive roll} \rightarrow \tilde{q} = [\tilde{q}_1 \quad \tilde{q}_2 \quad \tilde{q}_3]^T = [0.17 \quad 0 \quad 0]^T \Rightarrow \quad (11.59)$$

$$\theta = 2 \cdot \cos^{-1}(\sqrt{1 - 0.17^2}) \approx 19.58^\circ \quad (11.60)$$

$$\text{Positive pitch} \rightarrow \tilde{q} = [\tilde{q}_1 \quad \tilde{q}_2 \quad \tilde{q}_3]^T = [0 \quad -0.17 \quad 0]^T \Rightarrow \quad (11.61)$$

$$\theta = 2 \cdot \cos^{-1}(\sqrt{1 - (-0.17^2)}) \approx 19.58^\circ \quad (11.62)$$

$$\text{Reverse to initial} \rightarrow \tilde{q} = [\tilde{q}_1 \quad \tilde{q}_2 \quad \tilde{q}_3]^T = [-0.14 \quad 0.14 \quad 0]^T \Rightarrow \quad (11.63)$$

$$\theta = 2 \cdot \cos^{-1}(\sqrt{1 - (-0.14)^2 - (0.14)^2}) \approx 22.84^\circ \quad (11.64)$$

The results from the equations above show that the magnitude of the applied steps lies within the boundary of what is accepted according to the LQ controller specifications in Section 9.3.2 on page 74. The angles imply that steps of greater magnitude could be performed, however care should be taken if exciting the system further.

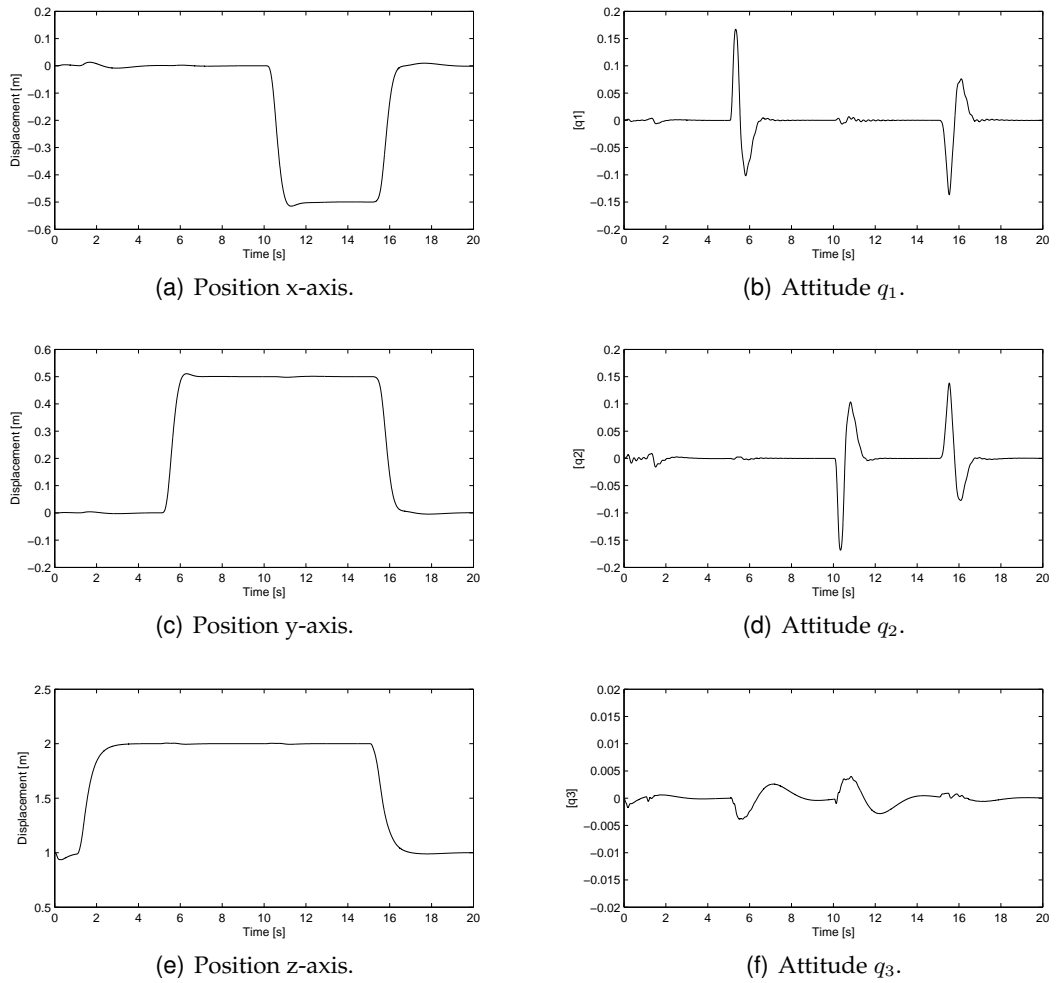
### Reference speed and voltage

The left column of Figure 11.3 on page 98 shows only concise moments of saturation which is acceptable. The right column of Figure 11.3 on page 98 shows the controller generated reference speeds which are bounded to  $\pm 40$  rad/s small signal. This boundary comes from that the maximum rotor speed is  $\approx 190$  rad/s, and as the hover speed is  $\approx 150$  rad/s the  $\pm 40$  rad/s boundaries seems rational. The lower limit is determined from the desire of having an evenly distributed interval around hover speed. This lower limit also ensures that the requirements of a minimum rotor speed  $8\text{ Hz} \approx 50$  rad/s is maintained. Only brief saturation exists in the column of Figure 11.3 on page 98.

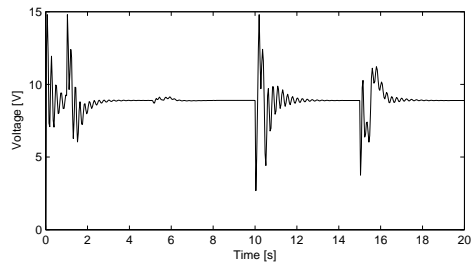
### Linear and angular velocity

The left column of Figure 11.4 on page 99 shows that the linear velocity exceeds the requirements of  $\pm 0.5$  m/s by approximately two times in the x and y axis and three times in the z-axis. The requirement has been set to  $\pm 0.5$  m/s due to safety issues and should be kept, however as the attitude remains within boundaries at all times the controller should be able to control the X-Pro even with steps at this magnitude. The steps applied should be divided by two in the x and y axis and three in the z-axis, to withstand the requirements.

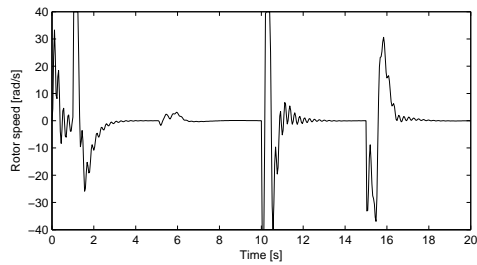
The right column of Figure 11.4 on page 99 shows that the angular velocity meets the requirements of  $\pm \pi$  rad/s as the maximum rotational displacement lies within 3 rad/s.



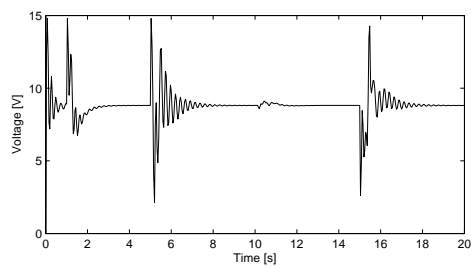
**FIGURE 11.2:** The left column shows displacement in position and the right column shows displacement in attitude.



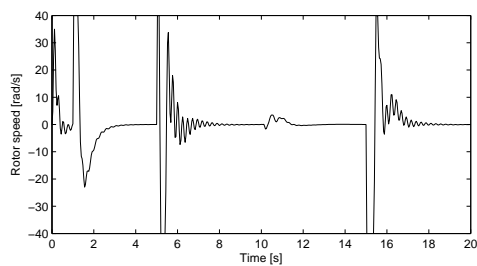
(a) Front motor voltage.



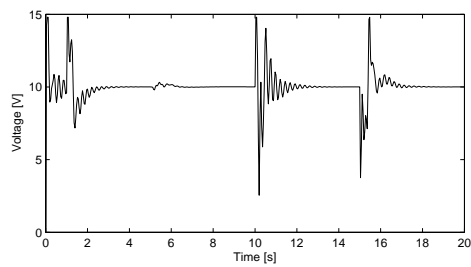
(b) Front reference speed.



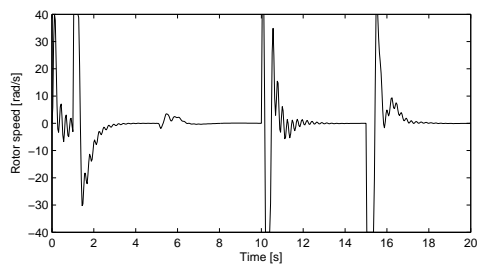
(c) Left motor voltage.



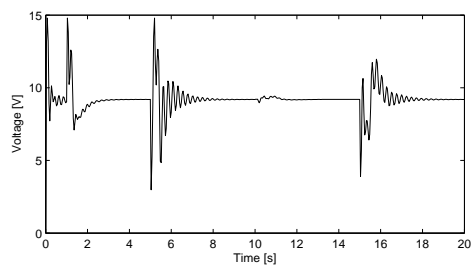
(d) Left reference speed.



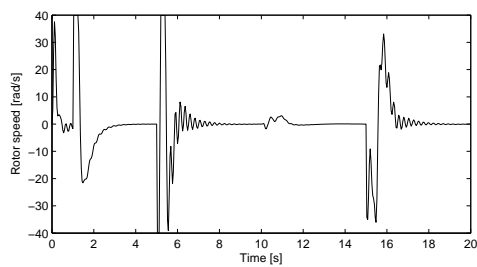
(e) Back motor voltage.



(f) Back reference speed.

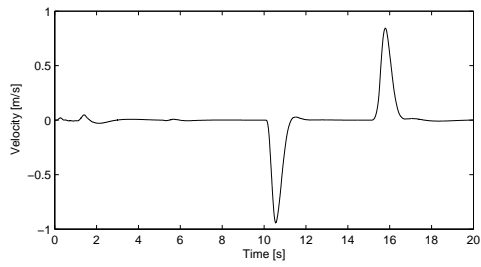


(g) Right motor voltage.

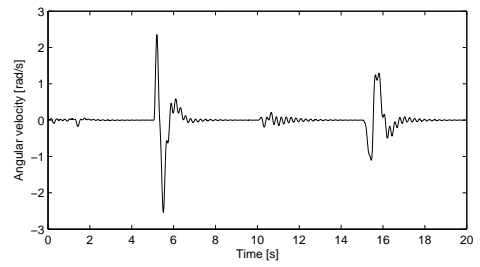


(h) Right reference speed.

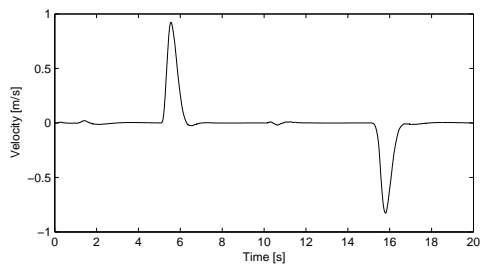
**FIGURE 11.3:** The left column shows the applied motor voltages and the right column shows the controller generated reference speeds.



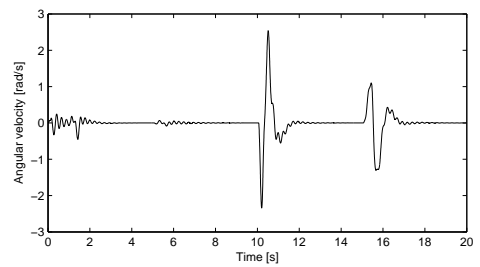
(a) Linear velocity x-axis.



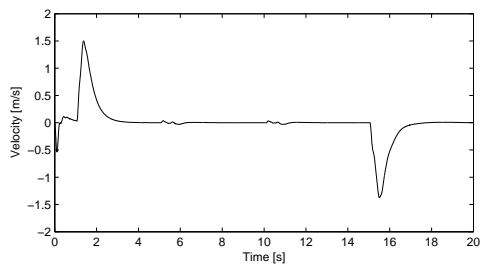
(b) Angular velocity x-axis.



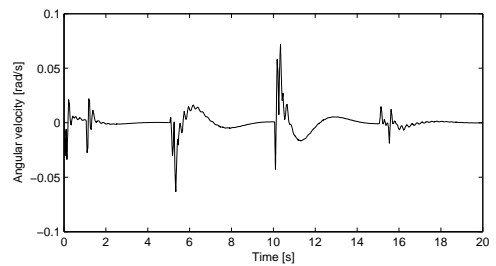
(c) Linear velocity y-axis.



(d) Angular velocity y-axis.



(e) Linear velocity z-axis.



(f) Angular velocity z-axis.

**FIGURE 11.4:** *The left column shows bla in bla and the right column shows bla in bla.*

## 11.6 CONCLUSION

This chapter concludes the Hover controller part by providing a controller structure that can stabilize the X-Pro in hover. The controller structure contains four PI controllers, one for each motor-gear-rotor system, and a LQ state feedback controller. In order to devise the LQ controller, the PI controllers were designed and integrated into the system equations, which later served as basis for the LQ controller. The latter mentioned system was linearized around the hover working point and discretized at 25 Hz, being the control frequency. The discret controller was designed based on the latter and verified through simulation with results as shown in section 11.5 on page 94. The graphs indicate that, although exposed by disturbances as mentioned in Section 11.5.1 on page 94, with an alternating step of the form,  $x : 0 \rightarrow 0.5 \text{ m}$ ,  $y : 0 \rightarrow 0.5 \text{ m}$  and  $z : 1 \rightarrow 2 \text{ m}$ , the controller has no problem stabilizing the X-Pro in its new reference. The controller shows good performance with almost no overshoot and no steady state error. The only requirement not met is the linear velocity which is exceeded by 0.5 m/s, 0.5 m/s and 1 m/s respectively. In order to accomodate this demand the steps can be made smaller, thereby maintaining the requirement. The hover controller is hereby ready to be implemented together with the estimator from [GHB07] in the soft real time target software and loaded onto the quad rotor main computer.

# PART IV

## PIECEWISE-AFFINE HYBRID SYSTEM CONTROLLER

*This part documents the work that has been carried out in order to apply theories of PAHS on convex polytopes to the quad rotor helicopter system. The work is motivated by the theories and results derived in the papers [vSH04], [vSHC06], and extends these theories whenever needed in the search for a feasible solution seen from a practical perspective. A method designated the Combinatoric Controllability Method is derived and presented with the purpose of accommodating certain problems and difficulties, which arise when applying the theories on dynamical systems in general. The part starts off with a nomenclature list which is presented in order to give the reader insight into the domain of PAHS. The preceding chapter 12 gives an introduction to PAHS in general and to the theories of which this work has been based upon. In Chapter 13 the Combinatoric Controllability Method is applied to a simplified quad rotor helicopter system. The last chapter 14 concerns the part closure.*

# CONTENTS OF PART IV

---

<b>12 Introduction</b>	<b>105</b>
12.1 Theories of PAHS . . . . .	105
12.1.1 Case study: Simple two dimensional motor system . . . . .	108
12.2 Hypothesis . . . . .	110
12.3 Control method applied to case study . . . . .	110
12.4 Combinatoric Controllability method synthesis . . . . .	116
12.5 Method applied to quad rotor helicopter system . . . . .	118
<b>13 Combinatoric controllability method</b>	<b>121</b>
13.1 Axis division . . . . .	121
13.2 Construction of a d-cube. . . . .	124
13.3 Facets and normal vectors . . . . .	125
13.4 Facets to close . . . . .	126
13.5 Linearization . . . . .	127
13.6 Form the control space polytope . . . . .	128
13.7 Bad vertex identification and polytope trimming . . . . .	129
13.8 Control law generation . . . . .	134
13.8.1 Simulation of control law on 3-cube . . . . .	137
<b>14 PAHS closure</b>	<b>139</b>

---



# PAHS NOMENCLATURE

Since a relatively unknown area for control engineering is touched in this part, a list of used terms with their meaning and definition placed here. The definitions are mainly taken from [GO97] and [Fuk04].

**Halfspace** A halfspace is effectively described by a linear inequality  $a^T x \leq b$  where  $a$  is a  $d$ -dimensional vector and  $b$  is a scalar, thus for any  $x$  that fulfil the inequality is a part of the halfspace. As the name implies will a halfspace divide the space in two thus providing notion of a discrete location, on one side or the other by testing the inequality.

**Polyhedron** An intersection of finitely many halfspaces in  $\mathbb{R}^d$ .

**d-Polytope** A polytope is an arbitrary bounded solid shaped figure. A polytope is said to be full dimensional if the  $d$ -polytope is described in  $\mathbb{R}^d$ .

**Convex**  $P$  is convex if a straight line between two points in  $P$  belongs entirely to  $P$ .

**Convex hull** The convex hull of a set vertices,  $V$ , is the smallest convex set containing  $V$ . In this thesis denoted by  $conv\{V\}$ .

**Co-planar** Is said about a set of points that are in the same hyper-plane.

**Supporting hyperplane** The hyperplane  $H$  supports  $P$  a face  $F$  if  $P \cap H = F$  and  $P$  lies in one of the closed halfspaces bounded by  $H$ .

**Face** A proper face  $F$  of a polytope  $P$  is the (nonempty) intersection of  $P$  with a supporting hyperplane of  $P$ . The term face is thus a cross dimension term, in fact  $F_i | i = 1 \dots d - 1$ ,  $d$  being the ambient dimension, is all possible face types.

**Facet / Ridge / Egde** is a face of a polytope  $P$  of dimension  $d-1$ ,  $d-2$  and  $1$  respectively.

**Vertex** Is a face of a polytope  $P$  of dimension  $0$  and is thus a point, that can be interpreted as a corner point to  $P$ .

**H- polytope** Is a polytope described using halfspace and is the the intersection of a system of halfspaces.

**V- polytope** Is describing a polytope using the vertices, thus a  $V$ - polytope is the convex hull of a finite set  $X = \{x^1, \dots, x^n$  of points in  $\mathbb{R}^d$

**Simple / simplicial polytope** A  $d$ -polytope is called simple if each vertex is contained in exactly  $d$  facets. A  $d$ -polytope is called simplicial if each facet contains exactly  $d$  vertices.

**Simplex** A polytope is called a simplex if has exactly  $d+1$  vertices and  $d+1$  facets. This structure is thus both simple and simplicial, and is the minimum figure that still span  $\mathbb{R}^d$ .

**d-cube** The d-cube has the same properties as the cube, which in general is a 3-cube. the formal definition is:

$$C_d := \text{conv}\{\alpha_1 e^1 + \alpha_2 e^2 + \dots + \alpha_d e^d \mid \alpha_1, \dots, \alpha_d \in \{+1, -1\}\} \quad (11.65)$$

here defined that the cube contain the origin in its interior, but the notion of cube is used still if the structure is scaled and translated in the space.

**Vertex figure**  $P/v$  If  $v$  is a vertex of  $P$ , then  $P/v := P \cap H$  is the polytope obtained by intersecting  $P$  with a hyperplane  $H$  that has  $v$  on one side and all the other vertices of  $P$  on the other side.

**(0,1)-polytope** A polytope all of whose vertex coordinates are 0 or 1, that is, whose vertex set is a subset of the vertex set  $\{0, 1\}^d$  of the unit cube.

**Isomorphism** Let  $Q_0$  be a set of vertices of  $Q$  and let  $P_0$  be a set of vertices of  $P$ ,  $Q$  and  $P$  are said to be isomorphic, if there is a bijection that maps  $Q_0 \rightarrow P_0$



# INTRODUCTION

In the modeling and control of dynamical systems, one often deals with first order ordinary differential equations describing the evolution over time for a system around a specific working point. However sometimes the linear system proves to be inadequate as it fails to describe the non-linear system in other areas away from that working point. Hybrid characterizations can often be used to combine continuous linear systems through discrete modes to form an improved description of the system at hand.

In the literature, several approaches have been proposed to model these mutual interactions i.e. hybrid interactions, however one particular model first introduced by [Son81], namely the class of Piecewise Affine Hybrid Systems has received considerable attention in recent years. Recently in [vSH04] and [vSHC06], control problems for this class on both full dimensional polytopes and full dimensional simplices, have been considered.

The written work in this part of the master's thesis seeks to utilize the results derived by the previously mentioned authors and furtherly extend the theories whenever needed in order to apply the methods on a quad rotor helicopter system. The following section will introduce the reader to theories and notions that serve as the foundation for the derivation of a Combinatoric Controllability Synthesis method for PAHS. A method that is based on a control problem for affine dynamical systems on a full dimensional convex polytope.

## 12.1 THEORIES OF PAHS

A Piecewise-Affine Hybrid System consists of a discrete automaton with a continuous-time affine system on a polyhedral set at each mode, and a switching mechanism between both discrete and continuous dynamics, [vSHC06, p. 938]. When the continuous state of a particular linear system in a mode reaches the boundary of the corresponding polyhedral set, a discrete event occurs transferring the system to a new discrete mode with a new linear system.

The same approach as [vSH04] and [vSHC06] is made in the sense that continuous affine systems in each mode are defined on full dimensional *convex polytopes*. This makes sense as a polyhedron is the unbounded version of a polytope, and as the property of convexity proves to be a necessity in the derivation of a control law, [vSH04, p. 22].

A polytope  $P_d$  can be characterized as the convex hull of a finite number of points,

denoted by its vertices. A point  $v_i \in \text{conv}(P_d)$ , ( $i = 1, \dots, M$ ), is called a vertex of  $P_d$  if it cannot be written as a convex combination of the points  $v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_m$  contained in  $P_d$ . Formally the convex hull of a subset  $S$  in  $\mathbb{R}^d$  is defined as the smallest convex set in  $\mathbb{R}^d$  still containing  $S$ . A face of a polytope  $P_d$  is the intersection of  $P_d$  with one of its supporting hyperplanes. A hyperplane divides a space into two halfspaces and is said to support  $P_d$  if two requirements are met: If  $P_d$  is entirely contained in one of the two halfspaces of the hyperplane and if  $P_d$  has at least one point contained on the hyperplane. If a face in a polytope  $P_d$  has dimension  $d$ , the faces of  $P_d$  with dimension  $d - 1$  are called facets.

The PAHS can through hybrid system formalism be characterized by the tuple  $\mathcal{H}$  presented in (12.1), [vSHC06, p.939].

$$\mathcal{H} = (Q, E, f, U, \{(X_q, \mathcal{A}_q) \mid q \in Q\}, \{(G_q(e), \mathcal{R}_q(e)) \mid (q, e) \in \text{dom}(f)\}) \quad (12.1)$$

where  $Q$  is the discrete locations,  $E$  the discrete events and  $f$  the discrete transition functions  $f : Q \times E \rightarrow Q$ . These three elements constitute the discrete automaton whereas the continuous affine systems are composed of  $\mathcal{A}_q = (A_q, B_q, a)$  defined on the polytope state set  $X_q$ . The input to the affine systems are the  $U$ , also defined on a polytope. The previously mentioned hybrid interactions between the automaton and the affine systems are described via the guard sets  $G_q(e)$  and the affine reset maps  $\mathcal{R}_q(e) : G_q(e) \rightarrow X_{f(q,e)}$ .

Given a discrete location  $q \in Q$ , the continuous affine systems are defined by the following Equation (12.2).

$$\dot{x}_q(t) = A_q x_q(t) + B_q u(t) + a_q \quad (12.2)$$

where  $x_q \in X_q$  and  $u \in U$ . With (12.1) and (12.2) defined the evolution of the PAHS can be described. Whenever the systems state  $x_q$  leaves the polytope set  $X_q$  an event  $e \in E$  will occur corresponding to the guard set  $G_q(e)$ , provoking a transition  $f(q, e)$  eventually leading to a new discrete location. With this evolution in mind the control problem evaluates to driving a state trajectory to a specific target facet  $F_j \subset X_q$  of the polytope  $P_d$ .

