

---

---

# Implementation of pitch on an ROV

- Retrofitting a Cougar XT -

---

---

Diploma project  
Rolf Magnus Roos

Aalborg University Esbjerg  
Department of Electronics & Computer Engineering  
Niels Bohrs Vej 8  
DK-6700 Esbjerg





**AALBORG UNIVERSITY**  
STUDENT REPORT

**Department of Electronics & Computer  
Engineering**

Niels Bohrs Vej 8  
DK-6700 Esbjerg  
<http://esbjerg.aau.dk>

**Title:**

Implementation of pitch on an ROV

**Theme:**

Diploma project

**Project Period:**

Fall Semester 2016

**Project Group:**

ED7-6

**Participants:**

Rolf Magnus Roos

**Supervisor(s):**

Zhenyu Yang  
Lean Ravnkilde Johansen

**Copies:** 1

**Page Numbers:** 63

**Date of Completion:**

January 10, 2017

**Abstract:**

The goal of the following project is to implement pitch manoeuvre upon an ROV system, which is currently incapable of doing this. The intention is that the pitch controller is fitted onto the ROV independent of the ROVs internal controllers.

To allow the ROV to pitch, a thruster is needed to generate the force that is required. Where the main focus of this project is to create the controller for the thruster which allows the ROV to be pitched.

To achieve this, a mathematical description of first the ROV and the thruster is extracted. The ROV model is based on a force balance, describing the forces impacting the pitch angle. It follows from this that the force required to be delivered for a given angle must be equal in magnitude but in opposite direction.

The whole system is combined in the Simulink toolbox in Matlab where the system can be simulated and analysed. From this a *PI*-controller is developed which allow the pitch angle to be maintained by the feedback system measuring the angle of the ROV.

# Contents

<b>Preface</b>	<b>0</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Project Description . . . . .	1
<b>2 Problem Analysis</b>	<b>4</b>
2.1 Cougar XT ROV . . . . .	4
2.2 Electronics Pod . . . . .	5
2.2.1 Control Unit . . . . .	5
2.2.2 Driver Board . . . . .	6
2.2.3 Pod Assembly . . . . .	7
2.3 ROV Dynamic Model . . . . .	8
2.3.1 Inertia Tensor . . . . .	10
2.3.2 ROV Torque . . . . .	13
2.3.3 Drag . . . . .	15
2.3.4 System Model . . . . .	17
2.4 Thruster Test . . . . .	19
2.4.1 Modelling Method . . . . .	19
2.4.2 SM-7 Test . . . . .	20
2.5 Conclusion . . . . .	24
<b>3 Solution</b>	<b>25</b>
3.1 SM-7 Model . . . . .	25
3.1.1 Data Preprocessing . . . . .	25
3.1.2 Model Identification . . . . .	28
3.2 System Simulation . . . . .	33
3.2.1 Thruster Block . . . . .	34
3.2.2 Disturbance Block . . . . .	35
3.2.3 ROV Block . . . . .	37
3.2.4 System Evaluation . . . . .	38
3.3 Controller Design . . . . .	40
3.3.1 P-Controller . . . . .	41
3.3.2 PI-Controller . . . . .	42
3.4 Conclusion . . . . .	43
<b>4 Discussion</b>	<b>45</b>
<b>5 Conclusion</b>	<b>49</b>
<b>6 Perspective</b>	<b>50</b>

<b>References</b>	<b>52</b>
<b>A Calculations</b>	<b>55</b>
A.1 Simplified Cougar model . . . . .	55
A.1.1 Volume & density of frame . . . . .	55
A.1.2 Foam volume & density . . . . .	56
A.1.3 Centre of Mass' . . . . .	57
<b>B DNV Drag Coefficients</b>	<b>59</b>
<b>C Code</b>	<b>60</b>
C.1 Load cell monitor . . . . .	60
C.2 Thruster Test Controller . . . . .	60
<b>D File Structure</b>	<b>63</b>

# Preface

The following project was written as the final project for my Diploma degree in Electronics and Computer engineering. The project was written in collaboration with SubC Partner, an offshore services and product supplier based in Esbjerg.

Throughout the project SubC Partner has been an invaluable help, and has been making all equipment used within the scope of the project available to me and should have a huge thanks for both this and the technical assistance which has been given to me as required.