In order to observe in which direction the trajectory  $\dot{x}_q(t)$  evolves, with respect to a given facet  $F_j$ , the normal vector  $n_j$  is employed. The normal vector  $n_j$  points from the surface of  $F_j$  out of polytope  $P$ . If the dot product  $n_j^T \dot{x}_q(t) > 0$  the trajectory has a positive component in the direction of  $n_j$ , hence evolving towards  $F_j$ . If  $n_j^T \dot{x}_q(t) \leq 0$  the trajectory is either along the  $F_j$  boundary or heading away from  $F_j$ , into  $P$ .

These scalar products prove to be the foundation of the [vSH04], [vSHC06] approach towards designing control laws for PAHS. With this in mind it seems intuitive that specific input signals  $U$  can maintain the inequalities, achieving a desired control objective. When it furthermore proves to be enough to apply the inequalities on the inputs  $U$  at the vertices of the polytope  $P_N$ , [vSHC06, p. 21], this indeed suggests that facets can be *closed* or *left open*. Closed in the sense of rejecting  $\hat{x}_q$  admission through  $F_j$  and open as in granted access through  $F_j$ . Proposition 12.1.1, [vSH04, p. 24], deals with the calculation of control signals to open or closed facets. In the Proposition the open facet is  $F_1$ .

**Proposition 12.1.1**

- (1)  $\forall j \in V_1$  :
  - (a)  $n_1^T(Av_j + Bu_j + a) > 0$
  - (b)  $\forall i \in W_j \setminus \{1\} : n_i^T(Av_j + Bu_j + a) \leq 0$
- (2)  $\forall j \in \{1, \dots, M\} \setminus V_1$  :
  - (a)  $\forall i \in W_j : n_i^T(Av_j + Bu_j + a) \leq 0$
  - (b)  $\sum_{i \in W_j} n_i^T(Av_j + Bu_j + a) < 0$

where  $V_1$  is the vertex set in facet  $F_1$ . Proposition 12.1.1.(1) is utilized when calculating control signals, to all the vertices contained in facet  $F_1$ , whilst Proposition 12.1.1.(2) is used when calculating control signals for all vertices not contained in  $F_1$ .

The papers [vSH04] and [vSHC06] present theories to control law generation and synthesis on convex polytopes and on simplices. Common for both papers is that they either define their geometric structures as simplices, which is a polytope with  $d + 1$  vertices and convex per definition, or they treat convex polytopes which they later triangulate (to simplices) in order to construct unique control laws. It is exactly the uniqueness in the solutions that occupies [vSH04] and [vSHC06].

The affine control law  $u = Fx + g$  is derived by means of (14) in [vSH04, p. 30] which is stated in (12.3).

$$\begin{bmatrix} v_1^T & 1 \\ \vdots & \vdots \\ v_{N+1}^T & 1 \end{bmatrix} \begin{bmatrix} F^T \\ \hline g^T \end{bmatrix} = \begin{bmatrix} u^T \\ \vdots \\ u_{N+1}^T \end{bmatrix} \quad (12.3)$$

It is evident from (12.3) that a unique solution can be found for  $F$  and  $g$ , if the system is defined on a simplex, as the vertex matrix is completely square. Unless the matrix is ill conditioned, it will prove invertible, thereby leading to a unique solution for  $F$  and  $g$ .

If the system was defined on a convex polytope with more than  $d + 1$  vertices, the vertex matrix would be  $m \times n$ , thereby not leading to a unique solution. Even as this is not considered an option in [vSH04] and [vSHC06], this thesis is determined to apply the pseudo inverse of the vertex matrix, which yields the least squares solution (best linear approximation to the actual solution). It should be noted that the pseudo inverse is one of many solutions, thereby not unique. The least squares solution would in practice lead to a control law that maintains the control objective with an absolute minimum of actuator action. This would be interesting from an engineering point of view as energy efficient and actuator wearing could be of importance.

The pseudo inverse of the vertex matrix in (12.3) leads to the following in (12.4).

$$\begin{bmatrix} F^T \\ \hline g^T \end{bmatrix} = \begin{bmatrix} v_1^T & 1 \\ \vdots & \vdots \\ v_{N+1}^T & 1 \end{bmatrix}^+ \begin{bmatrix} u^T \\ \vdots \\ u_{N+1}^T \end{bmatrix} \quad (12.4)$$

with the pseudo inverse of a matrix  $B$  defined as  $B^+ = (B^T B)^{-1} B^T$ .

If one were to apply the theories from this Section on a dynamical system, questions would arise as to how the polytopes should be generated in order to capture the state space in a feasible and computer tractable way. If the axes of two given states were split up, the dependencies in between could be described as a square or rectangle. If a square was considered and increased by another state, one could imagine a cubic figure. If the geometric figure was furtherly increased by states, hyper-cubes or hyper-rectangles would appear. The following case study of a simple two dimensional motor system will seek to elaborate on this latter paragraph and derive a control law.

### 12.1.1 CASE STUDY: SIMPLE TWO DIMENSIONAL MOTOR SYSTEM

This section will seek to apply the above defined PAHS theories in a practical case study, a simple two dimensional motor system. The case study will eventually reveal problems when the theories are applied. The state space representation of the two dimensional motor system can be defined as in (12.5).

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \quad (12.5)$$

The dependency between the two states can be visualized through the square form shown

in Figure 12.1. As the Figure shows there are two polytopes  $P_1$  and  $P_2$ . It is impor-

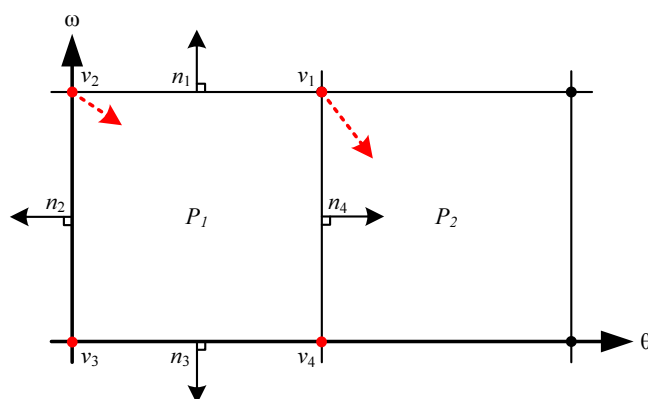


FIGURE 12.1: The state space is divided into squares. The red arrows and dots denote the trajectories of the system in Equation (12.5). The facets  $F_1$  and  $F_4$  are furthest away from the origin, and are therefore to be closed.

tant to notice that every square contains a unique linearized affine system. In this case study only  $P_1$  is considered and chosen as a unit square. The values of the four vertices,  $v_1, \dots, v_4$ , and normal vectors  $n_1, \dots, n_4$  are listed in Table 12.1.

Vertex	Value	Normal	Value
$v_1$	(1, 1)	$n_1$	(0, 1)
$v_2$	(0, 1)	$n_2$	(-1, 0)
$v_3$	(0, 0)	$n_3$	(0, -1)
$v_4$	(1, 0)	$n_4$	(1, 0)

TABLE 12.1: The vertices and normal vectors to  $P_1$  in Figure 12.1.

The system is at rest when  $\omega = 0$ , this gives that the facets  $F_1$  and  $F_4$  are wanted closed to ensure that the trajectory does not escape through these facets. To determine whether the facets can be closed, Proposition 12.1.1 on page 107 is used.

The Proposition states that if a vertex belongs to a closed facet set, Proposition 12.1.1.(2a) is to be used. As the closed facets are  $F_1$  and  $F_4$ , the only vertex belonging to the closed set is  $v_1$ . In order to determine whether the facet can be closed or not, a control signal to the given vertex has to be found. The calculation of 12.1.1.(2a) is carried out in Equation (12.6).

Proposition 12.1.1

$$(2a) \quad n_4^T (Av_1 + Bu_1 + a) \leq 0$$
$$\begin{bmatrix} 1 & 0 \end{bmatrix} \left( \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_1 \right) \leq 0 \quad (12.6)$$
$$1 + 0 \cdot u_1 \leq 0 \Rightarrow u_1 \in \emptyset$$

Due to the fact that no control signal to  $v_1$  can be found, the facet can not be closed and there exists no control law to  $P_1$ . This implies that  $F_4$  cannot be closed, hence will continue into the next polytope,  $P_2$ . This behavior is unwanted and a solution solving the mentioned problem is wanted. Therefore a hypothesis concerning the problem is stated in the following section.

## 12.2 HYPOTHESIS

From the previous, it seems to be desirable to part the state space of the dynamical system into squares or rectangles, however from the papers, a unique control law can only be found if applied on simplices. Further, if calculation of a control signal in a given vertex of the cube or rectangle fails, there is no method stated solving the resulting problem, hence the control law cannot be derived. With this in mind the following hypotheses can be stated.

*It is possible to partition the state space into rectangles or squares, these still being convex polytopes, not necessarily constructing a unique control law, however a sufficient one for control purposes. It is believed to be able to solve the problem formulated in the case study above, by formulating a method which alters the polytope structure until a control law can be applied.*

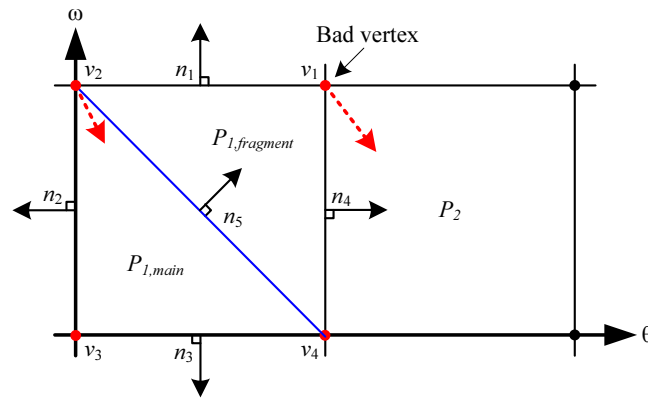
The following section will try to prove the hypothesis by continuing to solve the problem, stated in the case study.

## 12.3 CONTROL METHOD APPLIED TO CASE STUDY

This section states a method to surpass the problems described in the previous, thereby accommodating the hypothesis. The method proves its worth through combinatoric reasoning with respect to the polytope at hand.



The basic idea will be described through a practical application on the previously treated motor system. As the control signal  $u_1$  was impossible to generate at vertex  $v_1$ , the facet  $F_4$  could not be closed. An option is to restrict the space i.e. the geometric figure to one containing vertices that can be generated control signals for. It is proposed to *trim* the polytope  $P_1$  by discarding the *bad* vertex  $v_1$  and connecting  $v_2$  with  $v_4$  to form the new polytope  $P_1$  main structure. The main structure is defined as the main region where control can be performed, while the fragment is defined as the vertex figure of the bad vertex. The principle is illustrated in Figure 12.2.



**FIGURE 12.2:** Illustration of the main idea behind the combinatoric control method. The new facet  $F_5$  with normal vector  $n_5 = (1/\sqrt{2})[1 \ 1]$  appears after bad vertex  $v_1$  has been trimmed from  $P_1$ . This allows a controllable space within the new main structure of  $P_1$ . The  $P_1$  fragment is denoted the vertex figure of  $v_1$ .

In this way a new facet  $F_5$  arises with a new normal vector  $n_5 = (1/\sqrt{2})(1, 1)^T$  and the neighbour vertices  $v_2$  and  $v_4$  to the bad vertex  $v_1$  are now, from a control perspective, relative to  $n_5$ . As this eventually solves our problem it is noted that a harder constraint is put on the input at vertex  $v_2$ .

It is important at this point to emphasize that the vertex figure in  $v_1$  is preserved to uphold the isomorphic property, not only on the facet connecting the main structure of  $P_1$  with the vertex figure  $v_1$  of  $P_1$  ( $P_{1,fragment}$ ), but also on the facet connecting polytopes  $P_1$  and  $P_2$ . In this way it is still allowed for the state to reside inside vertex figure  $v_1$ . If the state in fact resides within vertex figure  $v_1$ , only a departure through facet  $F_4$  could be ensured, but as polytope  $P_2$  could be trimmed as  $P_1$  this would prove sufficient. The following will concern the generation of a control law for the main structure of  $P_1$ . The main goal for the derived control law is to prevent the system trajectory to enter the less controllable region i.e. vertex figure  $v_1$ . Another considered goal is to drive the trajectory

towards  $F_3$  as this eventually would lead to an equilibrium point.

As the new facet  $F_5$  is wanted closed, the use of Proposition 12.1.1.(2) can be applied to the vertices contained in the closed facet set,  $v_2$  and  $v_4$ . If  $v_2$  is considered, discussion can be made as to whether facet  $F_2$  should be closed or left open. If closed,  $v_2$  would be treated according to the before mentioned Proposition 12.1.1.(2), thereby applying a constraint in the form of an inequality on the input  $u_2$ . If open, vertex  $v_2$  would have to be treated with respect to Proposition 12.1.1.(1), also generating an inequality stating a harder constraint on  $u_2$ , as the actuator would have to try to push the trajectory opposite the systems evolving tendency. Proposition 12.1.1 is followed in the search for input signals at the vertices for the main structure of  $P_1$ . This indicates that  $F_2$  is regarded as closed leading to the inequalities for vertex  $v_2$ , stated in the following.

Proposition 12.1.1

$$(2a_1) \quad n_5^T(Av_2 + Bu_2) \leq 0$$

$$(1/\sqrt{2})[1 \ 1] \cdot \left( \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_2 \right) \leq 0$$

$$(1/\sqrt{2}) + (1/\sqrt{2})u_2 \leq 0$$

$$u_2 \leq -1$$

$$(2a_2) \quad n_2^T(Av_2 + Bu_2) \leq 0$$

$$[-1 \ 0] \cdot \left( \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_2 \right) \leq 0$$

$$-1 + 0 \cdot u_2 \leq 0$$

$$u_2 \in \emptyset$$

$$(2b) \quad (1/\sqrt{2})[1 \ 1] \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_2 + [-1 \ 0] \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_2 < \dots$$

$$- (1/\sqrt{2})[1 \ 1] \begin{bmatrix} 1 \\ 0 \end{bmatrix} - [-1 \ 0] \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$(1/\sqrt{2})u_2 < -(1/\sqrt{2}) + 1$$

$$u_2 < \sqrt{2} - 1$$

(12.7)

As can be observed from 12.7(2a<sub>2</sub>), that a control signal can not be derived for vertex  $v_2$  at facet  $F_2$ . In this case one must rely on the system trajectory and as  $-1$  indicates

a negative component of the dot product, the trajectory is directed away from  $F_2$ . This is considered acceptable and should in turn complement the previous discussion with respect to the opening or closing of  $F_2$ .

As two constraints  $u_2 \leq -1$  and  $u_2 < \sqrt{2} - 1$  are put on  $u_2$ , the resulting interval in which a specific  $u_2$  must be chosen is  $u_2 \leq -1$ .

The following statement (12.8) derives the control signal for  $v_4$ .

**Proposition 12.1.1**

$$(1a) \quad n_3^T (Av_4 + Bu_4) > 0$$

$$[0 \quad -1] \cdot \left( \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_4 \right) > 0$$

$$-u_4 > 0$$

$$u_4 < 0$$

(12.8)

$$(1b) \quad n_5^T (Av_4 + Bu_4) \leq 0$$

$$(1/\sqrt{2})[1 \quad 1] \cdot \left( \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_4 \right) \leq 0$$

$$(1/\sqrt{2})u_4 \leq 0$$

$$u_4 \leq 0$$

The two inequalities in (12.8) state that  $u_4$  must be chosen so that it maintains  $u_4 < 0$ .

With facet  $F_2$  defined as closed and  $F_3$  open, the control signal for  $v_3$  cant be derived as in the following.

Proposition 12.1.1

$$\begin{aligned}
(1a) \quad & n_3^T (Av_3 + Bu_3) > 0 \\
& [0 \ -1] \cdot \left( \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_3 \right) > 0 \\
& -1 u_3 > 0 \\
& u_3 < 0 \\
(1b) \quad & n_2^T (Av_3 + Bu_3) \leq 0 \\
& [-1 \ 0] \cdot \left( \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_3 \right) \leq 0 \\
& 0 \cdot u_3 \leq 0 \\
& u_3 \in \emptyset
\end{aligned} \tag{12.9}$$

hence  $u_3$  must be chosen in the interval  $u_3 < 0$ .

The inequalities in (12.7), (12.8) and (12.9) provides the basis for the derivation of a sufficient control law, leading the trajectory towards facet  $F_3$ . A choice of inputs is first apparent and these can eventually freely be chosen within the inequality stated boundaries and within actuator saturation boundaries. Given an optimization criterion the corresponding affine control law depends on the initial condition. So as the initial condition varies, this leads to a nontrivial high complexity procedure of always finding the optimal control, [vSH04, p. 27]. Therefore in the search for a control law the following inputs are chosen,  $u_2 = -2$ ,  $u_3 = -4$   $u_4 = -1$ .

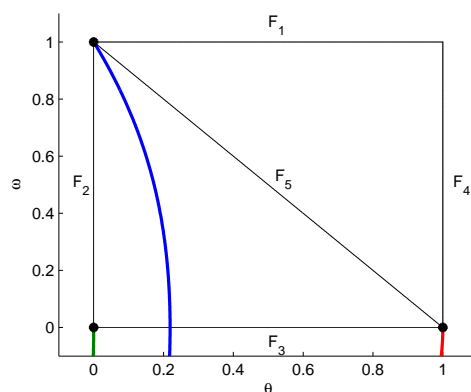
The affine control law  $u = Fx + g$  is now ready to be determined for the main structure. With the main polytope vertices  $v_2 = (0, 1)$ ,  $v_3 = (0, 0)$ ,  $v_4 = (1, 0)$  and inputs  $u_2 = -2$ ,  $u_3 = -4$   $u_4 = -1$  inserted into (12.3) on page 107 we have (12.10).

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ g \end{bmatrix} = \begin{bmatrix} -2 \\ -4 \\ -1 \end{bmatrix} \tag{12.10}$$

As the resulting main polytope in the motor example has a simplicial nature a unique solution can be found as the vertex matrix in this situation proves to be invertible. However for principle reasons the pseudo inverse solution is still be applied to 12.3 and obviously gives the same resulting control law defined in (12.11).

$$u = \begin{bmatrix} 3 & 2 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} - 4 \quad (12.11)$$

As the control law maintains Proposition 12.1.1, an exit through  $F_3$  is ensured in finite time. The following Figure 12.3 shows the simulation of the control law on the main structure.



**FIGURE 12.3:** Results of simulation of the control law on the main structure of the motor system.

As Figure 12.3 shows the trajectories, started in the three vertices, leaves through the desired  $F_3$ , the control law is verified.

A control law for the vertex figure  $v_1$  can also be derived, thereby closing  $F_1$ , through theories presented in this section. It is however considered out of scope to present this solution here. This concludes the control method application to a simple two dimensional motor system and the method can now be synthesized in to a compact form designated the *Combinatoric Controllability method*.

## 12.4 COMBINATORIC CONTROLLABILITY METHOD SYNTHESIS

This section serves to synthesize the method introduced in the previous section into a structured form denoted the *Combinatoric Controllability Method*. The following will discuss what provided the motivation for the derivation of the new method and sum up the acquired knowledge at this point.

In the previous section some of the theories of PAHS was used which was first presented in [vSH04], [vSHC06]. The idea presented in these papers, handle linear systems in isomorphic convex polytopes in a structured manner that fits into the already extensive framework of Hybrid Systems (HS). One of the forces of HS is the possibility to split the control problem into smaller decoupled problems. In PAHS these sub problems are represented in each polytope, which we remember was isomorphic to the neighbouring polytopes. More specifically the decoupledness is ensured by the notion of closing facets and ensuring exit in finite time, by examining the trajectory of the linear system at the vertices of the polytope. The papers work towards an automated control synthesis and the basic idea also seems clean and intuitive, but since the publication of the idea was made recently, there are still some topics that need special attention.

The algorithm proposed in [vSH04] states that even though the control is made on a polytope, the control law can only be formed by triangulating that polytope. The paper [vSHC06] works altogether on simplices. Dividing the state space into simplices and controlling where the facets are formed is not a trivial task. Not less by the fact that one has problems in conceiving more than three dimensions, and the paradox is that it is in the higher dimensions the method should prove its worth. Indeed there are algorithms for performing triangulations on polytopes, but most algorithms are not optimal in the number of produced simplices, and existing implementations can not accommodate special wishes from the designer on internal facet placing. Imagine if the added facet in Figure 12.2 on page 111 (Section 12.3) was drawn as to generate the opposite diagonal, then the control is difficult to obtain. As it will become evident later, the data storage needed is substantial. Triangulation will only push in the wrong direction. Both methods stated in [vSH04] and [vSHC06] fails to form a control law if the polytope worked on, does not allow a possible control signal in just one of the vertices. Should this happen, it will mean that calculations have to start over at least for that polytope. It seems that if the linear system was used to form the shape of the polytope the method would succeed more often. In the following the *Combinatoric Controllability* method will be presented,

that uses most of the method presented in [vSH04], but where it has difficulties, additions are made according to the results drawn from Section 12.3.

The chosen starting point is to form N-cubes that cover the area of the state space where control is wanted. These N-cubes are intuitive to form since the basis for this lies in doing N one dimensional axis divisions, N being the number of states or equivalently the number of dimensions. The axis divisions can with advantage be done on the basis of the non-linear model, thus reducing the chance of needing to go back to redefine axis divisions. The description of the cube consist of a set of vertices, information on which vertices belong to which facets and the normal vectors following these. Having the skeleton in place each polytope can be treated individually. Each polytope has a task of ensuring that the trajectory will move towards the control goal, which can be a point in the state space. The choice of which facets to close can be done automatic only by knowing the overall control goal and the location of the polytope in the state space. The control problem for the polytope at hand is now locally defined as a problem of closing facet that is wanted closed, and ensuring exit in finite time. As it turned out in the simple motor example in Section 12.3, some vertices can turn out impossible to generate a control signal for that guides the trajectory in the wanted direction. The solution to this is to slice this vertex away from the cube, meaning that the cube now consist of the main structure, where control still is feasible, and a vertex figure or simply just a fragment of the cube. For every bad vertex that can be found a new slice is taken from the main structure. Say that no bad vertices are left in the main structure, this will mean that a control law can be made for the region covered by the main structure. The fragments still need to have a control law calculated but with fewer facets closed than first intended.

For each structure is it now possible to form a polytope in the control space. The interior of this polytope will now contain all possible control signals that fulfil the stated control problem. Having the control laws for the whole cube makes it possible to save the end result and start on the next cube in the state space. When a control law for all cubes has been found the system can be fitted into the overall PAHS tuple  $\mathcal{H}$ .

The following Proposition 12.4.1 states the *Combinatoric Controllability method* in a compact form.

**Algorithm 12.4.1**

1. *Axis division*
2. *Construction of cube  $n$ , using axis divisions*
3. *Identify facets and normal vectors*
4. *Define facets to close*
5. *Form control space  $U$  for each vertex*
6. *Bad vertex identification and polytope trimming.*
  - (a) *Identify bad vertices, if none, **break**.*
  - (b) *Trim one bad vertex, results in fragment and new main structure.*
  - (c) *Form new common facet.*
  - (d) *Revise facets to close in both fragment and main structure.*
  - (e) *Go to (a).*
7. *Control law generation on main and fragment structures.*
  - (a) *Choose control signal from set of points in  $\text{conv}(U)$ .*
  - (b) *Find control law  $u = Fx + g$  on each polytope.*
8. *Move to next cube if exist go to (2), **else** all control laws are found, **break**.*

The outcome of the above stated method when run on all polytopes are the control laws for all structures in all polytopes. Left is to connect the polytopes through guard sets and reset maps to comprise the full PAHS system.

## 12.5 METHOD APPLIED TO QUAD ROTOR HELICOPTER SYSTEM

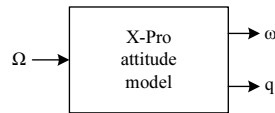
When applying PAHS on higher order system than the 2 dimensional motor example, there are some considerations that are important to take into account. One is the amount of data needed to represent all polytopes in the  $d$ -dimensional space. Looking at the model of the quad rotor helicopter it has 24 states. To see whether all states can be included when applying the PAHS control law the following can be stated. The number of polytopes is determined on the basis of the divisions of the axes. If each axis division is chosen to two, which is the minimum number, the number of polytopes are equal to



$2^{24} \approx 17 \cdot 10^6$ . If one was to add one division to any of the axes the number of polytopes is doubled. Therefore the number of states is to be reduced in order to be able to calculate a control law.

The system has integral states, and these can be removed without any concern of loss of dynamic characteristics. This gives a system of 20 states, and further the rotor speed controllers can be removed, which results in a dimension of 16. If all these states are included the minimum polytopes is  $2^{16} = 65536$  polytopes. Still this number is considered too large, hence the system must be divided into smaller parts.

The dependences in the linear model of the X-Pro, can be stated by the following:  $\Omega_r \rightarrow {}^1\omega_1 \rightarrow {}^0_1q \rightarrow {}^0v_1 \rightarrow {}^0P_1$ . These dependences can furtherly be divided into two, namely  $\Omega_r \rightarrow {}^1\omega_1 \rightarrow {}^0_1q$  and  ${}^0v_1 \rightarrow {}^0P_1$ . This division is only valid if there is a overlying supervisor which choses the correct attitude control and then updates the position controller based on the attitude control, in order to keep the X-Pro within the control objective. With this in mind, the system is now reduced to a 10 dimensional system. For simplicity the dynamics of the motor-gear-rotor system is considered being more than a decade faster than the dynamics of the X-Pro body. Since  ${}^1\omega_1$  uses the rotor speed in its calculations, the rotor speed is chosen as input to the attitude model. With this, the total system dimension is limited to 6. Figure 12.4 shows the delimited quad rotor system.



**FIGURE 12.4:** *The delimited attitude model in 6 dimensions, namely the angular velocity and attitude.*

As the *Combinatoric Controllability method* has been derived and tested on the simple two dimensional motor system from the case study, the next chapter will concern the application of the method to the six dimensional system shown in Figure 12.4.





# COMBINATORIC CONTROLLABILITY METHOD

# 13

This chapter revolves around the application of the *Combinatoric Controllability Method*, stated in Proposition 12.4.1 on page 118, to the simplified attitude system of the quad rotor. The method is briefly restated below.

1. Axis division
2. Construction of cube  $n$ , using axis divisions
3. Identify facets and normal vectors
4. Define facets to close
5. Form control space  $U$  for each vertex
6. Bad vertex identification and polytope trimming.
7. Control law generation on main and fragment structures.
8. Move to next cube *if exist* go to (2), **else** all control laws are found, **break**.

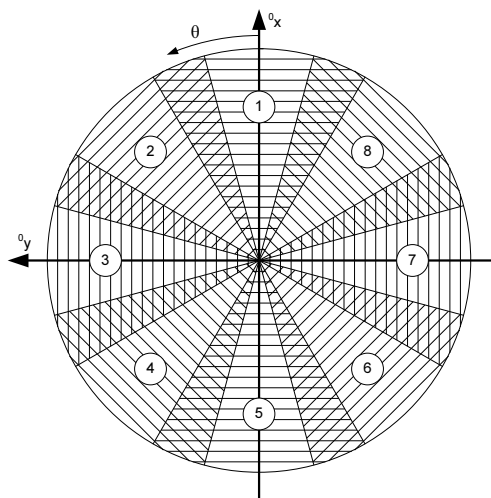
The following sections reflect the structure of the method from the first point to the last.

## 13.1 AXIS DIVISION

The general idea of a piecewise linear system representation is not new but the idea is still very relevant and power full tool for describing non-linear systems. When modelling any dynamical system it becomes clear by the non-linearities, which of the states needs the most divisions. The precise number of divisions is a tunable parameter that in one end, uses few divisions but has a poor description of the non-linear system underneath, otherwise using many divisions will provide better approximations by the linear systems in each discrete location, but as it was evident with the number of states, a increase in divisions will also result in an explosion in data that needs storage. Not many general notes on this topic can be made other than the choice of division must be founded in the control goals and the tolerance for error.

In the specific case the system at hand includes six states, most of these include areas that does not physically relate, one would be that the X-Pro is upside down. It is estimated that an angle of 30 degrees away from the horizontal plane will be possible to

bear, but the system should avoid crossing this boundary. Also from the interface specification we have limitations on the angular rate which are set to  $\pi \text{ rad/s}$ . One axis that does not have unsafe areas is the  $q_3$  state. This state translates to the pointing direction in the horizontal plane. From an intuitive point of view it should not matter which direction the X-Pro has as long as it is level in the air. Nevertheless will the linearisation cause that it is not possible to divert much from this linearisation point without producing wrong and even destabilising control signals. This non-linearity calls for division, but not in the general sense. The quaternion can be seen as a point on the surface of a 4-sphere, in practice this mean that when the X-Pro is turned more than 180 degrees, there comes a time where  $q_3$  flips sign. One way to contain this behaviour in a piecewise linear system is to place a discrete transition where this occurs. Having said that the divisions should form planar boundaries makes it impossible to place this boundary in a correct manner as the sign flip is related to all elements of  $q$ . The solution to this problem is simply not to handle this transient behaviour inside the PAHS state space, but instead keep track of the linearisation point in a discrete frame outside of the euclidean state space. On Figure 13.1 eight discrete locations, referring to eight different linearisations is depicted.



**FIGURE 13.1:** *The chosen parting of the  $q_3$  state, due to cyclic behaviour. This state is to be handled outside the general framework.*

Each location and thus linearisation span over 60 degree around the earth frames  $z$ -axis, the eight locations makes it possible to have an overlap between two neighbour locations. With this parting it is possible to have any  $q_3$  value as control objective. Without the overlap it would in practice be possible for some sort of zeno effect if the intersection

between two locations where chosen as control goal. The handling of the attitude then is a job of keeping the following equation updated:

$$q = q_{wp} \cdot \tilde{q} \quad (13.1)$$

Each time a discrete event occurs on the boundary of  $\tilde{q}_3$  the  $q_{wp}$  is rotated 45 degrees in the direction dictated by the event. Besides this, the reset map will have to move the present continuous state back inside the  $\tilde{q}_3$  discrete location. All states is thus represented as the small signal deviation in the state space.

Concerning  $\tilde{q}_1$  and  $\tilde{q}_2$  it is chosen to form narrow regions near the equilibrium as it is needed to have precise control in this region. The limit on this boundary is set to what equals five degrees away from the horizontal plane. To fit into the square boundaries seen in the  $\tilde{q}_1$ - $\tilde{q}_2$  plane both the 30 degree and the five degree limit is calculated as in Equation 13.5

$$q_{1:3} = \hat{e} \cdot \sin(\theta/2) \Rightarrow \quad (13.2)$$

$$q_1 = q_2 = \frac{1}{\sqrt{2}} \sin(\theta/2) \Rightarrow \quad (13.3)$$

$$\theta = 5^\circ \Rightarrow q_1 = q_2 \approx 0.03 \quad (13.4)$$

$$\theta = 30^\circ \Rightarrow q_1 = q_2 \approx 0.24 \quad (13.5)$$

From the start it is known that it is wanted to drive the system to the critical point, and since all axes in the state space are small signal deviations from a working point, this critical point is the origin. As it was evident in the the simple motor example, it was not possible to form a controller if the system had no vertices directly on the axes, since the critical point was the  $\theta$  axis. This observation can directly be translated to the need of a boundary placed in zero on all axes.

The choices for all boundaries have been presented and they are summed up in Figure 13.2.

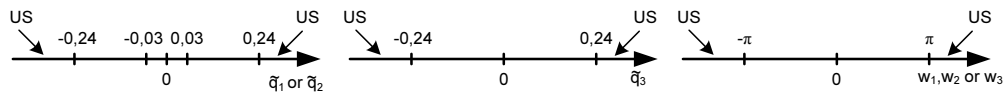


FIGURE 13.2: The divisions on all 6 axis. The term US is denoting an Unsafe Area, an area where the system is not defined and by all means these areas should be avoided to ensure safe operation.

As can be seen there is four segments on  $\tilde{q}_1$  and  $\tilde{q}_2$ , two segments on  $\tilde{q}_3, \omega_1, \omega_2$  and  $\omega_3$ , by simple multiplication the total number of polytopes can be calculated to 256 discrete locations. Using the axes it is possible to create all vertices in the state space and gather the vertices to form convex hyper cubes, which is exactly what the next section concerns.

## 13.2 CONSTRUCTION OF A D-CUBE.

Forming a cube of  $d$  dimension can intuitively be grasped according to the following. Start with the 0-cube which is a point, the zero in front of cube denotes the dimension it is described in thus making the term self-explanatory and unique. Now imagine that space has 1 dimension and that a copy of the structure is slid one unit along the new dimensions axis, the result is a 1-cube, which is a line. In two dimensions the former result is copied and slid one unit and the result is a 2-cube, which also is a square. The next step produces the cube, or 3-cube to follow the terminology. Performing this simple operation  $d$  times will produce the  $d$ -cube. Observe that the number of vertices used from each time step multiplies by two, thus the number of vertices for the N-cube is  $2^d$ . Remember that a facet can be seen as a part of a half-space, thus span  $d-1$ . For each dimension in the cube there are two facets, which generalizes to that the number of facets of the  $d$ -cube is  $2 \cdot d$ . Having six states resulting in a cube that 64 vertices and 12 facets, is the structure we seek to produce in the following. The first the numbering of cubes is to be defined, and an example of the numbering is shown in figure 13.3.

As it can be seen is the notion extendible to dimension N, as the numbering of each axis is done separate. Provided the axes divisions it is possible for the  $2^d$  vertices that form the  $d$ -cube. In three dimensions the eight vertices are listed in Equation (13.6), where e.g.  $X_1$  is the first boundary on the x-axis, thus a look-up in specific axis division.

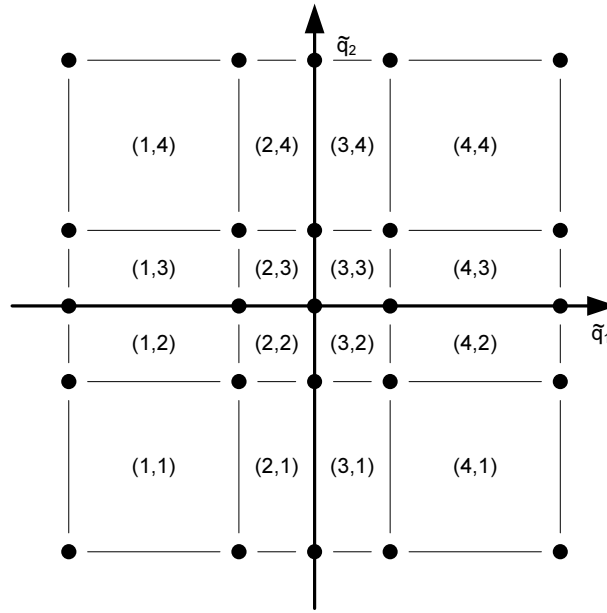


FIGURE 13.3: The principle of numbering cubes here shown on two dimensions.

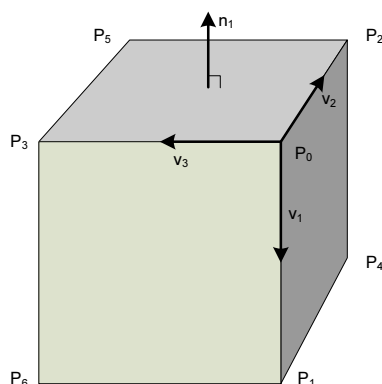
$$\begin{aligned}
 n &= (n_x, n_y, n_z) \\
 C_n &= \begin{pmatrix} (X_{n_x} & Y_{n_y} & Z_{n_z}) \\ (X_{n_x+1} & Y_{n_y} & Z_{n_z}) \\ (X_{n_x} & Y_{n_y+1} & Z_{n_z}) \\ (X_{n_x+1} & Y_{n_y+1} & Z_{n_z}) \\ (X_{n_x} & Y_{n_y} & Z_{n_z+1}) \\ (X_{n_x+1} & Y_{n_y} & Z_{n_z+1}) \\ (X_{n_x} & Y_{n_y+1} & Z_{n_z+1}) \\ (X_{n_x+1} & Y_{n_y+1} & Z_{n_z+1}) \end{pmatrix} \quad (13.6)
 \end{aligned}$$

From Equation (13.6) it is clear to see an implementation of this notion, using  $d$  for loops inside each other.

### 13.3 FACETS AND NORMAL VECTORS

The choice of  $d$ -cubes makes the identification of facets and the corresponding normal vectors relatively easy, mainly due to that each facet is orthogonal to one axis of the coordinate frame. The information that is sufficient for using the PAHS theories is a notion

on which vertices belong to a facet and what the normal vector to this facet is. In the following a possible implementation is stated.



**FIGURE 13.4:** Finding facets and normal vectors can be simplified knowing the properties for the general  $N$ -cube.

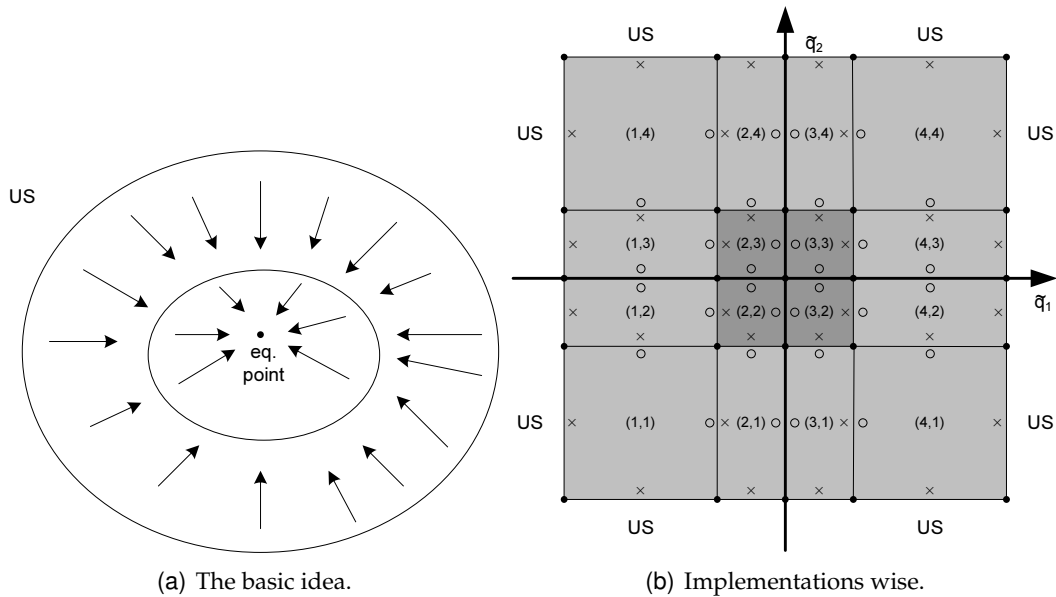
Figure 13.4 shows the 3-cube which can be used as a verification of the method. Choose any vertex in the set of  $2^d$  vertices. Using this vertex,  $v_b$ , as the basis we can find  $d$  facets for which this vertex is a part of. The start is to identify  $d$  edges from the basis vertex which is the same as to identify the neighbour vertex set to  $v_b$ . Towards all these vertices vectors are made, and combining  $d - 1$  vectors will span one of the  $2 \cdot N$  facets. The normal vector to this facet could be found by making the cross product, as it generalises to  $d$  dimensions with some modification, which will be handled later. Instead it can be used that the edges of the cube and thus the formed vectors are orthogonal on each other. This means that the vector that was not used to the facet is the inward normal vector and by flipping sign in this vector and normalising it the normal vector is formed. Determining which vertices that lie in this facet, can be done by verifying if the vectors in the facet span the vector constructed from  $v_b$  to all others vertices.

## 13.4 FACETS TO CLOSE

The whole idea to close facets serves the purpose to guide the trajectory of the continuous system. At this point it should also be clear that one can only state a wish to close a facet, it is entirely up to linear system and actuators to grant this wish for this specific facet. The general idea is depicted in the Figure 13.5(a) on the facing page. For each level the trajectory reaches it is prevented in leaving which in some sense is a notion of stability. In practice there is many combinations of closing facets, but with the main idea in mind it seems intuitive that by closing facets that are facing away from the control goal are



ideally closed.



**FIGURE 13.5:** (a) Is showing what is the main idea. The circles denote a set that the trajectory can not get out of ones inside. (b) Is the practical implementation, facets whose normal vector is pointing away from equilibrium point are closed The cross denotes closed and circle is open.

Figure 13.5(b) introduce the notion open and closed facet using crosses and circles. Looking at polytope (1, 1), the facet facing to the right and the one facing upwards are open, as they are in the direction of the control goal which is the origo. The Figure is the graphical representation in two dimensions, but moving to higher dimensions makes it impossible to visualise. The automated procedure in  $d$ -dimension utilize the projection of the normal vector onto a vector pointing from origo to the centre of the polytope. Should this projection be negative, it indicate that the facet should be closed. In this manner all facets of the cube can be evaluated, leading to that  $d$  facet are left open and  $d$  facets are closed.

## 13.5 LINEARIZATION

As each polytope consist of a unique linear system it must be possible to linearize the the six dimensional system from Section 12.5 on page 118, in the centre of each polytope. The total number of polytopes is 256, this implies that a automated linearization process is needed.

Since the linearization of  ${}^1\dot{\omega}_1$  already has been automated using the Jacobian, see Section 11.2, only the linearization of  ${}^0_1\dot{q}$  is to be automated, due to the fact that this is linearized by hand. Equation (11.33) on page 88 is used for the automatizing. The skew matrix,  $\mathbf{S}(\bar{\omega}(t))$  multiplied which  ${}^0_1\tilde{q}_{1:3}$ . The skew matrix gives the ability to linearize  ${}^0_1\dot{q}$  in any polytope.

The affine part of the linearized system is given as the system bias just before evaluating the Jacobian when all the states are evaluated in their respective working point.

The linearization of the system in the polytopes is derived. The following section will concern the control space.

## 13.6 FORM THE CONTROL SPACE POLYTOPE

The task of this section is to represent all possible solutions to the control problem seen from one specific vertex. The inequalities of Proposition 12.1.1 on page 107 can with advantages be considered as halfspaces in the four dimensional control space. The basic description of a polytope in the  $\mathbf{H}$ -representation is written as  $Ax \leq b$  where rows of  $A$  contains the slope of one plane and the same row in  $b$  is a scalar that controls the shift of the plane in the control space. To make the inequalities from Proposition 12.1.1 on page 107 fit to the described halfspace notation the equations are rearranged like shown in Equation (13.7).

$$\begin{aligned} n_1^T(Av_j + Bu_j + a) &> 0 \\ -n_1^T(Av_j + Bu_j + a) &< 0 \\ -n_1^T Bu_j &< n_1^T(Av_j + a) \end{aligned} \quad (13.7)$$

Equation (13.7) evaluated on one facet will provide one half-space, and evaluating all  $n$  facets that have vertex  $v_j$  in it, with according to the closed and open facets, will provide  $n$  halfspaces in the control space. It is important to notice is that the initial condition of the  $U_j$  polytope already is a bounded cube, due to physical limitations in the motors. In a simple 2 dimensional case the control polytope be looking as depicted in Figure 13.6 on the next page.

On the left part of the figure only the saturation limits are placed, showing the obvious bounded starting point. Adding the extra constraints could result in the figure to the right. The set of control signals for  $v_j$  are then contained in  $\text{conv}\{U_j\}$ . The proposition

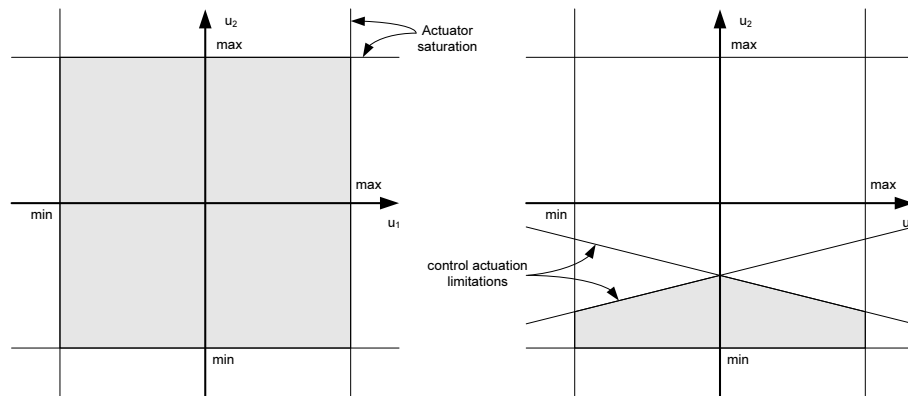


FIGURE 13.6: From the left: The initial bounded polytope  $U$  in the control space. Extra constraints add more halfspaces to the description of  $U$ , which limits the volume of the resulting polytope.

contain both  $<$  and  $\leq$  resulting in the problem that some facets are described by open half spaces, that is the facet is not contained in the solution, while others are. Should the exact polytope be formed, it would require that the inequality signs follow the half-spaces individually, like for this example of the unit square that does not contain the axis.

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} x \begin{bmatrix} \leq \\ < \\ \leq \\ < \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (13.8)$$

The forming of all the  $U$  polytopes, one for each vertex, has one case that needs special attention. When the halfspaces in the control space does not form a polytope. This topic is in some sense deeply rooted together with the forming of a control polytope, since to form the polytope states that control is possible otherwise the polytope can not be formed. But since this topic is where the *Combinatoric Controllability Method* differ from the method stated in [vSH04], the topic is handled separate in the following section.

## 13.7 BAD VERTEX IDENTIFICATION AND POLYTOPE TRIMMING

From the case study on the simple motor we defined the notion of bad vertex, which in short was a vertex that had no possible control signals that fulfilled the control goal. In

the frame work of the  $U_j$  polytope it is clear that when  $\text{conv}\{U_j\}$  is empty there is also no control solution. There are two reasons that can cause the  $\text{conv}\{U_j\}$  be empty. One being if the constraints in relation to the control problem, called  $H_c$ , then  $H_{c1} \cap H_{c2} \dots \cap H_{cN} \in \emptyset$ . In this case the vertex is stated as a proper bad vertex since this problem is related to that facet that is wanted closed. The other possibility is when a the constraints related to saturation limits for the actuators, called  $H_s$  have the characteristics of  $H_s \cap H_c \in \emptyset$  when  $H_{c1} \cap H_{c2} \dots \cap H_{cN} \neq \emptyset$ , should a vertex that fulfil this last statement be found, it will be a indication that the axis divisions from the beginning have been formed in a way that will be infeasible for control. Implementation wise it is possible to use a geometric software tool called polymake [GJ00]. The functionalities of this software are many, and they primarily lie in the field of polyhedral calculations. The downside to this is the lack of a interface to the program so that the functionalities can be used from Matlab, thus the interactions between Matlab and polymake for now been done manual. This needed human interaction has resulted in automatization of the full procedure have not been possible, which is needed in order to go head-on with a large dynamical system. The following will provide an idea on what is needed to carry out a trimming of the main structure.

Given that all vertices of the polytope  $P$  evaluates either as control vertices or proper bad vertices will provide the needed basis for a trimming of  $P$ . Further more it is assumed that any allocation/trimming of a vertex figure from the  $(0,1)$ -polytope will produce two convex polytopes. Lastly it is needed that at any time of the trimming procedure, the main structure will span  $\mathbb{R}^d$ . Should this last requirement fail is it an indication of that the main structure has collapsed and control has failed.

Evaluating the  $U$  polytope for all vertices with respect to feasibility will provide a set of bad vertex candidates. The chosen method of handling one bad vertex at a time give rise to the a notion of vertex classification which will decide which of the bad vertices should be handled first. This classification is proposed in the following manner:

**Algorithm 13.7.1**

1. Identify the vertex nearest the control goal and classify this as zero
2. All class one vertices are found by following all edges from the class zero vertex
3. From all these class one vertices one can arrive at the set of class two vertices by following uncovered edges.
4. The classification continues and is finished when the class set contains only one vertex, this will be in class  $d$ , when the cube is in  $\mathbb{R}^d$ .

Amongst the bad vertex candidates the one with the highest classification is chosen to be the basis for the first trim. By this method it is ensured that the main structure at all time is nearest the control goal.

Given that vertex  $v_j$  is a proper bad vertex with the highest classification, the method states a trim should be performed. As stated before will the trim mean that the vertex figure,  $P/v_j$ , is separated from the main structure. It is clear that by this procedure there will be formed a new facet over which the main structure and the fragment is isomorphic. To find this facet is no trivial problem, which is illustrated in two steps on the Figure 13.7.

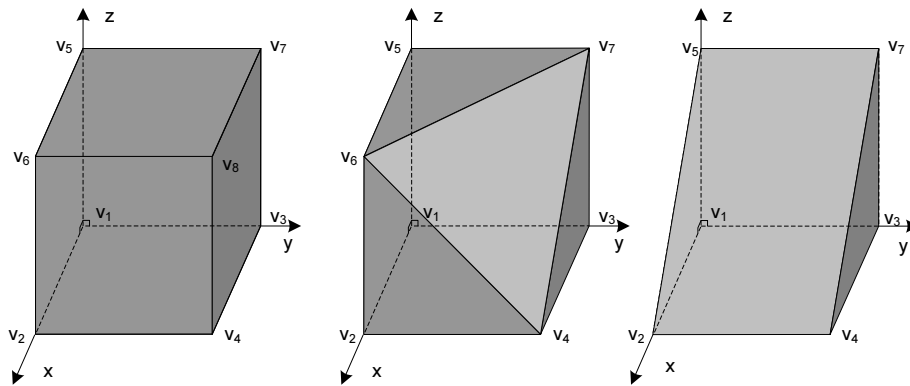


FIGURE 13.7: The resulting main structure after two vertices are trimmed.

From the figure it is seen that when trimming the first vertex is the vertex figure to  $v_8$  is a (0,1)-simplex but for the second trimming the vertex figure is forming (0,1)-polytope, thus nothing can be said about the vertex figure in general other than it will remain a (0,1)-polytope. Common for them both are that the new formed facet has the property that the supporting hyper plane divide the space in a way that the bad vertex is on one side and the remaining vertices not in the facet are on the other side. For the overall control method it is advantageous to represent facets using the  $V$ -representation, but here it is clear

that by moving to  $H$ -representation will provide the link for finding the hyper plane supported by a set of vertices that fulfil the before mentioned properties. The problem is now a task of transforming from the  $V$ - representation to the  $H$ -representation, which in the literature is stated as the *facet enumeration problem* [Fuk04]. Now the overall idea to find the newly formed facet, is to guess in a set of  $d$  vertices, and verify if the supporting halfspace fulfilled the stated properties. Having a 6-dimensional state space opens for a rather large amount of combinations, for which one can pull out  $d$  vertices of a set of  $2^d$  vertices. Roughly the number of combinations is calculated in Equation (13.9)

$$\frac{62!}{(62-6)! \cdot 6!} - 12 \cdot \frac{31!}{(31-6)! \cdot 6!} \approx 52.6 \text{ mill combinations} \quad (13.9)$$

In this calculation it is utilize that amongst the 64 vertices the bad vertex and the diagonal to this vertex should never be in the guessed set of vertices. Further more is all the solutions of facets that already is present subtracted. From this example is it clear that an efficient algorithm is needed, but knowing that there is a solution, it is chosen to not work on refining this algorithm.

Given that a facet is found that fulfil the stated properties the reverse transformation to  $V$ -representation is needed, known as the *vertex enumeration problem*. This will again form the notion on which vertices belong to the new facet. To complete the facet is to form the outwards normal vector to the facet in both main structure and fragment. The cross product will produce a vector that is orthogonal to the two vectors used in the calculations, fortunately the cross product generalizes in  $d$ -dimensions, with minor modifications. This can be used to form a normal vector for the newly formed facet. The general cross product in dimension  $d$  is stated in Equation (13.10).

$$\bigwedge(\mathbf{v}_1, \dots, \mathbf{v}_{d-1}) = \begin{vmatrix} v_1^1 & \dots & v_1^d \\ \vdots & \ddots & \vdots \\ v_{d-1}^1 & \dots & v_{d-1}^d \\ \mathbf{e}_1 & \dots & \mathbf{e}_d \end{vmatrix} \quad (13.10)$$

The structure of the calculation is recognized from the cross product in 3 dimensions expect that the basis vectors of the frame are placed in the bottom instead of the top of the determinant, this is done to ensure that the vector follows the right hand rule also in even dimensions. The result have been verified by taking the scalar product between the formed normal vector and any vector used in the determinant, which all yielded zero,

thus a normal vector can be formed in any dimension. Calculating the determinant of big matrices is feasible but becomes time consuming thus this method should only be used when geometric reasoning can not be applied.

Given that the two new structures are formed, choices have to be made on which facets to close or open. Due to the classification method proposed in Proposition 13.7.1 on page 131 the main structure will always be the closest to the control goal, thus any new facet formed seen from the main structure should be closed, complementary facet seen from the fragment is kept open. The other facets of the fragment copy the properties of closed and open from the main structure of which they were ones part of. The only facets that need special attention are the facets in the fragment that can not be closed, the solution is to open what can not be closed, still making a control that is the best possible, and let the neighbouring cube have pressure the trajectory more directly towards the control goal. Remember that the unsafe area was roughly define in the axis division, a fragmentation of a cube that lies up against an unsafe area will itself be defined as a unsafe area if the facet to this area can not be closed. In this manner the unsafe area is too automatically modified from the original rough estimate to a feasible solution, as is the case for the individual polytopes. Knowing that the procedure has many steps the method is stated in compact form in Proposition 13.7.2 which also mark the end of the trimming, and the task of control law generation can begun.

**Algorithm 13.7.2**

1. *Feasibility study - find all bad vertex candidates. If no bad vertex is found the polytope is fully trimmed , **break**.*
2. *choose bad vertex with highest classification.*
3. *Trim main with respect to chosen bad vertex.*
  - (a) *Find supporting half-space that fulfils trimming properties.*
  - (b) *Add common facet to both main structure and fragment.*
  - (c) *newly formed facet is wanted closed seen from the main structure.*
  - (d) *facets that could not be closed in fragment structure is opened.*
4. *Repeat the procedure, **go to** (1).*

## 13.8 CONTROL LAW GENERATION

This section intends to apply Equation (12.3) on page 107 to the trimmed polytope main structure and fragment in order to arrive at a sufficient control law. However as indicated in the previous section, the lack of an adequate algorithm for structure updating of the 6-cube, makes it impossible to generate a control law. Hence it has been chosen to consider a three dimensional cube in order to present a working control law, thereby maintaining the *Combinatoric Controllability method* sequence.

The considered state space system is constituted by the three angular velocity states  $(\omega_x, \omega_y, \omega_z)$ . The system is linearized in  $(3\pi/4, 3\pi/4, 3\pi/4)$  and has the following state space representation in (13.11).

$$\begin{aligned} \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} &= \begin{bmatrix} 0 & -2.056 & -2.056 \\ 2.056 & 0 & 2.056 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \\ &\begin{bmatrix} 0.015 & 0.013 & 0.015 & -0.042 \\ 0.013 & 0.015 & -0.042 & 0.015 \\ 0.004 & -0.004 & 0.004 & -0.004 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} + \begin{bmatrix} -4.844 \\ 4.844 \\ 0 \end{bmatrix} \end{aligned} \quad (13.11)$$

The affine system is defined on the 3-cube shown in Figure 13.8, with the vertices and normals as defined in 13.1.

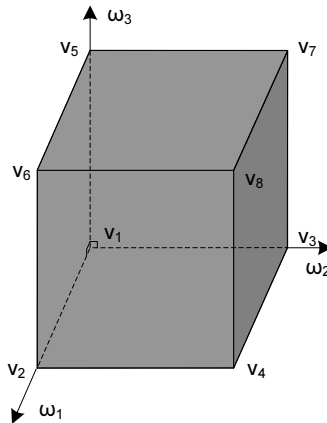


FIGURE 13.8: The polytope (3-cube) which the linear system is defined on.

The control objective is to drive the trajectory towards origo of the state space, hence the open facets are  $(F_1, F_2, F_3)$  and the closed facets  $(F_4, F_5, F_6)$ .



Vertex	Value	Normal	Value
$v_1$	$(\pi/2, \pi/2, \pi/2)^T$	$n_1$	$(0, 0, -1)^T$
$v_2$	$(\pi, \pi/2, \pi/2)^T$	$n_2$	$(-1, 0, 0)^T$
$v_3$	$(\pi/2, \pi, \pi/2)^T$	$n_3$	$(0, -1, 0)^T$
$v_4$	$(\pi, \pi, \pi/2)^T$	$n_4$	$(0, 0, 1)^T$
$v_5$	$(\pi/2, \pi/2, \pi)^T$	$n_5$	$(0, 1, 0)^T$
$v_6$	$(\pi, \pi/2, \pi)^T$	$n_6$	$(1, 0, 0)^T$
$v_7$	$(\pi/2, \pi, \pi)^T$		
$v_8$	$(\pi, \pi, \pi)^T$		

**TABLE 13.1:** *The vertices and normal vectors to the 3-cube. The facets enumeration of the polytope relate directly to the subscript of the normal vectors.*

In order to arrive at a control law one must first ensure that control signals can be calculated at the vertices of the polytope. As calculation of control signals according to the form presented in Section 13.6 on page 128 will occupy a great deal of space, it has been chosen to present an example of the control signal calculation for vertex  $v_8$  in the following.

As  $v_8$  is the only vertex contained in the closed set the inequalities are derived according to Proposition (12.1.1)(2). From a practical perspective it will prove sufficient to calculate for Proposition (12.1.1)(2a) and with the harder  $<$  instead of  $\leq$ . This assessment is made from a performance point of view, with the tradeoff in the sense that the trajectory is not allowed to evolve directly on the boundary of the facets ( $F_4, F_5, F_6$ ). As the goal of the control is to direct the trajectory towards the facets this assessment is considered acceptable. The inequalities for the calculation of the control space polytope  $U_8$ , are expressed in (13.12).

$$\begin{aligned}
 & n_i^T \left( \begin{bmatrix} 0 & -2.056 & -2.056 \\ 2.056 & 0 & 2.056 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \pi \\ \pi \\ \pi \end{bmatrix} + \right. \\
 & \left. \begin{bmatrix} 0.015 & 0.013 & 0.015 & -0.042 \\ 0.013 & 0.015 & -0.042 & 0.015 \\ 0.004 & -0.004 & 0.004 & -0.004 \end{bmatrix} \begin{bmatrix} u_{8F} \\ u_{8L} \\ u_{8B} \\ u_{8R} \end{bmatrix} + \begin{bmatrix} -4.844 \\ 4.844 \\ 0 \end{bmatrix} \right) < 0 \tag{13.12}
 \end{aligned}$$

In order to form the above stated inequalities into halfspace notation, (13.12) is rearranged according to (13.13)

$\forall i \in \{4, 5, 6\} :$

$$n_i^T \begin{bmatrix} 0.015 & 0.013 & 0.015 & -0.042 \\ 0.013 & 0.015 & -0.042 & 0.015 \\ 0.004 & -0.004 & 0.004 & -0.004 \end{bmatrix} \begin{bmatrix} u_{8F} \\ u_{8L} \\ u_{8B} \\ u_{8R} \end{bmatrix} < -n_i^T \left( \begin{bmatrix} 0 & -2.056 & -2.056 \\ 2.056 & 0 & 2.056 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \pi \\ \pi \\ \pi \end{bmatrix} + \begin{bmatrix} -4.844 \\ 4.844 \\ 0 \end{bmatrix} \right) \quad (13.13)$$

With the equation stated above the  $U_8$  control polytope is defined. Within this polytope, control signals for the four motors must be chosen. The smallest control signal set still residing in  $U_8$  is  $[-435 \ 435 \ 435 \ 0]$ . The reason for the large size of the control signals are due to the lower and upper limits of the actuators. These are set to  $-3000$  and  $3000$  respectively in order to arrive at a feasible control law. If more realistically control signals are wanted the axis divisions must be revisited.

With the control signals for vertices  $v_1, \dots, v_7$  calculated in the same way, we arrive at the following.

$$\begin{bmatrix} \pi/2 & \pi/2 & \pi/2 & 1 \\ \pi & \pi/2 & \pi/2 & 1 \\ \pi/2 & \pi & \pi/2 & 1 \\ \pi & \pi & \pi/2 & 1 \\ \pi/2 & \pi/2 & \pi & 1 \\ \pi & \pi/2 & \pi & 1 \\ \pi/2 & \pi & \pi & 1 \\ \pi & \pi & \pi & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ g \end{bmatrix} = \begin{bmatrix} -277 & 277 & 277 & 0 \\ -859 & 859 & 859 & 0 \\ -1 & 0 & 0 & 0 \\ -356 & 356 & 0 & -356 \\ -356 & 356 & 0 & -356 \\ -1 & 0 & 0 & 0 \\ -1105 & 1105 & 0 & -1105 \\ -435 & 435 & 435 & 0 \end{bmatrix} \quad (13.14)$$

Taking the pseudo inverse (`pinv` in MATLAB<sup>®</sup>) of the vertex matrix provides the affine control law  $u = Fx + g$  stated in (13.15)

$$u = \begin{bmatrix} 13.875 & -64.230 & -64.230 \\ -13.875 & 64.230 & 64.230 \\ 161.907 & -111.552 & -111.552 \\ 175.782 & -175.782 & -175.782 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \begin{bmatrix} -153.825 \\ 153.575 \\ 340.637 \\ 187.062 \end{bmatrix} \quad (13.15)$$

In the following section the control law is simulated on the 3-cube.

### 13.8.1 SIMULATION OF CONTROL LAW ON 3-CUBE

The results of the simulation of the closed loop system stated in 13.16.

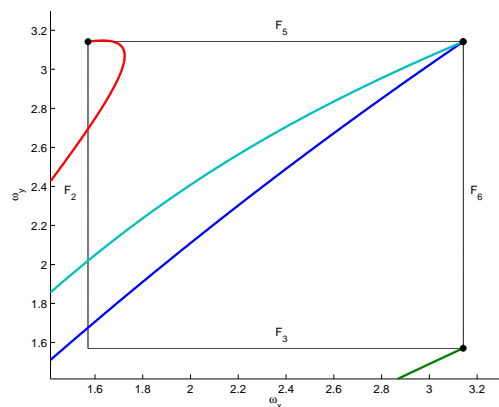
$$\dot{x} = (A + BF)x + (Bg + a) \quad (13.16)$$

are shown in Figure 13.9, which is constituted by three two dimensional subplots corresponding to the  $\omega_x\omega_y$ ,  $\omega_x\omega_z$  and  $\omega_y\omega_z$  axes. The system is started in the vertices  $v_4, v_6, v_7, v_8$  and the trajectories are plotted. For all the plots applies that the trajectories must leave either the left or bottom facet corresponding to the facets in the open set  $F_1, F_2$  and  $F_3$ .

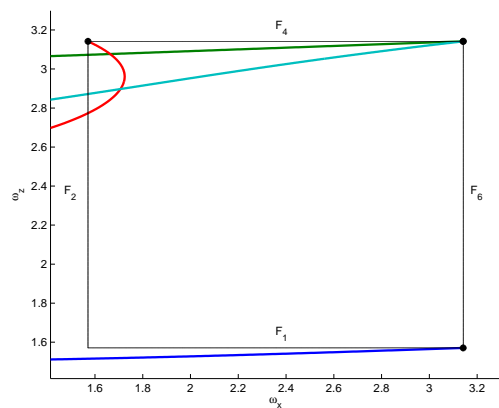
As can be seen from the plots all trajectories are driven correctly towards the facets except two, which are contained in subfigures 13.9(a) and 13.9(c). The trajectories that arise from points  $(\pi/2, \pi)$  and  $(\pi, \pi)$  in Figure 13.9(a) and 13.9(c) respectively, slightly exits the cube before they enter again converging towards the correct facets. It turns out that these points constitute  $v_7$  and that the control law fails to close facet  $F_5$ . The latter results calls for a recapitalization.

When taking the pseudo inverse of the vertex matrix this indeed provides a least squares solution with respect to the choice of input signals in the concerned  $U$  space. However it seems as the choice of specific input signals, as apparent in the simplicial case [vSH04, p.27], cannot freely be made in the  $U$  space for systems defined on other polytopes than simplices, in this case the 3-cube.

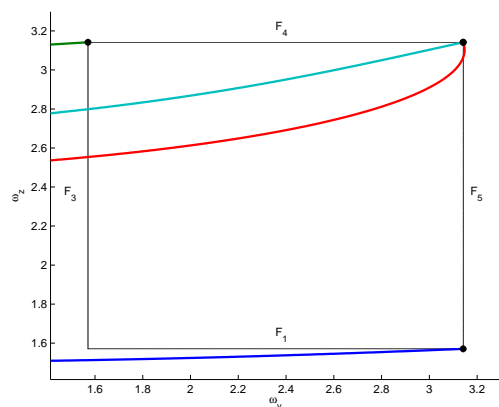
Various combinations of inputs in  $U$ , however by far all, have been tested and no control law derived has as good a performance as the one derived in (13.15). As a probable control law that completely maintains the control demands on the 3-cube could be found, this strongly indicates that it is no longer up to the freedom of choice when choosing control signals.



(a)  $\omega_x$  and  $\omega_y$  axes.



(b)  $\omega_x$  and  $\omega_z$  axes.



(c)  $\omega_y$  and  $\omega_z$  axes.

FIGURE 13.9: Verification of the control law applied to 3-cube system.



# PAHS CLOSURE

In the following a short review of the method is stated and the surrounding framework is sketched to provide the reader with a understanding on what is possible, and not at least still poses a challenge. Standardized method have been proposed for state space parting, automatic trajectory guiding and automatic trimming. One of the forces of this method is that they are building on the knowledge of the affine systems that are to fill each discrete location in the state space. The automatic trimming will, when succesfull, result in a complete set of control signals in each vertex that will fulfill the stated control problem.

To convert the sets of control signals into a control law, has been proposed in two ways. One being that a clever choice of control signals will provide a control law for the full polytope, the other possibility is to perform a triangulation on the polytope and calculate a unique control law for each simplex, and the large amount of controllers fill the polytope. Both approaches provide the possibility to handle each cube as a decoupled control system not influenced by the neighbouring cubes. All that is needed is that each cube is based on the same clearly defined control goal. With the proposals the automatically generated controller is taken a step closer.

This part has mainly concerned the forming of the control synthesis, but for the full implemented controller some parts need further work. One big but not very difficult task is to fit the control laws, linear systems, reset map and transition function into the PAHS tuple. It is proposed that the change between cubes will be represented as a external location transition, since this event should follow by a change of linear system. Internal transitions, that is inside the cube, does not change the linear system but will only change the control law. Due to these different levels of transitions it is proposed that the hybrid tuple should handle these two levels. In order to generate location transition events a system is needed that has knowledge of all the discrete locations inside the state space. It seems that the  $H$ -representation of the bounded polytopes will be the best choice of representation due to the simple procedure of testing if a point is in a given cube, and under this cube there is a relative small number of polytopes which the cube is consisting of. If the discrete location is known at one time instance it will only be necessary to let the location observer test for the polytopes that belong to the neighbour cube set to the present location, which minimizes the needed number of on line calculations.

In the verification of the control law it is shown that when constructing the control on

---

the basis of the overdetermined vertex matrix not all control laws formed does actually fulfil the control goal, further studies on this subject should be carried out before it can be ruled out that direct control on polytopes without triangulation is possible. Should it turn out that triangulation is needed, will actually mean that any triangulation method can be used, and the choice should be guided by the effectiveness of the method. The method called *delaunay triangulation* performs reasonable and already has a Matlab implementation, *delaunayn*. Even if the most efficient algorithm was implemented the number of triangulations on the 6-cube can never come under 324 simplices per cube. Which is one of the reasons why it is interesting not to perform the triangulations if it can be avoided, since the number of controllers for the six dimensional case described will at its minimum be  $(4 \cdot 4 \cdot 8 \cdot 2 \cdot 2 \cdot 2) \cdot 324 = 331776$  control laws. One of the more practical issues is the algorithm for finding the common facet between one fragment and the main structure. One calculation heavy method have been proposed, but it is believed that a more efficient method exist, which is strongly backed up by geometric computational tools as polymake [GJ00] which is developed by the Discrete Geometry group at the Institute of Mathematics at Technische Universität in Berlin. It is strongly believed that there exist large knowledge banks on this topic, but since the field of this work reaches into another specialist field, the pieces that form the solution has posed a timed consuming problem and has thus not been finished.

# **PART V**

# **CLOSURE**

*This part contains two chapters which serves as thesis closure. The first chapter 15 concludes on each part of the thesis and eventually on the goals displayed in section 1.2 on page 3. The second chapter 16 deals with the future perspectives of the project.*







# CONCLUSION

The overall goals for this master's thesis has been to render a quad rotor helicopter capable of autonomous hover flight, and to apply theories from Piecewise Affine Hybrid Systems to the quad rotor system, with the aim of deriving an adequate control law within that framework. The objectives related to these goals, stated in section 1.2 on page 3, will be concluded upon in the following, starting objectives towards achieving autonomous hover flight.

## AUTONOMOUS HOVER FLIGHT

A whole new electronic platform has been built onto the mechanical helicopter structure. The platform is fully equipped with sensors, computing power and two wireless connections. One R/C connection dedicated for manual flight control and one WiFi broadband connection for data logging. With a battery installed on the helicopter it proves to be a fully wireless application. The main part of the sensor sampling is handled by a slave processor which has been designed using timed Finite State Automata. The using of FSA has proven to be a power full design tool, that clearly handles fault detection and fault handling, which has ensured that the data sampling is done reliably by the slave processor. Effort has been put into assuring safety before, during and after flight, with adding of status indicators, arming switch and a manual control feature. An angular dampening controller has been implemented, aiding a human pilot during manual flight. Weight has been of the essence, thus adding the new hardware has only raised the weight from 2400 g to 2600 g. Seen from the main CPU (Gumstix), the range finder, robstix and GPS drivers have been implemented as stand alone applications in C. The GPS is corrected by DGPS information relayed via a broadband Ethernet connection. At a later stage the real time workshop environment for MATLAB<sup>®</sup>/Simulink<sup>®</sup> was implemented providing a fast intuitive prototyping of estimator and controller. The platform is fully functional, and only the GPS module awaits implementation in the soft realtime target environment, before implementing estimator and controller. This concludes and verifies the first platform objective towards the goal of autonomous hover flight.

The hover controller was built from the basis of a common interface towards the group [GHB07], working on the estimation of state variables. The state estimator was finished at a late stage in the project period, thus making implementation of the controller im-

---

possible. The hover controller could as a result of this only be verified on the non-linear model, with appropriate disturbances emulated.

The chosen controller design consists of cascade coupled controllers, with PI controllers on each rotor system and an overall LQ controller, handling the stabilizing of the quad rotor helicopter in the air. The controller design is a model based approach, thus the model developed in [KSG<sup>+</sup>06], has been used with few extensions. The representation of the attitude was converted to the use of quaternions and the motor parameters were re-evaluated as four different motor-gear systems, as the performance of these was not alike. The developed rotor speed controllers provide the tunable parameter that is used to gain alike performance from the four rotor systems. Simulations of the rotor systems with the rotor speed controllers show similar step responses, when operating inside the saturation limits of the motor voltage, and is accepted as a reduction of the overall complexity of the helicopter. An overall LQ state feedback controller has been implemented on the non-linear model. Integral states has been added to all three position states and on the third element of the quaternion, which effectively removes steady state error in position and heading. Verification of the full controller is carried out by applying steps in the position reference in all directions. The simulation is carried out with added delays on the state estimate by 20 ms, and delay on the control signals to the model also by 20 ms. The limits set in the requirement specification are met except with respect to the translational velocity. This is however due the magnitude of the step input, which normally would not occur for the position reference. The developed controller is on behalf of this considered tolerant to disturbances, thereby appropriate to implement.

## PIECEWISE AFFINE HYBRID SYSTEMS CONTROLLER

The second objective of this thesis has been to investigate the usage of newly stated work done on Piecewise Affine Hybrid Systems, on the dynamical quad rotor system at hand. For a feasible system, it has been necessary to reduce the number of states in the description of the quad rotor helicopter. The reduced system has six states, which represents attitude and angular velocity. With this reduction, the system still is complex and has many of the properties that can be found in many dynamical systems. The basic idea behind the method, is a mathematical proof of that it is possible to give a statement on the continuous trajectory, formed by a linear system, starting anywhere on a given facet, by investigating the trajectory in the vertices of that facet. By projection onto the normal vector of the facet it is possible to form statements on the interval for the control signal

and thus the trajectory can only move one way through this facet. With the notion of closed and open facets, the idea is to observe the state space as a  $d$ -dimensional space, which has to be filled with convex polytopes, each of which are isomorphic to their corresponding neighbouring polytope. By closing the right facets it is possible to have a piecewise approach for making control on a strongly non-linear system using a set of linear approximations made from the non-linear model.

Additions have been made to the method without jeopardizing the general usage of the method. A method for parting the state space into a set of isomorphic cubes have been presented. The practical handling of the cube and the information related to it, has been represented as class known from object oriented programming, making the interfaces between steps of the algorithm as clean as possible. It has been ensured that each cube is a small linear control problem, completely decoupled from the neighbouring cubes. Contributing to the automation a method for deciding which facets to close has been proposed based purely on the location of the cube in relation to the control goal. Furthermore a theoretical solution to identify bad vertices, that arise due to dynamical couplings, where control signals only, has an influence through the dynamics of the system, has been stated. Should a bad vertex be identified in a cube, a theoretical algorithm is presented for further dividing the cube into smaller polytopes, thus ensuring an automated progress in the task of finding a control law using the chosen cubical grid. A representation of the control signals that fulfil the local control problem has been implemented, using a geometric abstraction into halfspaces. At this point in the algorithm it is argued that there is two different possible approaches for forming a control law on the local control problem at hand. Either can it be chosen to triangulate the polytope at hand and form individual but unique control laws for each simplex. The triangulation will result in an explosion of discrete locations, which seems impractical with respect to the implementation. On the other side it seems possible to chose among the possible control signals and form a global control law for the polytope, even though the derived control law performs purely on a overdetermined vertex matrix. The problem is clarified by the forming of the least squares solution to the control problem on the polytope. These phase portrait clearly indicate the correct tendency, which is taken as an indication that the performed steps to arrive at the control law has be done in a correct manner. Nevertheless, the fact that the control law can not close the facet that is wanted closed, thus more research should be made into choosing control signal for the full polytope that can be realize by a linear control law.

The greatest challenge in PAHS resides in keeping the amount of data storage low

---

and to keep the number of on-line calculations to a minimum. The parameters that can be tuned are concerning the number of continuous states, the number of axis divisions and the choice of geometric figure to form the final discrete location. Knowing that the minimum number of simplices contained in the 5-cube is 67, but when considering of the 6-cube the number explodes to 324 [GO97]. By working on cubes and only splitting up the cube when the control goal can not be fulfilled, will reduce the number of discrete locations considerable compared to the triangulated solution.

With the contributions made to the algorithm, the work of fitting any dynamical system into the PAHS framework has been made more easy and another step is taken in achieving automatic and computerised controller generation.



# PERSPECTIVES

There are two sides to the perspectives, one is concerning the developed platform and the other being the further work needed on the PAHS area to form a basic implementation that will provide further clues of bottle necks.

The physical construction has been equipped with a reasonable set of sensors and the on-board computer is considered sufficient for further expansions. The bottleneck in the construction as it is now are related to the mechanical construction, or more precisely the rotor blades and the motors. The present motors are brushed motor which since the purchase have been pressured over the peak performance and since the weight has been further increased by the fitting of sensors, the motors are believed to have reached their performance limit, since the manoeuvrability are deeply correlated with the amount of extra lift that can be applied by the motors from hover and to maximum revolutions. It is proposed that the motors are replaced with four new brush less motor. These types of motors has become very popular for electrical helicopters. The greatest advantages of using these are first of all a higher efficiency and less electromagnetic noise is produced which might make it possible to remove the large ferrite beds, placed to eliminate noise inflicted on data lines. This will also remove weight and provide more manoeuvrability. Should a more powerful brush less motor be purchased it should be possible to increase the lift by adding a extra rotor blade as the new Draganflyer X-Pro now is equipped with three blades instead of 2. It has been noticed that the wireless connection has a relative short range, and that the usage of the full 54Mbit far from is used or needed in the present application. Should the next goal be some short of trajectory planning or object avoidance, is it recommended that wireless connection is replaced by a technology with a longer range, e.g. a class 1. bluethooth unit.

The second objective of this thesis has been looking into a new area denoted PAHS which is a area with much potential, but since the area is relative new the tools that can aid the work are not gathered in one program packet, thus the task of implementing the theories of PAHS often involves implementing functionalities from the ground, or review programs to find the needed building blocks. The developed method denoted *combinatoric controllability method* has pointed towards algorithms that e.g. determine if a given set of halfspaces form a hull. Other topics are simply a matter of coming up with efficient algorithm e.g. find the facet formed when a vertex figure is removed from the main structure or calculating the d-dimensional cross product. These topic area surely handled

---

by others in different fields of work, thus the needed knowledge should be adopted to fit into the stated control problem. Amongst tools and literature [GO97], [Fuk07] and [GJ00].

# BIBLIOGRAPHY

- [And07] Palle Andersen. *Optimal Control - 2nd February 2007*. Department of Process Control, Aalborg University, Viewed 26-04-2007. URL <http://www.control.aau.dk/~pa/kurser/Optimal/optnote.pdf>.
- [AP05] Palle Andersen and Tom S. Pedersen. *Modeldannelse - 9. marts 2005*. Department of Process Control, Aalborg University, 2005.
- [Atm] Atmel. *8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash*. URL <http://www.atmel.com/atmel/acrobat/doc2467.pdf>. Viewed 01-11-2006.
- [AVR] AVR Freaks. *Design Note #007, Little and Big Endian*. URL [http://www.avrfreaks.net/Tools/ToolFiles/239/DN\\_007.pdf](http://www.avrfreaks.net/Tools/ToolFiles/239/DN_007.pdf). Viewed 01-11-2006.
- [BBG<sup>+</sup>06] Jakob Bjørn, Ole Binderup, Sigurgeir Gislason, Jesper Haukrogh, Morten Kjærgaard, and Martin Sørensen. *Worksheets on Draganflyer X-Pro*. Technical report, Aalborg University, 2006. On the CD-ROM: [/pdf/worksheets.pdf](#).
- [Bha] Dan D. V. Bhanderi. *Linux Soft Real-Time Target v2.3*. Department of Process Control, Aalborg University. URL <http://www.control.aau.dk/~danji/downloads/>. Viewed 13-03-2007.
- [Bha05] Dan D. V. Bhanderi. *Spacecraft Attitude Determination with Earth Albedo Corrected Sun Sensor Measurements*. Aalborg university, 2005. ISBN 87-90664-26-4.
- [Cra05] John J. Craig. *Introduction to Robotics Mechanics and Control, 3th. ed.* Prentice Hall, 2005. ISBN 0-13-123629-6.
- [FENP02] Gene F. Franklin, Abbas Emami-Naeini, and J. David Powell. *Feedback Control of Dynamic Systems, 4th. ed.* Prentice Hall, 2002. ISBN 0-13-032393-4.
- [FPW98] Gene F. Franklin, J. David Powell, and Michael Workman. *Digital Control of Dynamic Systems, 3rd. ed.* Addison Wesley Longman, Inc., 1998. ISBN 0-201-33153-5.

- [Fuk04] Komei Fukuda. *Frequently asked questions in polyhedral computation*. Technical report, Swiss federal institute of technology, june 2004. URL <ftp://ftp.ifor.math.ethz.ch/pub/fukuda/reports/polyfaq040618.pdf>.
- [Fuk07] Komei Fukuda. *cdd and cddplus Homepage*. Technical report, Swiss federal institute of technology, February 2007. URL [http://www.ifor.math.ethz.ch/~fukuda/cdd\\_home/cdd.html](http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html).
- [GHB07] Sigurgeir Gislason, Jesper Haukrogh, and Ole Binderup. *Platform development, Sensor modelling and Estimation of a Draganflyer X-Pro*. Technical report, Aalborg University, 2007.
- [GJ00] Ewgenij Gawrilow and Michael Joswig. polymake: a framework for analyzing convex polytopes. In Gil Kalai and Günter M. Ziegler, editors, *Polytopes — Combinatorics and Computation*, pages 43–74. Birkhäuser, 2000.
- [GO97] Jacob E. Goodman and Joseph O’Rourke. *Handbook of discrete and computational geometry*. CRC Press, 1997. ISBN 0-8493-8524-5.
- [Guma] Gumstix, Inc. *Buildroot - GumstixDocsWiki*. URL <http://docwiki.gumstix.org/Buildroot>. Viewed 21-05-2007.
- [Gumb] Gumstix, Inc. *gumstix - way small computing*. URL <http://www.gumstix.com>. Viewed 13-05-2007.
- [i2c00] *The I<sup>2</sup>C-Bus Specification*, January 2000. URL [http://www.semiconductors.philips.com/acrobat\\_download/literature/9398/39340011.pdf](http://www.semiconductors.philips.com/acrobat_download/literature/9398/39340011.pdf). Viewed 10-11-2006.
- [KSG<sup>+</sup>06] Morten Kjærgaard, Martin Sørensen, Sigurgeir Gislason, Jesper Haukrogh, Ole Binderup, and Mikael Berg Andersen. *Draganflyer X-Pro Modelling and Control*. Technical report, Aalborg University, 2006. On the CD-ROM: [/pdf/06gr831\\_report.pdf](#).
- [Mat07] Mathworks. *Homepage for MATLAB<sup>®</sup>*. Technical report, Viewed 16-04-2007. URL <http://www.mathworks.com/access/helpdesk/help/toolbox/control/ref/ctrb.html>.
- [PNI06] PNI Corp. *PNI MicroMag 3 - 3-Axis Magnetic Sensor Module*, June 2006. URL [https://www.pnicorp.com/downloadResource/c40c/manuals/110/MicroMag3+3-Axis+Sensor+Module\\_June+2006.pdf](https://www.pnicorp.com/downloadResource/c40c/manuals/110/MicroMag3+3-Axis+Sensor+Module_June+2006.pdf). Viewed 25-10-2006.



- [Pro89] Raymond W. Prouty. *Helicopter Performance, Stability, and Control*. PWS Publishers, 1989. ISBN 1-57524-209-5.
- [SB00] Raymond A. Serway and Robert J. Beichner. *Physics - For Scientists and Engineers with modern physics*. Harcourt College Publishers, 2000. ISBN 0-03-022657-0.
- [Son81] E.D. Sontag. *Nonlinear regulation: The piecewise-linear approach*. Technical report, 1981. Published on IEEE Transactions on Automatic Control, 26, 346-358.
- [TM04] A. Tayebi and S. McGilvray. *Attitude Stabilization of a Four-rotor Aerial Robot*. Technical report, Department of Electrical Engineering, Lakehead University, December 2004. Published at 43rd IEEE Conference on Decision and Control.
- [Tsk07] K. Tskalis. *Popov-Belevich-Hautus test*. Technical report, Viewed 16-04-2007. URL <http://www.eas.asu.edu/~tsakalis/notes/sco.pdf>.
- [UB90] J. Zweig (UIUC) and C. Partridge (BBN). *TCP Alternate Checksum Options*. Technical report, Network Working Group, March 1990.
- [Ubl] Ublox. *SAM-LS, GPS Smart antenna module data sheet extended*. Technical report. URL [http://www.u-blox.com/customersupport/gps.g3/ANTARIS\\_Protocol\\_Specification\(GPS.G3-X-03002\).chm](http://www.u-blox.com/customersupport/gps.g3/ANTARIS_Protocol_Specification(GPS.G3-X-03002).chm). Viewed 12-11-2006.
- [vSH04] J.H van Schuppen and L.C.G.J.M. Habets. *A control problem for affine dynamical systems on a full dimensional polytope*. Technical report, Centrum voor Wiskunde en Informatica, 2004. Automatica 40, 21-35.
- [vSHC06] J.H van Schuppen, L.C.G.J.M. Habets, and P.J. Collins. *Reachability and Control synthesis for Piecewise-Affine Hybrid Systems on Simplices*. Technical report, Centrum voor Wiskunde en Informatica, June 2006. IEEE Transactions on Automatic Control, Vol 51, No.6.
- [Wer94] James R. Wertz. *Spacecraft Attitude Determination and Control*. Kluwer Academic Publishers, 1994. ISBN 90-277-1204-2.
- [Wiś00] Rafał Wiśniewski. *Lecture notes on modeling of a spacecraft*. Technical report, Aalborg Universitet, Afdeling for Proceskontrol, March 2000. URL <http://www.control.auc.dk/~raf/ModellingOfMech/Rep8sem.ps>. Viewed 12-03-2007.

[Wola] Wolfram research. *Euler Parameters*. URL <http://mathworld.wolfram.com/EulerParameters.html>. Viewed 07-03-2007.

[Wolb] Wolfram research. *Quaternion*. URL <http://mathworld.wolfram.com/Quaternion.html>. Viewed 07-03-2007.

# ABBREVIATIONS

<b>AAU</b> Aalborg University	<b>OS</b> Operating System
<b>API</b> Application Program Interface	<b>PAHS</b> Piecewise Affine Hybrid Systems
<b>CEP</b> Circular Error Probability	<b>PCB</b> Printed Circuit Board
<b>CM</b> Center of Mass	<b>PBH</b> Popov-Belevich-Hautus
<b>CPU</b> Central Processing Unit	<b>PI</b> Proportional Integral
<b>DCM</b> Direct Cosine Matrix	<b>PWM</b> Pulse Width Modulation
<b>DES</b> Discrete Event System	<b>R/C</b> Remote Control
<b>DGPS</b> Differential Global Positioning System	<b>RTCM</b> Radio Technical Commission for Maritime services
<b>DHM</b> Development Host Machine	<b>SBAS</b> Satellite Based Augmentation System
<b>FSA</b> Finite State Automata	<b>SDRAM</b> Synchronous Dynamic Random Access Memory
<b>G2R</b> Gumstix-to-Robostix connection	<b>SEP</b> Spherical Error Probability
<b>GPIO</b> General Purpose Input/Output	<b>SPI</b> Serial Peripheral Interface
<b>GPS</b> Global Positioning System	<b>UART</b> Universal Asynchronous Receiver Transmitter
<b>HS</b> Hybrid Systems	<b>UAV</b> Unmanned Aerial Vehicle
<b>I/O</b> Input/Output	<b>UBX</b> U-BloX
<b>I<sup>2</sup>C</b> Inter-Integrated Circuit	<b>X-Pro</b> Draganflyer X-Pro
<b>IAS</b> Intelligent Autonomous Systems	
<b>IMU</b> Inertial Measurement Unit	
<b>ISR</b> Interrupt Service Routine	
<b>LED</b> Light Emitting Diode	
<b>LNX</b> Linux Soft Real Time Target	
<b>LQ</b> Linear Quadratic	
<b>MIMO</b> Multiple Input Multiple Output	
<b>MOSFET</b> Metal Oxide Semiconductor Field-effect Transistor	
<b>ODE</b> Ordinary Differential Equations	



# **PART VI**

# **APPENDICES**

*This part contains the appendices for this master's thesis. They individually elaborate on various subjects in chapters throughout the main thesis.*

# CONTENTS OF PART VI

---

<b>A</b>	<b>Body model structure verification</b>	<b>159</b>
A.1	Test scenarios . . . . .	159
<b>B</b>	<b>Quaternions</b>	<b>167</b>
B.1	Introduction . . . . .	167
B.2	Quaternion multiplication . . . . .	168
B.3	Rotation of quaternions . . . . .	168
B.4	Time derivative of a quaternion . . . . .	169
<b>C</b>	<b>Robostix software</b>	<b>171</b>
C.1	Assignment of timers . . . . .	171
C.1.1	Tasks . . . . .	171
C.1.2	Priorities . . . . .	172
C.1.3	Available timers . . . . .	173
C.1.4	Allocation of timers . . . . .	173
C.1.5	Timer/Counter1 + Timer/Counter3 . . . . .	173
C.1.6	Timer/Counter0 . . . . .	174
C.1.7	Timer/Counter2 . . . . .	175
C.2	Motor driver . . . . .	175
C.2.1	Interface . . . . .	176
C.3	Magnetometer driver . . . . .	176
C.3.1	Overall design . . . . .	176
C.3.2	Interface . . . . .	179
C.4	Tachometer driver . . . . .	179
C.4.1	Overall design . . . . .	180
C.4.2	Interface . . . . .	180
C.5	IMU driver . . . . .	182
C.5.1	Overall design . . . . .	182
C.5.2	Interface . . . . .	183
C.6	Robostix/Gumstix interface . . . . .	184
C.6.1	Packet design . . . . .	186

<b>D Acceptance test specification</b>	<b>191</b>
D.1 Verification of sensor data: . . . . .	191
D.1.1 robostix packet . . . . .	191
D.1.2 Range finder . . . . .	192
D.1.3 GPS module . . . . .	195

---







# BODY MODEL STRUCTURE VERIFICATION

*This appendix contains a description of the verification of the body model structure, which has been developed for the X-Pro. It is desired to conduct an isolated test of the body model, but it has not been possible to find a method for testing the actual values used in the model. Therefore the verification will be based on a structural test, investigating the overall correctness of the body model and the signs used in it.*

## A.1 TEST SCENARIOS

The verification of the body model takes its starting point in the functionality that the pilot of the X-Pro has when maneuvering the X-Pro in the air with the remote control as it is these functionalities the model should have as well. The inputs that the pilot can control from the remote control are *Throttle*, *Pitch*, *Roll* and *Yaw*.

Each of the four controller options are to be tested in an individual test case that contains a description of how the test is carried out along with a description of what the expected test result is and the actual test result. In the test of the model the position vector of the X-Pro  ${}^0P_1$  and the rotation matrix  ${}^0R_1$ , see Equation (A.1), relative to the universal coordinate system are computed and afterwards plotted to illustrate the movement of the X-Pro.

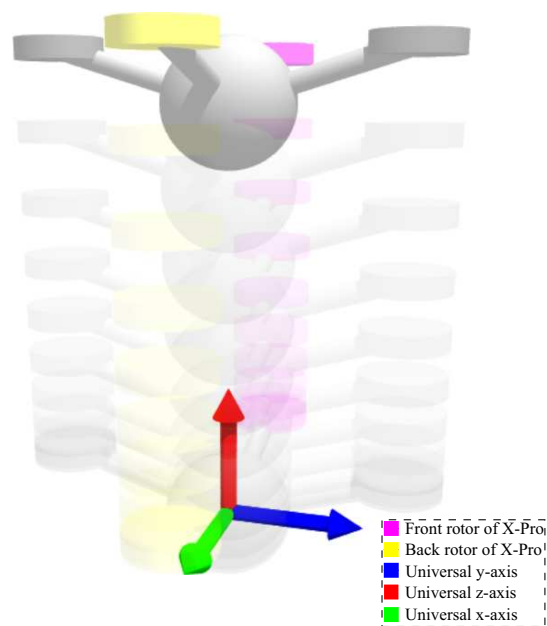
$${}^0R_1 = \begin{bmatrix} {}^0\hat{X}_1 & {}^0\hat{Y}_1 & {}^0\hat{Z}_1 \end{bmatrix} \quad (\text{A.1})$$

The position and the orientation of the X-Pro are saved and plotted for 10 different times which illustrates the motion of the X-Pro with the given test case input.

### TEST CASE 1: THROTTLE

**Description:** The X-Pro is oriented horizontally with the pitch, roll and yaw controls set to zero. The sum of forces from the rotors  ${}^1f_{aero,j}$  is set to values above and beneath the force of gravity affecting the X-Pro.

**Expected result:** For throttle values beneath the force of gravity should result in an acceleration straight downwards and values above the gravity a straight upwards acceleration is expected, which can be seen in Figure A.1 on the next page.



**FIGURE A.1:** Illustration of how the X-Pro is expected to move when giving the model a throttle input higher than the force exerted by gravity. This is expected to lead to a straight upwards acceleration.

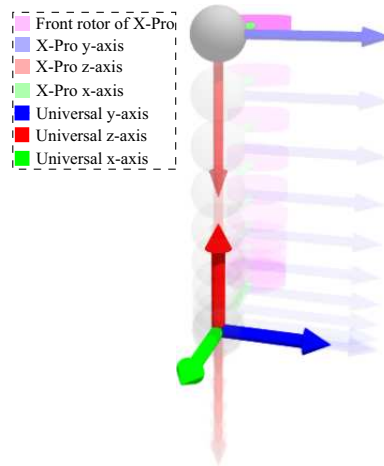
**Test result:** Both a positive and negative throttle input has been tested but only the test with a positive input are illustrated, see Figure A.2 on the facing page. The figure shown that with a positive throttle input the model of the X-Pro performs a straight upwards acceleration with no rotation.

## TEST CASE 2: PITCH

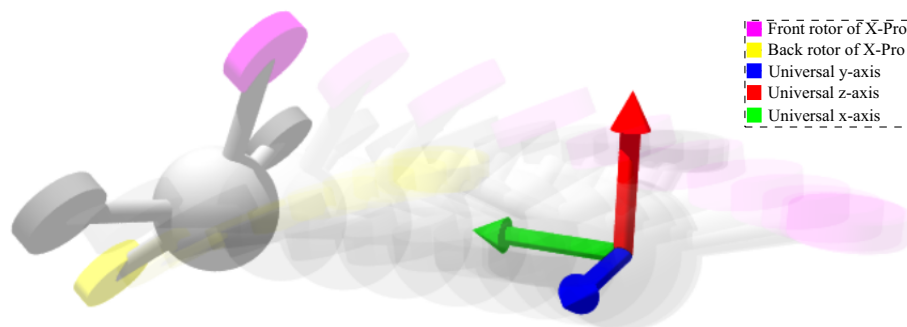
**Description:** The X-Pro is oriented horizontally with the roll and yaw controls are set to zero. Furthermore the sum of  ${}^1f_{aero,j}$  from the rotors are also set to be equal to the force of gravity acting on the X-Pro. The pitch control are set to a small value above zero which means that the front rotor is increased and the back rotor is decreased, which is keeping the sum of total aero forces  ${}^1f_{aero,j}$  the same.

**Expected result:** As the sum of  ${}^1f_{aero,j}$  is kept constant it should result in the X-Pro should stay in its original position but as the resulting force of the X-Pro has changed direction giving the X-Pro a force in  $x$ -axis and decreasing the force in  $z$ -axis, which should result in a positive movement along with a downwards acceleration of the X-Pro as shown in Figure A.3 on the next page.

**Test result:** The model has been tested with both a positive, i.e. higher aero force on

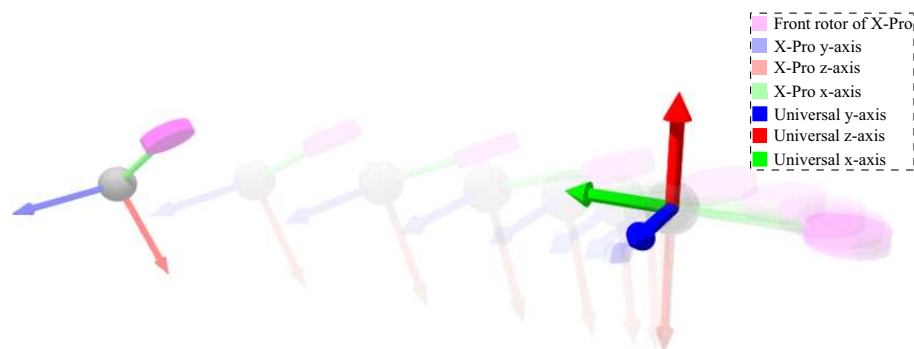


**FIGURE A.2:** *The result of the throttle test of the X-Pro with a positive throttle as input.*



**FIGURE A.3:** *Illustration of the expected movement of the X-Pro when the model receives a pitch input, i.e. higher aero force on the front rotor and a lower aero force on the back rotor. As the figure shows the X-Pro performs a movement in the positive universal x-axis direction along with a slight drop in attitude.*

front and lower on back rotor, and negative pitch on the front rotor. Figure A.4 shows the case where the positive pitch has been given. As the figure shown the X-Pro has an acceleration in the positive  $z$ -axis.



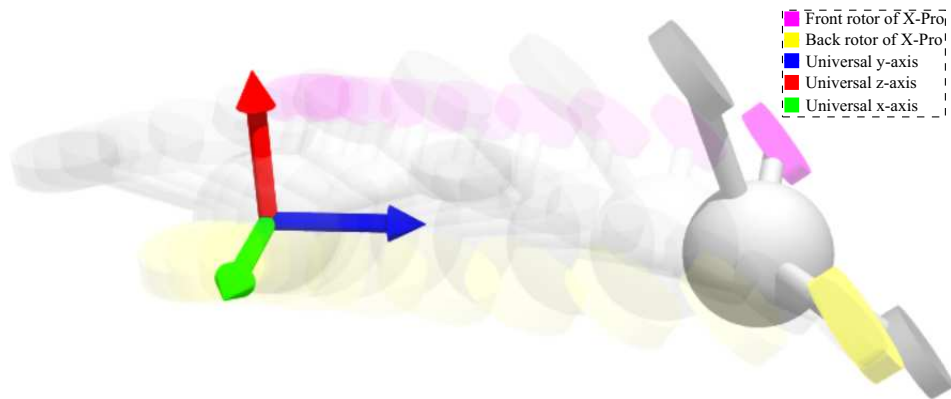
**FIGURE A.4:** *The result of the pitch test of the X-Pro where an acceleration in the positive  $x$ -axis direction along with a slightly downwards acceleration in the  $z$ -axis direction and a change in orientation can be seen.*

### TEST CASE 3: ROLL

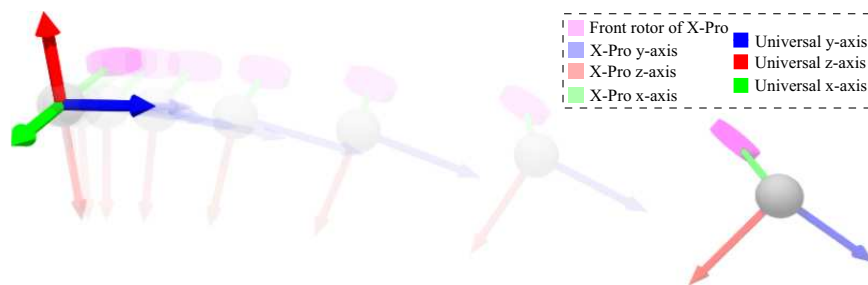
**Description:** The X-Pro is oriented horizontally with the pitch and yaw controls are set to zero. The sum of  ${}^1f_{aero,j}$  from the rotors are also set to be equal to the force of gravity acting on the X-Pro. The roll control are set to a small value above zero which means that rotor 4 is increased and rotor 2 is decreased, which is keeping the sum of total aero forces  ${}^1f_{aero,j}$  the same.

**Expected result:** The sum of forces from the rotors  $f_{aero,j}$  are the same as in the throttle case, i.e. equal to the gravity force acting on the X-Pro. But since the direction of the resulting force has changed to give the X-Pro a force in  $y$ -axis and a decreased force in  $z$ -axis the movement of the X-Pro is expected to accelerate in the positive  $y$ -axis along with a negative acceleration in the  $z$ -axis, which is illustrated in Figure A.5 on the next page.

**Test result:** The roll test has been performed where both a roll in the positive and negative  $y$ -axis direction has been performed. The case with a roll in positive  $y$ -axis direction is depicted on Figure A.6 on the facing page, which shows that the X-Pro accelerates in the positive  $y$ -axis direction and a slight downwards acceleration along with a small change in the orientation of the X-Pro.



**FIGURE A.5:** Illustration of the expected movement of the X-Pro when the model receives a roll input, i.e. higher aero force on the left rotor and a lower aero force on the right rotor. The expected movement of the X-Pro is as shown a movement in positive y-axis direction and a fall in the z- axis direction.

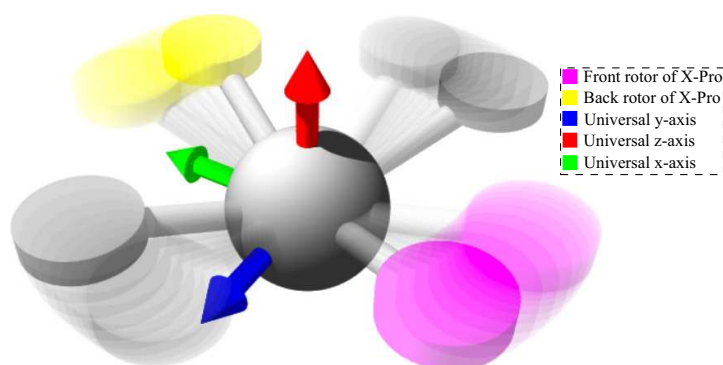


**FIGURE A.6:** The result of the roll test of the X-Pro which shows an acceleration in the positive y-axis direction along with a slightly downwards acceleration in the z-axis direction and a change in orientation.

## TEST CASE 4: YAW

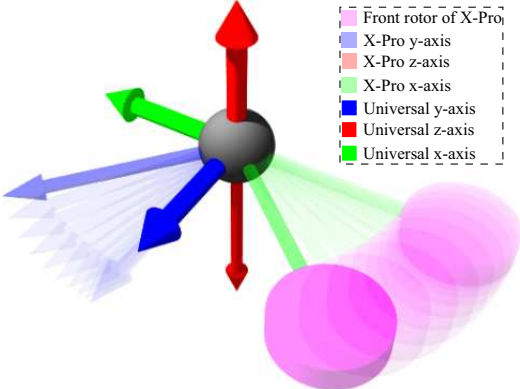
**Description:** The X-Pro is oriented horizontally with the pitch and roll controls are set to zero. The sum of  $f_{aero,j}$  from the rotors are also set to be equal to the force of gravity acting on the X-Pro. The yaw control are set to a small value above zero which means that the aero moment  ${}^1n_{aero,j}$  of rotor 1 and 3 are increased and decreased for rotor 2 and 4 with the same factor, which should keep the sum of total aero moments  ${}^1n_{aero,j}$  the same.

**Expected result:** As the sum of aero moments are kept constant the X-Pro should stay steady in the same height but rotate around its  $z$ -axis because the moment generated by rotor 1 and 3 are higher then from rotor 2 and 4, which should result in a clockwise rotation of the whole X-Pro as illustrated in Figure A.7.



**FIGURE A.7:** Illustration of the expected movement of the X-Pro when the model receives a yaw input, i.e. higher aero moment on the front and back rotor and a lower aero moment on the left and right rotor. The expected result is that the X-Pro should rotate clockwise around itself without changing its position.

**Test result:** The yaw test were performed both when giving the X-Pro higher aero moments on the front and back rotor and lowering the left and right rotor and the inverse case. Figure A.8 on the facing page shows the that when giving higher aero moments on front and back and lower on left and right rotor which results in a change in orientation on the X-Pro while keeping the same position.



**FIGURE A.8:** *The result of the yaw test of the X-Pro where the X-Pro rotates clockwise around the z-axis while staying in the same position.*





# QUATERNIONS



This appendix is the written work of group 1037b, [GHB07]. The appendix gives an overview of some basic properties of the quaternions and how it is used on the X-Pro. The appendix begins with an introduction and definitions of specific properties for the quaternions. This is followed by an example of how quaternion multiplication and rotation by the use of quaternions are performed. The appendix ends with the time derivative of the quaternion.

## B.1 INTRODUCTION

The quaternion consists of four parts

$$q = [q_1 \quad q_2 \quad q_3 \quad q_4]^T = \begin{bmatrix} q_{1:3} \\ q_4 \end{bmatrix} = \begin{bmatrix} \hat{e} \sin(\frac{\theta}{2}) \\ \cos(\frac{\theta}{2}) \end{bmatrix} = q_1 i + q_2 j + q_3 k + q_4 \quad (\text{B.1})$$

Where  $q_{1:3}$  is a vector part and  $q_4$  is a scalar part and  $\theta$  is the angle that is going to be rotated. The four parameters in the vector  $q$  describes a finite rotation about an arbitrary axis, which is called Euler parameters. The quaternions satisfy the following rules, known as Hamilton's rules[Wolb].

$$i^2 + j^2 + k^2 = i j k = 1, \quad (\text{B.2})$$

$$ij = k, \quad ji = -k, \quad (\text{B.3})$$

$$jk = i, \quad kj = -i, \quad (\text{B.4})$$

$$ki = j, \quad ik = -j. \quad (\text{B.5})$$

Because of the nature of the quaternion the conjugated of the quaternion is defined in [Wolb] to be:

$$q^* = [-q_1 \quad -q_2 \quad -q_3 \quad q_4]^T = -q_1 i - q_2 j - q_3 k + q_4 \quad (\text{B.6})$$

In addition to the Euler parameters the quaternions have another property stated in Euler's rotation theorem [Wola] the relationship between the four quantities is that the norm equals one:

$$|q| = \sqrt{q q^*} = \sqrt{q^* q} = \sqrt{(q_1^2 + q_2^2 + q_3^2 + q_4^2)} = 1 \quad (\text{B.7})$$

Furthermore the inverse of the quaternion is found to be equal to the complex conjugated quaternion, which yields:

$$\forall q(|q| = 1) \quad q^{-1} \equiv q^* \quad (\text{B.8})$$

## B.2 QUATERNION MULTIPLICATION

Because of the fact that the quaternions are non-commutative multiplication of quaternions has to be carried out using quaternion multiplication. The usage of quaternion multiplication is given in an example:

$$q p = (q_1 i + q_2 j + q_3 k + q_4)(p_1 i + p_2 j + p_3 k + p_4) \quad (\text{B.9})$$

$$= \begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \quad (\text{B.10})$$

$$= \begin{bmatrix} S(q_{1:3}) + I_{3 \times 3} \cdot q_4 & q_{1:3} \\ -q_{1:3}^T & q_4 \end{bmatrix} p \quad (\text{B.11})$$

$$= \begin{bmatrix} -S(p_{1:3}) + I_{3 \times 3} \cdot p_4 & p_{1:3} \\ -p_{1:3}^T & p_4 \end{bmatrix} q \quad (\text{B.12})$$

As it can be seen the skew-symmetric matrix changes sign depending on the order of the two quaternions.

## B.3 ROTATION OF QUATERNIONS

The quaternion describes the three-dimensional rotation between two coordinate systems described in  $\mathfrak{R}^3$ . It is desired represent a vector  ${}^1\omega_1$ , which describes the angular velocity of the object in coordinate system {1} in its own coordinate system, relative to the universal coordinate system {0} instead. To fulfill this demand it the vector  ${}^1\omega_1$  it is necessary to rotate it to coordinate system {0}, which is done in Equation B.13, and rotated back again to the initial coordinate system in Equation B.14.

$$\begin{bmatrix} {}^0\omega_1 \\ 0 \end{bmatrix} = {}^0_1q \begin{bmatrix} {}^1\omega_1 \\ 0 \end{bmatrix} {}^0_1q^* \quad (\text{B.13})$$

$$\begin{bmatrix} {}^1\omega_1 \\ 0 \end{bmatrix} = {}^0_1q^* \begin{bmatrix} {}^0\omega_1 \\ 0 \end{bmatrix} {}^0_1q \quad (\text{B.14})$$

Where  $\begin{bmatrix} 0 & \omega_1 & 0 \end{bmatrix}^T$  is the original angular velocity matrix represented on quaternion form with the real part  $q_4 = 0$ .

## B.4 TIME DERIVATIVE OF A QUATERNION

To determinate the attitude of the X-Pro the time derivative of the quaternion is needed to be known, to predict how the X-Pro rotates. The time derivative of the quaternion is described in both [Wer94] and [Wiś00], which both are described in the following:

**Extract B.4.1 (from [Wer94])** According to [Wer94, p. 512]:

$$q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T = q_1i + q_2j + q_3k + q_4 \quad (\text{B.15})$$

$$\dot{q} = \frac{1}{2}\Omega q \quad (\text{B.16})$$

$$\Omega = \begin{bmatrix} 0 & \omega_w & -\omega_v & \omega_u \\ -\omega_w & 0 & \omega_u & \omega_v \\ \omega_v & -\omega_u & 0 & \omega_w \\ -\omega_u & -\omega_v & -\omega_w & 0 \end{bmatrix} \quad (\text{B.17})$$

where

- $q$  is a quaternion representing the orientation of the rigid body with respect to the reference system [Wer94, p. 511].
- $\omega$  is the instantaneous angular velocity vector.

**Extract B.4.2 (from [Wiś00])** According to [Wiś00, pp. 9, 12, 18-19]:

$$q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T = q_0 + q_1i + q_2j + q_3k \quad (\text{B.18})$$

$$\frac{d}{dt} {}^p q = \frac{1}{2} R(\Omega_{po}) {}^p q \quad (\text{B.19})$$

$$R(q) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \quad (\text{B.20})$$

which is equivalent to:

$$\dot{q}_0 = -\frac{1}{2}\Omega_{po}\mathbf{q} \quad (\text{B.21a})$$

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega_{po}q_0 - \frac{1}{2}\Omega_{po} \times \mathbf{q} \quad (\text{B.21b})$$

where

- ${}^p_oq$  describes  $\{p\}$  in  $\{o\}$  [Wiś00, p. 12]
- $\Omega_{po}$  describes the angular velocity of the coordinate system  $\{p\}$   $\{o\}$ , resolved in  $\{p\}$  coordinates [Wiś00, p. 18].

Converted to the notation used in this project, both sources state:

$${}^0_1\dot{q} = \frac{1}{2} {}^0_1q \begin{bmatrix} {}^1\omega_1 \\ 0 \end{bmatrix} \quad (\text{B.22})$$

Notice that the angular velocity is expressed in coordinates of the X-Pro. The quaternion derivative can also be expressed as a function of the angular velocity expressed in world coordinates since:

$${}^0_1\dot{q} = \frac{1}{2} {}^0_1q \left( {}^0_1q \begin{bmatrix} \omega_1 \\ 0 \end{bmatrix} {}^0_1q^* \right) \quad (\text{B.23a})$$

$$= \frac{1}{2} \begin{bmatrix} \omega_1 \\ 0 \end{bmatrix} {}^0_1q \quad (\text{B.23b})$$

${}^0_1\dot{q}$  is then integrated element-wise.



## C.1 ASSIGNMENT OF TIMERS

This section describes the assignment of the timers available on the Robostix processor, Atmega128. First the various tasks, where timers are considered necessary are described and furthermore what priorities lay the grounds for distributing the timers. Secondly, it describes what tasks the four timers on the Atmega128, are assigned to, and how they are configured.

### C.1.1 TASKS

The Atmega128 contains four timers, which can be used for different tasks on the processor. The tasks where time is considered a necessity are:

**PWM for motors:** The four motors are each interfaced through a PWM signal with a period of  $1/300$  s. It should be noted, that the signal is inverted, as a signal of constantly 5 V (100% duty-cycle) will result in no voltage to the motors, and vice versa. The resolution must be high, since the nominal operating area is considered to be very narrow compared to the possible operating area of the motors.

**R/C-receiver:** The R/C-input is read through a single interrupt-pin, and it must be possible to measure the time between interrupts in a reasonable resolution. The time interval should be at least 1 ms, except if the Robostix is enabled in the middle of a signal, by which the time interval could be down to  $\approx 0$  ms. Nominally, there should be between 1 ms and 2 ms between pulses, but between sequences/packages of R/C inputs, there can be up to 22.5 ms –  $5 \times 1$  ms. For precision control, it is desired to also have a high resolution on the time interval measurements.

**Tachometer:** At hover, the rotors rotate at  $\approx 146$  rad/s  $\approx 23.25$  rounds/s. The maximum expected rotor velocity is 30 rounds/s, which would introduce a pulse every  $1/30$  s. If the rotor is put to a halt, no pulses would be introduced, by which the rotor velocity would not be updated. It is a possibility to define a timeout, after which the rotor is defined as zero or low velocity. Based on experiences from the previous project on the X-Pro with the same tachometers, it is found that a threshold of  $1/8$  s is suitable [KSG<sup>+</sup>06, Fig. 13.2 p. 105].

**Magnetometer:** To take measurements from the magnetometer, a commandbyte is transmitted to it, and after a delay, the magnetometer replies with `Data Ready (DRDY)`. If the magnetometer does not respond within a certain maximum period, it can be assumed that either the commandbyte has not been transmitted correctly, the magnetometer is without power, or the communication is lost. The same can be assumed if the magnetometer replies before a certain minimum period. Due to this, it must be possible for the magnetometer driver to determine the time between sending the commandbyte and receiving `DRDY`.

Also, the magnetometer transmits through Serial Peripheral Interface (SPI) and two extra I/O-ports. The Atmega128 has an SPI port, but this is used for programming it, so a software SPI-driver is developed. According to the magnetometer datasheet [PNI06], a pulse must be at least 100 ns long. One clock cycle on the Atmega128 is  $1/16 \mu s = 62.5 \text{ ns}$ , and a maximum of 16 bits are transmitted, which gives a minimum of  $2 \cdot 16 \cdot 100 \text{ ns} = 3.2 \mu s$  transmission time. Additionally, there is a risk that it will take more clock cycles to calculate the correct transmission sequence. A prototype implementation of the software SPI has been developed and implemented, and showed  $\approx 60 \mu s$  for receiving 16 bits, which is considered acceptable. Thus, a timer for timing the SPI transmission is not necessary.

### C.1.2 PRIORITIES

The allocation of the timers is based on the following priorities:

- *Some timers are by hardware dedicated to specific purposes.* This priority is made with respect to the fact, that two of the timers are dedicated for producing PWM-signals, which is useful for the motor HW interface.
- *Many software interrupts should be avoided.* This is due to the fact, that each interrupt creates an overhead. If the interrupt can be avoided by changing the prescaling of the clock to the timer or choosing a timer with larger resolution, this should be done first.
- *Tasks are grouped by similar time characteristics.* Since there is a limited number of timers, some tasks must share the same timer. This grouping is based on tasks, where the timer period or clock frequency can be the same, without violating the demands described in C.1.1 on the previous page.

### C.1.3 AVAILABLE TIMERS

The Atmega128 has four timers available, which are described here, based on the datasheet [Atm, p. 92-161]:

**Timer/Counter0:** 8-bit timer, one double-buffered output compare register for glitch-free PWM generation, designed for external 32 kHz watch crystal or prescaled system clock ( $1/1$ ,  $1/8$ ,  $1/32$ ,  $1/64$ ,  $1/128$ ,  $1/256$  or  $1/1024$ ).

**Timer/Counter1:** 16-bit timer, three double-buffered output compare registers for glitch-free PWM generation, possibility of external clock / event-generator, shared prescaler with Timer/Counter2 and Timer/Counter3 (individual prescale select of  $1/1$ ,  $1/8$ ,  $1/64$ ,  $1/256$  or  $1/1024$ ).

**Timer/Counter2:** 8-bit timer, one double-buffered output compare register for glitch-free PWM generation, possibility of external clock / event-generator, same prescaler as Timer/Counter1.

**Timer/Counter3:** Same properties as Timer/Counter1. Please note, that the output compare register `OC3B` and `OC3C` are not available on the Atmega128 output pins, since there are occupied by the tachometers.

### C.1.4 ALLOCATION OF TIMERS

Based on the tasks, the available timers and the priorities, the following assignment of timers has been made:

### C.1.5 TIMER/COUNTER1 + TIMER/COUNTER3

Timer/Counter1 and Timer/Counter3 are primarily assigned to generation of four PWM-pulses for the motor HW interface. Both timers have been configured as follows:

- Waveform Generation Mode: Fast PWM, `ICRn` as `TOP`.
- Compare Output Mode for 1A, 1B, 1C and 3A: Inverting mode (Set `OCnx` on compare match, clear at `BOTTOM`)
- Prescaler:  $1/1 \Rightarrow 16$  MHz
- `TOP`-value defines via `ICR1` and `ICR3`:  $\frac{16\text{MHz}}{300\text{Hz}} - 1 \approx 53\,332$  counts

This gives a theoretical maximum resolution of

$$\frac{\log(\text{TOP} + 1)}{\log(2)} = \frac{\log(53\,332 + 1)}{\log(2)} \approx 15.7 \text{ bits} \quad (\text{C.1})$$

and a frequency of:

$$\frac{16 \cdot 10^6}{53\,333} \approx 300.002 \text{ Hz} \quad (\text{C.2})$$

### C.1.6 TIMER/COUNTER0

Timer/Counter0 is assigned to the magnetometer, and is used to determine whether samples are returned from the magnetometer within a specific interval. The tachometer also uses this timer for determining intervals between pulses. The timer is configured as follows:

- Waveform Generation Mode: Normal, MAX as TOP.
- Compare Output Mode: Normal port operation, OC0 disconnected.
- Prescaler:  $1/1024 \Rightarrow 15\,625 \text{ Hz}$
- Interrupt on overflow. Increments an 8 bit variable, giving a timer with a total of 16 bits.

The resolution of this counter is:

$$\frac{1 \text{ count}}{15\,625 \text{ counts/s}} = 64 \mu\text{s/count} \quad (\text{C.3})$$

Using only the 8 bit timer, the maximum counter period is:

$$\frac{256 \text{ counts}}{15\,625 \text{ counts/s}} = 16.384 \text{ ms} \quad (\text{C.4})$$

This is not enough for determining the maximum delay of the magnetometer. Using the 16 bits available by combining the counter and the overflow counter, the maximum counter period is:

$$\frac{65\,536 \text{ counts}}{15\,625 \text{ counts/s}} = 4.19 \text{ s} \quad (\text{C.5})$$



### C.1.7 TIMER/COUNTER2

This timer/counter is assigned to the R/C-module to calculate the length of each channel. To avoid timer interrupts all the time timer/counter2 is configured by the following:

- Waveform Generation Mode: Normal,  $MAX$  as  $TOP$ .
- Compare Output Mode: Normal port operation,  $OC2$  disconnected.
- Prescaler:  $1/64 \Rightarrow 250$  kHz
- Interrupt on overflow. Increments an 8 bit variable, giving a timer with a total of 16 bits.

The resolution of this counter is:

$$\frac{1 \text{ count}}{250000 \text{ counts/s}} = 4 \mu\text{s/count} \quad (\text{C.6})$$

Using only the 8 bit timer, the maximum counter period is:

$$\frac{256 \text{ counts}}{250000 \text{ counts/s}} = 1.024 \text{ ms} \quad (\text{C.7})$$

This is not enough for determining the total length of all R/C channels. Using the 16 bits available by combining the counter and the overflow counter, the maximum counter period is:

$$\frac{65\,536 \text{ counts}}{250000 \text{ counts/s}} = 262.144 \text{ ms} \quad (\text{C.8})$$

With the timers assigned to the necessary drivers, the design of the drivers can be initiated. The motor is treated in the following.

## C.2 MOTOR DRIVER

As described in Section C.1.5 on page 173, Timer/Counter1 and Timer/Counter3 are used for generating the PWM signals. The duty cycle is set by changing the output compare registers, which are each assigned to a motor. Both timers have double-buffered output compare registers, which essentially means, that new output compare values will be buffered until the timer hits the  $TOP$  value, after which they will take effect. This ensures a glitch-free PWM-signal when changing the duty cycles.

### C.2.1 INTERFACE

Two functions are considered necessary for the Motor SW driver: One for initializing the PWM signals, and one for updating the dutycycles.

**void motor\_hw\_init(void):** This function initializes the timers with the parameters described in Section C.1.5 on page 173. By default, the dutycycle is set to 0%.

**void motor\_set\_duty(motor\_package\_t \*):** This function takes four dutycycles in a package, saturates them to be between `BOTTOM` and `TOP`, and puts them in the output compare buffer, see Figure C.1.

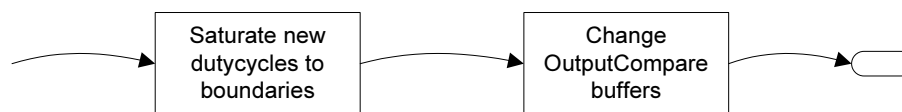


FIGURE C.1: *The procedure for updating dutycycles.*

This concludes the motor PWM driver. Next up is the driver for the magnetometer.

## C.3 MAGNETOMETER DRIVER

The magnetometer has an onboard processor, which takes samples on demand. In its datasheet, a sequence of signals for taking a sample is described, and an FSA is designed based on this description.

### C.3.1 OVERALL DESIGN

The FSA has been designed in such a way, that it is possible to detect certain errors in the communication. These errors are:

- Unexpected signals such as `DRDY` when sending a command, or `!DRDY` when receiving a sample. This could indicate a loose wire, or a fault in the transmission.
- Timeout, when the magnetometer does not respond within a maximum delay. This could indicate either a fault when transmitting the command, a loose wire, or the magnetometer is without power.
- `DRDY` before a minimum, indicating a loose wire, or a fault in the transmission of the command.

When an error has been detected, the driver will retry to collect a sample. After three retries, the communication to the magnetometer is declared lost to the main system, but the driver will continuously try to re-establish contact to the magnetometer. Figures C.2, C.3 and C.4 illustrate how the driver is designed to control the communication with the magnetometer, keeping track of samples and shifts between sampling one axis at the time, respectively. The magnetometer driver will independently request samples for one axis at the time, and make them available to the main system.

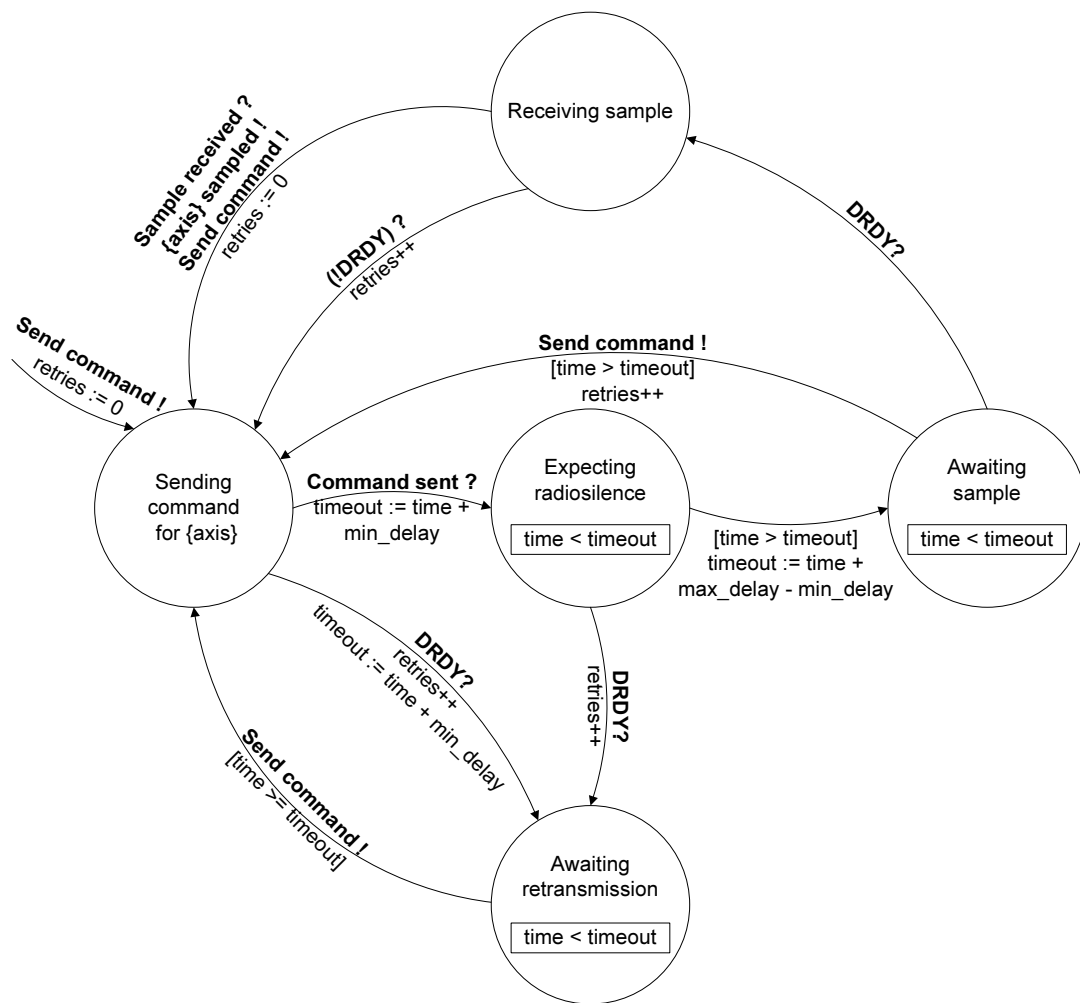


FIGURE C.2: The FSA for the magnetometer driver.

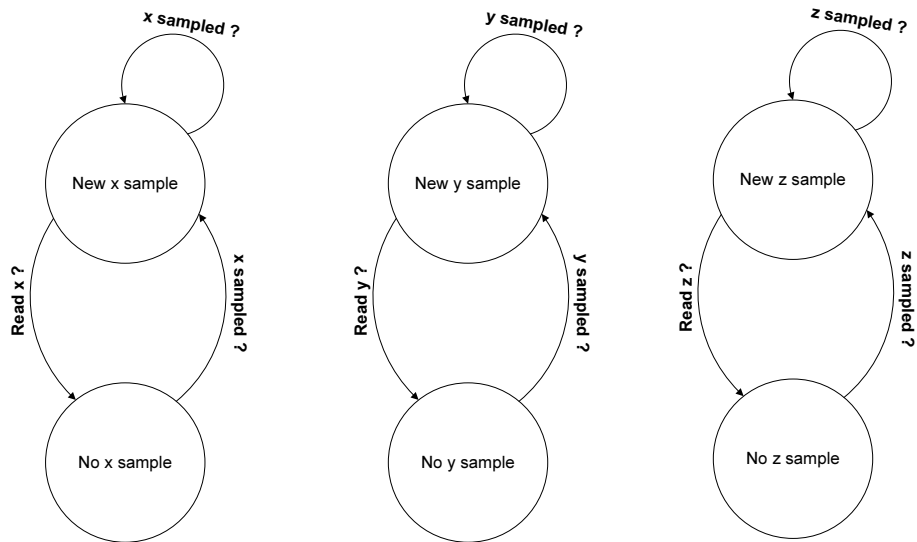


FIGURE C.3: The FSA for the availability of new samples.

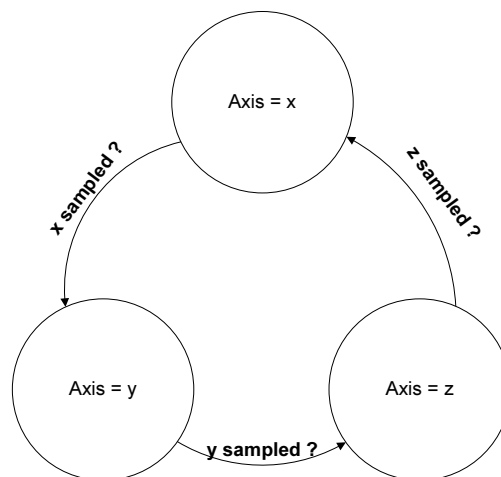


FIGURE C.4: The FSA for the current axis being sampled.

### C.3.2 INTERFACE

The magnetometer driver interface defines three types specific for the magnetometer driver:

**mag\_axis\_t:** This is an enumeration type of the values `mag_x_axis`, `mag_y_axis` and `mag_z_axis` to be used in function calls when reading a sensor.

**mag\_sensorvalue\_t:** This is the type used for storing received samples.

**mag\_return\_t:** This is an enumeration type of the values `mag_new_sample`, `mag_no_sample` and `mag_comm_error`, which is the possible values to be returned, when the main system requests the magnetometer driver for samples.

The interface is also defined by the functions, which can be called from the main system:

**void magnetometer\_init(void):** Initiates the Atmega128 hardware associated with the magnetometer driver, and initiates sampling from the magnetometer.

**mag\_return\_t magnetometer\_readsensor(mag\_axis\_t, mag\_sensorvalue\_t \*):** The function used for continuously asking the magnetometer driver for new samples of a certain axis. This function should be called with a maximum interval of the minimum delay between sending a command byte and receiving a sample.

**void magnetometer\_update\_state(void):** This should be called as frequent as possible, as it updates the state of the internal FSA.

**void magnetometer\_receive\_timerrollover(void):** ] This is used by the Timer/Counter0 overflow Interrupt Service Routine (ISR) to notify the magnetometer driver, that the timer has rolled over.

## C.4 TACHOMETER DRIVER

The tachometers have the purpose of measuring the velocity of each rotor. The tachometers are located on the X-Pro by the rotors, whereas the measurements are conducted by generating a pulse each time a rotor has taken one revolution. These pulses are lead directly into the Robostix through four interrupt pins, i.e. `INT3` to `INT6`. The FSA for the tachometer driver is described in the following.

### C.4.1 OVERALL DESIGN

The FSA for the tachometers is designed in such a way that if an error occurs in the sensor or the connection between the sensor and Robostix, the sensor should be discarded, i.e.:

- A minimum time between two pulses from the same tachometer has been set. If pulses occur more frequent than this minimum time, the reading is discarded. This could be an indication of a loose wire, noise induction, or a sensor specific error.
- A maximum time between two pulses has been introduced to define, that if the time is larger than this value, the velocity of the rotors are one third of the velocity in hover and are therefore defined to stand still. This is set because the time to detect that the rotors are standing still would take too long time if it would detect it at all.

Figure C.5 illustrates the FSA for the tachometer driver. A revolution is defined as the time between two measured pulses signals from the same tachometer to the Robostix. When the driver is initialized it will wait for the first interrupt to come before doing anything else. The driver then measures the time it takes for the rotor to take one revolution.

This second pulse has to occur between a minimum and maximum time to be accepted. If two pulses from the same tachometer occurs within the minimum time the sample will be discarded and the driver will wait for another interrupt from that tachometer. If this happens three times in a row before a sample has been accepted, the tachometer will be discarded and the Robostix must be reset to make the discarded tachometer functional again. If the time is higher, than the maximum allowed time, the driver considers the rotor to be standing still. Figure C.6 illustrates how the tachometer driver updates the state that shows if a new sample is ready.

### C.4.2 INTERFACE

There are three types defined for the tachometer driver:

**tacho\_return\_t:** Is an enumeration of the type of values, `tacho_new_sample`, `tacho_no_sample` and `tacho_comm_error`, which are the values that can be returned when the main system investigates if there are new tachometer samples.

**tacho\_id\_t:** Is an enumeration of the type of values, `tacho_front`, `tacho_left`, `tacho_back` and `tacho_right`, which are the values that can be returned from the tachometer

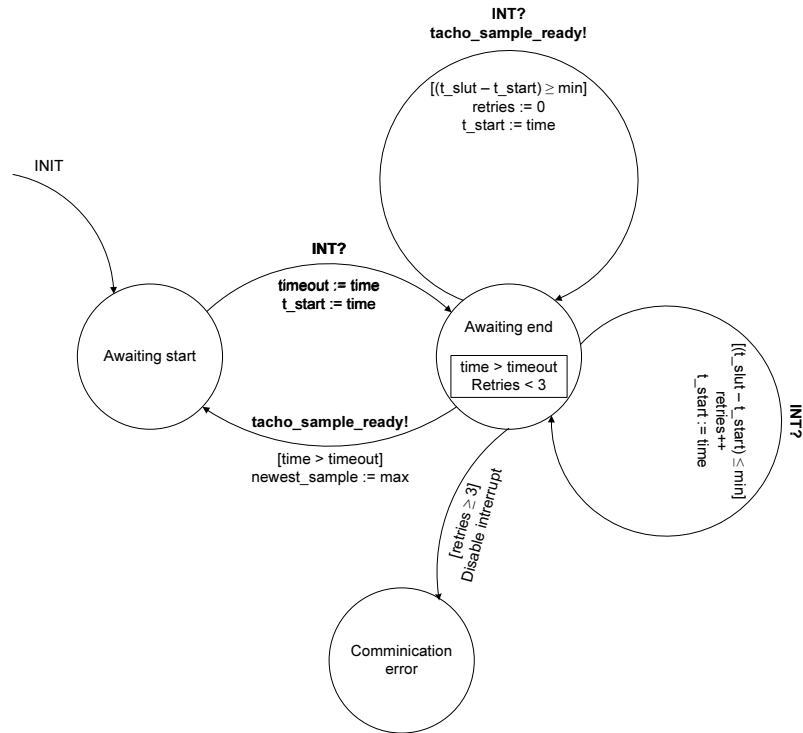


FIGURE C.5: The FSA for the tachometer driver.

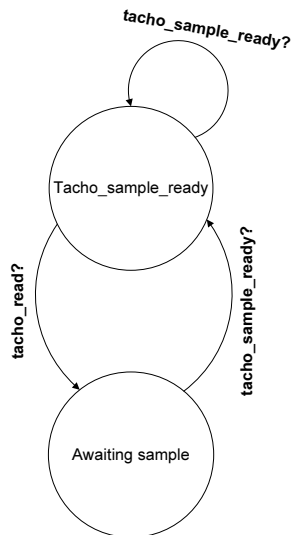


FIGURE C.6: The FSA for the availability of new samples in the tachometer.

ISR.

**tacho\_state\_t:** Is an enumeration of the type of values, `tacho_awaiting_t_start`, `tacho_awaiting_t_end` and `tacho_comm_err`, which are the possible states, the internal FSA of the tachometer can enter.

**long\_time\_t:** Is defines the type to be an unsigned 16bit integer. This type definition is made in the driver for Counter0.

Furthermore two functions are defined which can be called from the main system:

**void tacho\_hw\_init(void):** This function initializes the Robostix hardware used for the tachometer driver.

**tacho\_return\_t tacho\_sample\_ready(tacho\_id\_t, long\_time\_t \*):** This function is used from the main system, to check if there is a new tachometer measurement from any of the four tachometers.

This concludes the tachometer driver. Next up is the driver concerning the R/C module.

## C.5 IMU DRIVER

The IMU have the purpose of measuring how the X-Pro is moves during flight by measuring the angular velocity and linear acceleration. The measurements from the IMU are lead to the UART0 on the Robostix where an interrupt is generated every time a byte is received. The FSA design for the IMU is described in the following.

### C.5.1 OVERALL DESIGN

The FSA for the IMU is designed such that if an error happens in a complete IMU packet the packet will be discarded and the driver will search for the beginning of the next packet. The errors that are accounted for are:

- A checksum for each packet the Robostix receives is calculated to verify that the packet is correct. If there is a mismatch in the checksum the packet is discarded.
- Timeout occurs if a successful packet has not been received within a maximum delay. This could indicate either several faults in the received packet, a loose wire, or the IMU is without power.



Since the IMU is of great importance for controlling of the X-Pro, both when using the R/C and in autonomous flight, the IMU is not discarded permanently although several errors are detected, in the hope that one packet can be transferred to the Robostix successfully or the connection can be reestablished. The bytes send out as parts of a packet can be considered the result of a Discrete Event System running on the IMU - Figure C.7 illustrates an estimator for the system, based on the data sheet. Figure C.8 illustrates how the Robostix indicates that there is a new sample ready from the IMU.

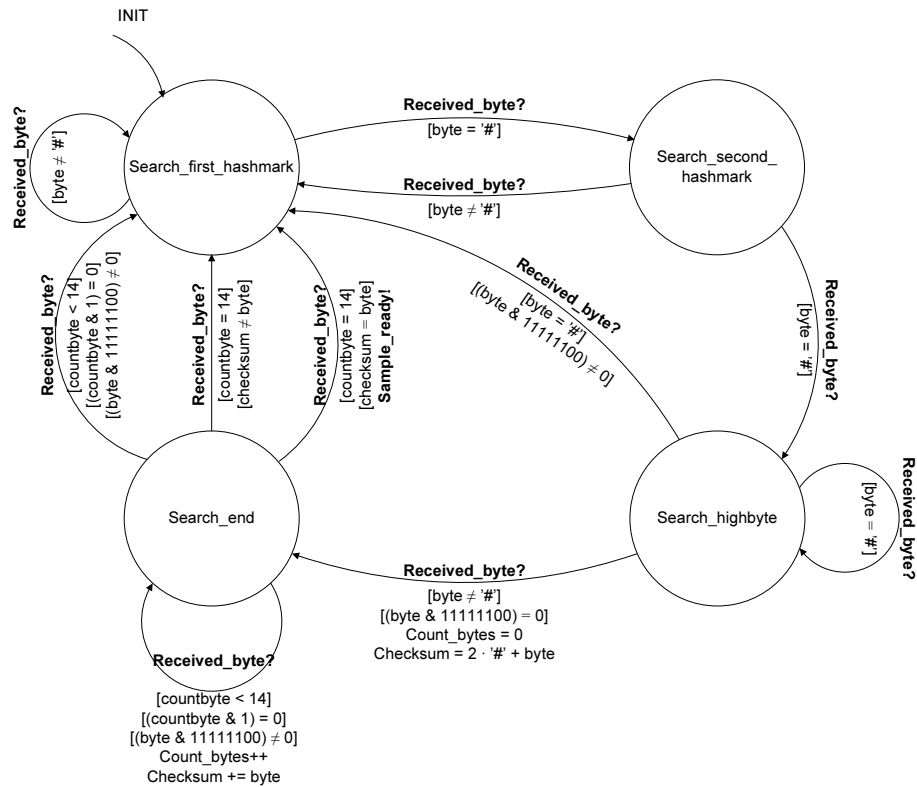


FIGURE C.7: The FSA for the IMU driver.

### C.5.2 INTERFACE

The IMU driver interface defines one type specific for the IMU driver:

**imu\_return\_t:** Is an enumeration of the type of values, `imu_new_sample`, `imu_no_sample` and `imu_comm_error`, which are the values that can be returned when the main system investigates if there are new imu samples.

**imu\_sample\_t:** This is a structure that contains the 14 bytes in the IMU packet, which is the type they are returned in.

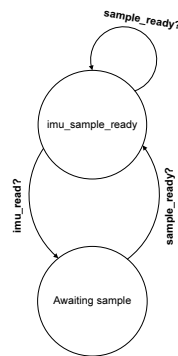


FIGURE C.8: *The FSA for the availability of new samples.*

Furthermore four functions are defined which can be called from the main system:

**imu\_return\_t imu\_sample\_ready(volatile imu\_sample\_t\*\*):** The function used for continuously asking the IMU driver for new samples.

**void imu\_receive\_timerrollover(void):** This is used by the Timer/Counter0 overflow ISR to notify the IMU driver, that the timer has rolled over.

**void imu\_hw\_init(void):** Initiates the Atmega128 hardware associated with the IMU driver.

**void imu\_update\_state(void):** This should be called as frequent as possible, as it updates the state of the internal FSA.

This concludes the final driver design. Now the only component left to describe is the communication with the Gumstix.

## C.6 ROBOSTIX/GUMSTIX INTERFACE

This section deals with the Robostix/Gumstix interface which is based on communication through an I<sup>2</sup>C bus. The main task for the software on the Robostix described in the previous section is to transfer sensor samples to the Gumstix. Equally important is it to receive calculated control inputs from the Gumstix. The following will describe specifically what data is sent back and forth.

### Data for transmission

**Robostix → Gumstix** The sensors sampled by the Robostix are listed in Table C.1 on the next page, along with information on their respective size of a sample and sampling frequency.

Sensor	Data-type	Size	Sample frequency
4× tachometer	4× uint16_t	4 × 2 bytes	[8 Hz, 30 Hz] for each
IMU	14× char	14 bytes	100 Hz
3× Magnetometer	3× uint16_t	3 × 2 bytes	≈ 25-50 Hz <sup>a</sup> (sequentially)
R/C status	—	(1 bit)	44.4 Hz
4× Motor input control (During other than Gumstix-pilot)	4× uint16_t	4 × 2 bytes	Unknown, max. 100 Hz

<sup>a</sup>Dependent on configuration.

TABLE C.1: The information of the sensor connected to the Robostix board.

Since it is possible to detect certain sensor errors on the Robostix, it has also been made possible to transmit these informations to the Gumstix. This information has not been included in the data for the sensors listed in the table, except for the R/C sensor. This is because it depends on the properties of the communication protocol, which will be designed later.

**Gumstix → Robostix** The motor control input transmitted from the Gumstix to the Robostix is listed in Table C.2.

Information	Data-type	Size	Sample frequency
4× Motor input control (During Gumstix-pilot)	4× uint16_t	4 × 2 bytes	Unknown, max. 100 Hz

TABLE C.2: The information of the sensor connected to the Robostix board.

As all data back and forth has been handled, the I<sup>2</sup>C interface will be described next.

### I<sup>2</sup>C bus connection

The available communication media between the Gumstix and the Robostix is an I<sup>2</sup>C bus. This is also connected to the sonic range finder SRF08, and a few devices used internally on the Gumstix and Robostix boards. Both Robostix and Gumstix support an I<sup>2</sup>C clock frequency of 400 kbit/s<sup>1</sup>. Consequently, the protocol design has been carried out with this bandwidth restraint.

**Bus setup** The I<sup>2</sup>C bus is a master/slave setup. Optionally, multiple masters can be present on the same bus, but with the risk of package collisions. The SRF08 is a slave, and can only be configured to be a master by its manufacturer, while both the Robostix

<sup>1</sup>This is defined as Fast-mode [i2c00, p. 8]

and the Gumstix can be configured to be either master or slave. The master can write to a slave, and read from a slave, while the slave can not initiate a transmission. Since it is necessary to transmit data both ways between Gumstix and Robostix, there are three options for the bus setup:

- Gumstix master / Robostix slave setup
- Robostix master / Gumstix slave setup
- Gumstix and Robostix dual masters setup

In either of the above mentioned cases, an issue is introduced with respect to the time of the sampling for each sensor. Delays will be present from sensors being sampled by the Robostix until the samples can be transmitted to the Gumstix. In the Robostix master / Gumstix slave setup and the dual masters setup, congestion can occur on the I<sup>2</sup>C bus when sensors are sampled at the same time. In the Gumstix master / Robostix slave setup, the Gumstix collects the samples from the Robostix on a regular basis, thereby also introducing delays. The magnitude of the delays depends on the chosen I<sup>2</sup>C setup, as well as the possible solutions to it, such as timestamping samples.

Considering the benefits and drawbacks, the Gumstix master / Robostix slave setup was chosen as the appropriate setup. The most frequently updated sensor is the IMU at 100 Hz, so it was decided to let the Gumstix collect samples from the Robostix at the same frequency.

The I<sup>2</sup>C bus clock frequency is set to 400 kbits/s, which results in a maximum of 500 bytes per  $\frac{1}{100}$  s, including bytes for addressing.

### C.6.1 PACKET DESIGN

This section will define the packet structure for containing the information described in the previous section. Furthermore it will describe the chosen methods for calculating checksums and storing sensor status information. At the end, the final packet for transmitting the relevant data is presented.

#### **Package contents**

**Package length** Since each sensor has an individual sampling frequency, different possibilities for structuring the packet could be considered, i.e. fixed length packet and variable length packet. However due to the fact, that an API for the I<sup>2</sup>C bus has already been

developed and tested by the manufacturer of the Gumstix, it was desirable to use this. Since it does not support requests for variable length packets, it has been chosen to use packet of fixed lengths.

**Sensor status** The status of all sensors are transmitted as well as the status of the Robostix. With a few exceptions, the possible states are common for most sensors:

**Tachometers:** Each tachometer can return either *new sample*, *error* or *no sample*.

**IMU:** An IMU package is regarded as one sample, and can return either *new sample*, *error* or *no sample*.

**Magnetometer:** Each axis is sampled using the same sensor. Due to this, each axis of the magnetometer can return *new sample* or *no sample*, while the magnetometer itself can be either *ok* or *error*.

**R/C:** Samples from the R/C module are not forwarded to the Gumstix. Consequently, the R/C module will only report *ok* or *error*.

**Motor input control:** The motor input is only relevant when either the Robostix or a human is the pilot of the X-Pro. Since the values are calculated internally, an *error* state is not possible. Only *new sample* or *no sample* are regarded relevant.

**Robostix status:** *Idle*, *Human pilot*, *Robostix pilot* and *Gumstix pilot*. This, along with the status of all other sensors, is regarded sufficient information on the status of the Robostix.

For those sensors with two or four states, the information can be stored in one or two bits respectively. It was chosen to use the two bit representation of the data, since it contains the possibility for error detection, and since the remaining space on the second byte can be used for part of a checksum of the entire package.

**Checksum** A checksum is included in packets between Robostix and Gumstix, enabling the Gumstix to detect data corruption occurring in the transmission, if any.

The Fletcher 8 bit checksum algorithm was chosen due to its increased data integrity compared to a regular sum of the bytes, while retaining a relatively undemanding algorithm compared to other methods. For the string  $D$ , the checksum is based on the following formulas [UB90, App. I]:

$$A = \sum_{i=1}^n D[i] \tag{C.9a}$$

$$B = \sum_{i=1}^n ((n - i) \cdot D[i]) \quad (\text{C.9b})$$

This can be implemented as:

```

for (i = 1; i <= n; i++){
    A += D[i];
    B += A;
}

```

(C.10)

For packages from the Robostix to the Gumstix, the storage of sensor status' has left 6 bits to be used for other purposes, such as  $A$  truncated to 6 bits. Since the sensor status' will be placed in the start of the package, it would be necessary to calculate the checksum in advance when the Robostix is contacted. This can be a demanding task, compared to summing up the checksum as the bytes are sent, and include it as the final two bytes. Due to this, it has been chosen to use one byte for  $A$  and one for  $B$ , placed at the end of the package for checksum in both directions

For packet from the Gumstix to the Robostix, one byte is used for  $A$  and one for  $B$ . The complete packet structure is now ready to be defined.

### Complete package structure

This section describes the final structure of packets sent between Robostix and Gumstix. Even though the Robostix [AVR] is little endian, 16 bit values are transmitted in big endian format, which is also common for all sensors used in the project.

**Gumstix to Robostix** Table C.3 describe the package structure for transmission from the Gumstix to the Robostix. The abbreviations used are:

$MOTDC_F$ ,  $MOTDC_L$ ,  $MOTDC_B$ ,  $MOTDC_R$ : The duty cycles for the front, left, back and right motor respectively.

$CS_A$ ,  $CS_B$ : The checksum variables  $A$  and  $B$ , which are both unsigned 8 bit integers.

Byte	0	1	2	3	4	5	6	7	8	9
Contents	$MOTDC_F$	$MOTDC_L$	$MOTDC_B$	$MOTDC_R$	$CS_A$	$CS_B$				

**TABLE C.3:** Package structure for transmission of motor input control signals from the Gumstix to the Robostix

The size of this package sums up to 11 bytes, include initiation of transmission.

**Robostix to Gumstix** Table C.4 on the following page gives a description of the package, which is received by the Gumstix, when it requests the Robostix for sensor samples. The package is structured with status bytes first, then samples and the checksum last.

The abbreviations used are:

$ST_1$ : This is the first status byte, which uses two bits for the Robostix state, and one bit each for R/C status, calculated motor input *new sample*, magnetometer status, and magnetometer axes x, y and z *new sample*.

Bit	7	6	5	4	3	2	1	0
Contents	$^{RB}ST_1$	$^{RB}ST_2$	$^{RC}ST$	$^{MOT}NS$	$^{MAG}ST$	$^{MAG}NS_X$	$^{MAG}NS_Y$	$^{MAG}NS_Z$

The Robostix states  $^{RB}ST_1$  and  $^{RB}ST_2$  are defined in the following table:

$^{RB}ST_1$	$^{RB}ST_2$	State
0	0	Idle
0	1	Human pilot
1	0	Gumstix pilot
1	1	Robostix pilot

$ST_2$ : The second status byte contains information on the tachometers:

Bit	7	6	5	4	3	2	1	0
Contents	$^{TM}ST_F$	$^{TM}NS_F$	$^{TM}ST_L$	$^{TM}NS_L$	$^{TM}ST_B$	$^{TM}NS_B$	$^{TM}ST_R$	$^{TM}NS_R$

$ST_3$ : The third byte gives information of the known status of the IMU and indicates if an IMU package has been received:

Bit	7	6	5	4	3	2	1	0
Contents	$^{IMU}ST$	$^{IMU}NS$	0	0	0	0	0	0

$^{IMU}SA$ : Package received from IMU, without header and own checksum.

$^{MAG}SA_X, ^{MAG}SA_Y, ^{MAG}SA_Z$ : Magnetometer samples for the x, y and z axis respectively.

$^{TM}SA_F, ^{TM}SA_L, ^{TM}SA_B, ^{TM}SA_R$ : Tachometer samples for the front, left, back and right rotor respectively.

$^{MOT}CA_F, ^{MOT}CA_L, ^{MOT}CA_B, ^{MOT}CA_R$ : Calculated duty cycles for the front, left, back and right motor respectively.

$^{CS}A, ^{CS}B$ : Checksum variables A and B of the Fletcher checksum.

<b>Byte</b>	<b>0</b>				<b>1</b>				<b>2</b>							
<b>Contents</b>	ST <sub>1</sub>				ST <sub>2</sub>				ST <sub>3</sub>							
<b>Byte</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>		
<b>Contents</b>	IMU <sub>SA</sub>															
<b>Byte</b>	<b>17</b>			<b>18</b>			<b>19</b>			<b>20</b>			<b>21</b>		<b>22</b>	
<b>Contents</b>	MAGSA <sub>X</sub>						MAGSA <sub>Y</sub>						MAGSA <sub>Z</sub>			
<b>Byte</b>	<b>23</b>		<b>24</b>	<b>25</b>		<b>26</b>	<b>27</b>		<b>28</b>	<b>29</b>		<b>30</b>				
<b>Contents</b>	TM <sub>SAF</sub>			TM <sub>SAL</sub>			TM <sub>SAB</sub>			TM <sub>SAR</sub>						
<b>Byte</b>	<b>31</b>	<b>32</b>		<b>33</b>	<b>34</b>		<b>35</b>	<b>36</b>		<b>37</b>	<b>38</b>					
<b>Contents</b>	MOT <sub>CAF</sub>		MOT <sub>CAL</sub>		MOT <sub>CAB</sub>		MOT <sub>CAR</sub>									
<b>Byte</b>	<b>39</b>							<b>40</b>								
<b>Contents</b>	CS <sub>A</sub>							CS <sub>B</sub>								

TABLE C.4: Package structure for transmission of motor input control signals and sensor samples from the Robostix to the Gumstix

This concludes the Robostix software environment. The next section concerns the Gumstix software.



# ACCEPTANCE TEST SPECIFICATION



# D

This chapter concerns the acceptance test specification of the X-Pro platform. This is not written with respect to a previously written specification of requirements. However in this specific case the latter does not exist due to the fact that various requirements were developed during the 9th. semester and were never quite clear from the beginning. Hence this acceptance test will revolve around important issues from a user perspective point of view.

## D.1 VERIFICATION OF SENSOR DATA:

### D.1.1 ROBOSTIX PACKET

The Robostix packet is to be examined in order to fully verify the drivers written for the sensors. The software developed in MATLAB<sup>®</sup> and Simulink<sup>®</sup> based on the soft realtime target software [Bha] constitutes the test environment. The software can be found on the CD-ROM in `/source/` directory.

#### Overall Robostix packet test

This test will serve to verify that even during full rotary movement, the robostix packets are received correctly on the gumstix and further to the DHM.

1. **What to test?:** Is the robostix packets received with a frequency of 100 Hz on the gumstix and DHM?. Furthermore what is the jitter level, if any, of these packets?
2. **Necessary conditions:** A running robostix, gumstix and DHM. A full loaded battery.
3. **Expected results:** The packets are expected to be received with 100 Hz with almost no jitter.
4. **How to test?:** Run the target software as described above. Fix the quad rotor to a platform that can withhold the helicopter even in the full rotary state. Take the quad rotor into full rotary movement. Stay for one minute and collect data.
5. **Test results:** Within a time of 73.974746 s the DHM and thereby also the gumstix, received 7398 robostix packets.  $7398/73.974746 = 100.007$  Hz. The frequency of the

robustix packets is thereby verified to maintain the demand of 100 Hz. Figure D.1 shows the time stamps on the left and the jitter level on the right.

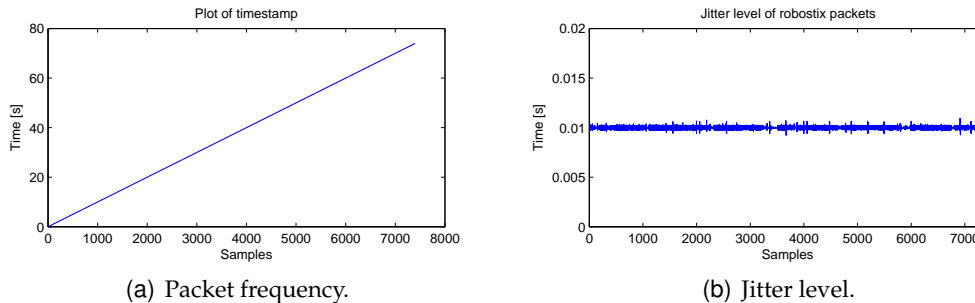


FIGURE D.1: Robustix packet frequency on the left and jitter level on the right.

The jitter level is accepted as the variations around 0.01 are quite small.

## D.1.2 RANGE FINDER

The ultra sonic range finder is connected to the I<sup>2</sup>C bus, and a overall test concerning this sensor is needed. Four tests are to be performed each test is described underneath and with their results immediately following.

### Packet verification

**What to test?:** Make sure that the data written in the packet, is put in the right order and that variables seem plausible.

**Necessary conditions:** It has been chosen to examen the packet content that is delivered to the DHM, thus the *gumstix\_main* program has to be running on the gumstix and the DHM has to run the data collection program, *gumstix\_dhm\_data*. The max range has been fitted in advance so that the range finder can complete the ranging in between sampling, the range finder gain is set to max. The priority of the developed software is set to the highest possibles. This is done to suppress other software on the gumstix and give the needed CPU time to the developed software.

**Expected results:** The packet is expected to look something like: [length], [light], [sec], [usec]

**How to test:** The system is turned on for one minute period. The data is saved in matlab and it is verified that the data is in the data structure. A plot of the sampled data is made.

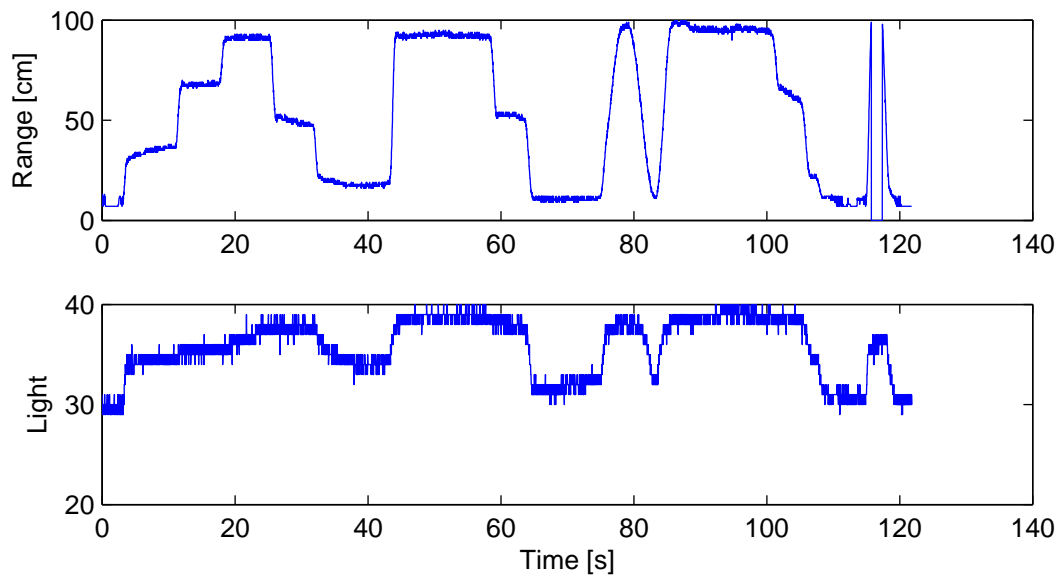


FIGURE D.2: The range and the light plotted over time when the X-Pro is displaced by hand.

**Test results:** The sampled data is shown in Figure D.2.

All together the content of the packet is verified, the range reading has some expected noise on it but not in a degree that would seem that the sensor or the communication line is faulty. Looking closer on the signal it can be seen that the range reading goes abruptly to zero and the end of the sample, this is the indication that the range finder did not receive a echo in the defined listening periode, thus out of range. The range limit was set to 99 cm which coincide whit the fall out of the range. The light sensor is verified since it has clear tendency which in strongly correlated with the movement, seen by the range.

### Frequency and jitter of sampling

**What to test?:** The sampling frequency has to be verified thus it has a direct effect on the estimation and control that relay on the sampling frequency. Jitter is unavoidable but it is wanted to estimate the worst case jitter between samples.

**Necessary conditions:** Same conditions as in Packet verification test.

**Expected results:** The frequency has to be 50Hz. The jitter has to be  $\pm 10\%$  or less of the sample periode in order to neglect the effect of the jitter.

**How to test:** The system is operated in non rotary movement for 1 minute, while data from the range finder is recorded on the host machine. All developed software

should be running on the Gumstix to give a realistic image of the possible CPU load on the Gumstix. During the sampling the range finder should be introduced to distances from standing on the ground to maximum range (>1m). The test data is to be saved and plots are made with matlab.

**Test results:** The sampled data is shown in Figure D.3.

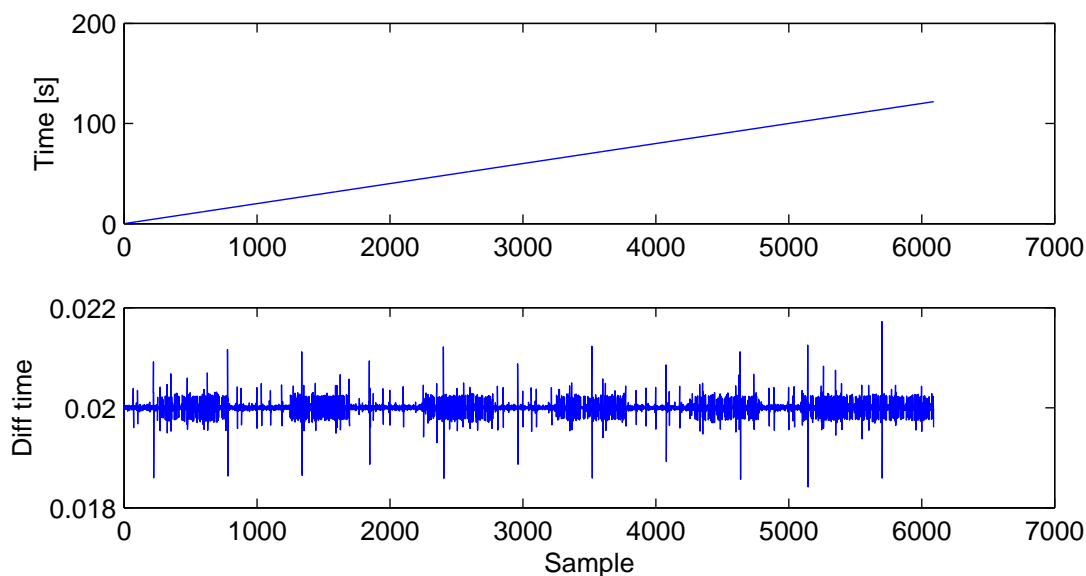


FIGURE D.3: From the top: Time stamp over samples and the time between two following samples.

The top graph show the samples and the corresponding time stamp, at first glance this curve seems perfect, but by taking the difference between two on each other following samples reveals the plot in the bottom of figure D.3. The mean time between samples is 19.99994 ms which is a highly acceptable sample rate. The jitter which in some sense is depicted in the the bottom graph, the maximum difference between tow samples is subtracted from the mean value, which yields a jitter of maximum 8.62 percent which is accepted but should be kept in mind when implementation is done.

### Maximum rotary movement

**What to test?:** The large fluctuating currents will induce electromagnetic noise that might influence the I<sup>2</sup>C bus that could result in erroneous reception.

**Necessary conditions:** Same conditions as in the packet verification test. Further more the X-Pro are strapped down and the motors are allowed to be running at nearly

maximum speed.

**Expected results:** It is expected that there will be no faulty reception on the I<sup>2</sup>C bus when the motors are running at maximum speed.

**How to test:** All software is started with the X-Pro connected to the battery and the Robostix is put into human control via the remote control. Data is logged with the X-Pro strapped down.

**Test results:** As it is evident in Figure D.4 there is no fall out of the signal from the range finder, seen by that there is no samples missing when plotting the time stamp for the test with maximum rotary movement.

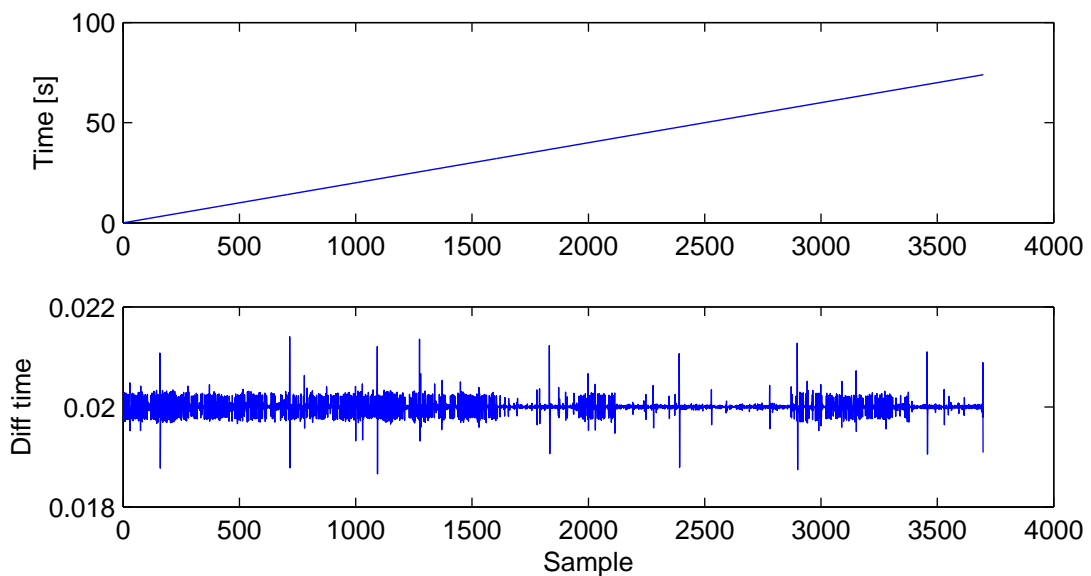


FIGURE D.4: From the top: Time stamp as a function of samples, and the time between two following samples.

### D.1.3 GPS MODULE

This section concerns the gps module mounted on the X-Pro. It has in previous work been verified through module tests that the gps in itself functions correctly. It is however imperative that it functions equally as well in its X-Pro environment. In order to verify the latter issue, both a non rotary and a full rotary test is performed in order to stress the gps module as it is likely to be stressed during X-Pro operation. The test is performed in both rotary states if not explicitly stated otherwise. The software test environment is constituted by the executables `x_pro` and `x_pro_startup` which are located on the

Gumstix and DHM respectively, and CD-ROM in `/source/`. It is important to run the `x_pro` program on the Gumstix with the `gps` argument as this configures the GPS module to output the correct packages and maximum frequency 4 Hz.

### Packet verification

Although the protocol specifies a large amount of packages that can be requested, only few are relevant. The packages are `NAV_SOL`, `NAV_STATUS`, `NAV_DOP`, `NAV_DGPS` and `NAV_SVINFO`. In order to verify that the packages are correctly received, the following items are examined.

**What to test?:** Are the before mentioned packages received correctly in both a non rotary and a full rotary state ?.

**Necessary conditions:** The test must be performed out in the open as

the `gps` module should have complete access to satellites. The weather has proved to be a significant factor in the detection of satellites and a good weather forecast with blue sky is therefore preferable. During the data logging, a *3D* fix must be obtained. In order for the test to proceed into the full rotary state the *3D* fix must also be attained.

**Expected results:** It is expected that the packages are received correctly. Furthermore that the package components are correct and have no outliers in their sequence. It is expected that the test is carried out successfully, also confirming that the `gps` module works adequately in the X-Pro environment,

**How to test?:** Find a nice spot in the field with good access to the sky and not too many buildings around. Due to Denmark's geographic position most of the satellites are geostationary to the south. Ensure at least that no buildings or other blocking devices are located at this position relative to the X-Pro. Power up and initiate programs on both gumstix and DHM for data logging. Start logging and wait till at least *3D* fix has been obtained. If fix is not achieved within five minutes restart log and try again until successful. Drive the X-Pro into a full rotary state on the ground. Perform data logging for another ten minutes and end test. Perform a full sanity check for all packages and their respective components. NB. remember to enable RTCM packages.

**Test results:** All the mentioned packages were received correctly both in rotary and non-rotary movement. A successful sanity check of the packages was also carried out. However it seems as the module has dropped one package set exactly at the instant when a *3D* fix was acquired. Refer to the results in the frequency and jitter section.

## Frequency and Jitter

It is important that packages are received at a constant rate, especially for the future controller and estimator which must be able to rely on punctual data.

**What to test?:** Can a 4 Hz frequency be guaranteed?. Does jitter occur and if so how profound?.

**Necessary conditions:** A sequence of data must be present from the previous packet verification test.

**Expected results:** It is expected that packages are received with a 250 ms interval. A minimum amount of jitter is expected, however if it occurs it should be constant within a small margin.

**How to test?:** Examine some of the gps packets from the data log. Every packet has a time stamp, so plot the difference in time between each sample.

**Test results:** The following graphs in Figure D.5 illustrate the rate at which the packets are received, and the 3D fix.

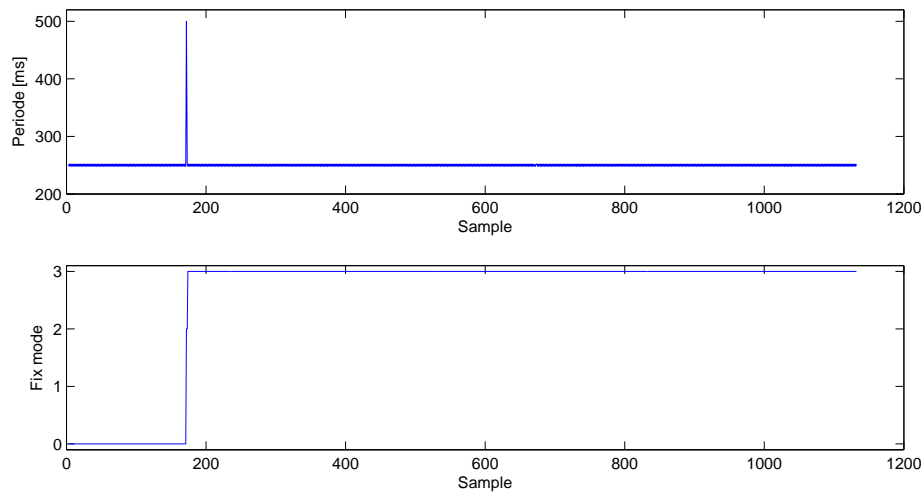


FIGURE D.5:

As mentioned earlier an outlier which is generated due to a missing packet set, is identified from the Figure at sample 171. The missing package instant occurs exactly when a 3D fix is obtained. The group has several possible ideas to rectify this problem, however it is believed that the problem lies within the gps module. It is believed that the module uses excessive computation power at the mentioned instant and therefore excludes packet I/O.

Apart from this single outlier the packets are received with a 250 ms interval, and has a constant small jitter level as seen on the graph.

### DGPS verification

The DGPS feature is important in order to obtain an absolute position of approx. two meters CEP.

**What to test?:** Does the gps module receive the RTCM packages ?. If so, is the latency of the RTCM packages less than or equal to one second? i.e. is the position estimate corrected in less than or equal to one second.

**Necessary conditions:** A sequence of data must be present from the previous packet verification test.

**Expected results:** The DGPS feature should be enabled and position estimates should be corrected within one second.

**How to test?:** After a successful package verification test - package data with DGPS information should be available. The **NAV\_DGPS** package contains the relevant data. More specifically the **AGE** and **STATUS** part of the payload will reveal whether the RTCM packages are received correctly and with what frequency the position estimate is updated.

**Test results:** The GPS module receives RTCM packets, and the latency of these are illustrated by the following Figure D.6.

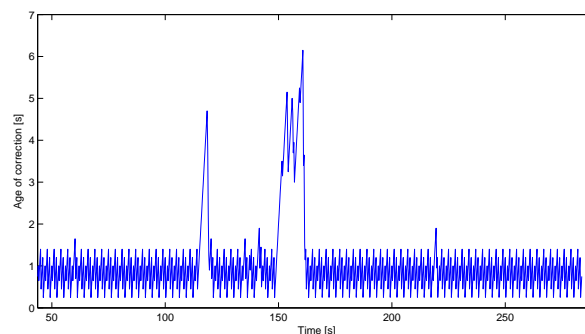


FIGURE D.6:

The Figure shows a fair amount of jitter. Correction packets are received mostly within a second, sometimes above. At two striking time intervals, correction packets are around five seconds delayed. It is believed that this time delay is due to



propagation delay in the network way from the Suldrup antenna to AAU. Furthermore the bandwidth used for the DGPS reception is shared and cannot be guaranteed.