Aalborg University Esbjerg, January 10, 2017

---

Rolf Magnus Roos  
<rroos13@student.aau.dk>

# Chapter 1

## Introduction

Remotely operated underwater vehicles (ROVs) describes a type of submersible vehicle with a very wide range of applications. It spans from small inspection ROVs, small vehicles carrying little more than a camera and a light source, crawler type ROVs which along with the ability to traverse through the water columns implements some type of locomotion to travel across submerged structures or the ocean floor to large work-class ROVs which can weight many hundreds of kilos and operate large and complex tools allowing the ROV to solve a wide range of tasks underwater.

This project will focus upon retrofitting a Saab Seaeye Cougar XT ROV with the goal of implementing a 7<sup>th</sup> thruster onto the ROV frame. The objective of the extra thruster is allow the ROV to be pitched, which the regular Cougar XT is not able to. The extra thruster will be implemented as a separate system from the ROV control system. This is done as the ROV platform is proprietary hardware and software as well as to ensure there will be no warranty issues. It does mean however, that it is unknown how the control system of the ROV will react when the ROV is being pitched.

In order to implement the new thruster, a stand alone system will be created which will be attached to the Cougar frame. Control of the thruster will be achieved using a dedicated thruster controller and communication to the topside will be achieved using communication interfaces on the ROV.

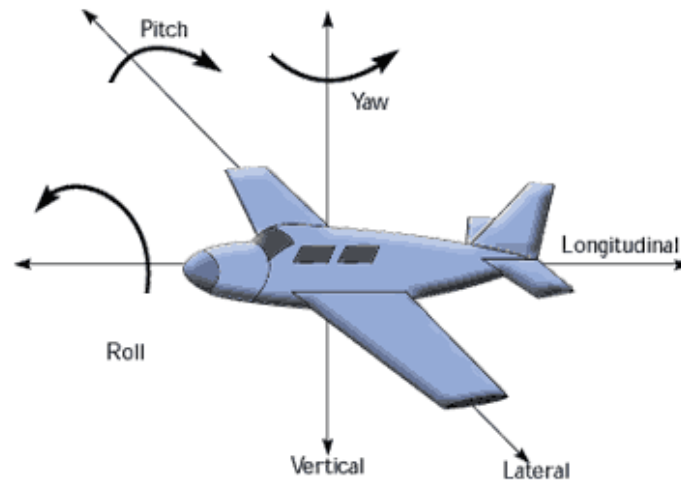
The project idea is based on an earlier incarnation of the same thruster being used as a ballast system. The idea being that the thruster will generate a force opposite to the changing buoyancy as the weight of the ROV shifts. As this system was being worked upon the idea to extend the system and create a dedicated thruster controller, able to maintain the pitch angle of the ROV was conceived.

The project is created in collaboration with SubC Partner, an offshore product and service provider located in the city of Esbjerg. The equipment for the project is all supplied by the company.

### 1.1 Project Description

As stated in the preceding section, the goal of the following project is to fit a thruster onto an ROV, the Cougar XT by Saab Seaeye. This thruster however, have to perform

a very specific task, allow the ROV to perform pitch manoeuvring. Figure 1.1 show the referencing convention for *pitch*, *yaw* & *roll*.



**Figure 1.1:** Pitch, roll & yaw explanation [1]

Retrofitting an ROV with a new thruster, and creating a system that allows the ROV to manoeuvre in a way not intended by the manufacturer is a risky procedure, and there are much that needs to be done. For this reason the task is divided into a number of smaller tasks that has to be performed.

The first task that needs to be accomplished is to create the electronics that allow the thruster to be controlled. This circuitry is dependent upon two factors, the first is the thruster. As the thruster for the project is the *SM-7* by Saab Seaeye, the signals to control it is defined from this. Secondly, as the pitch has to be controlled from the topside, some communication has to be implemented which facilitates this.

The next task is to create the control scheme for the system. As the idea of the system is to allow the user to set a wanted angle of the ROV, the system designed must be able to take this angular input and maintain this for the ROV. For this reason, a feedback system using an angular feedback line is used. To design this system, a model of the ROV system has to be created. This model will describe the ROV as it is being pitched, and the model have two functions. One is that it can be used to calculate the amount of thrust is needed to maintain a pitch angle, the second is it will assist in designing the controller for the thruster as it will allow simulation of the system without risking any hardware.

After this, the *SM-7* has to be described mathematically as well. This will first of all be used to simulate the full system so a controller can be created which can maintain the pitch of the ROV. For the implementation of a controller for the system it can also be used to evaluate the feedback of the system vs. the expected behaviour from the models.

Finally, the controller will have to be implemented for the Cougar system. This will require a few tests on mock ups to ensure the system behaves as expected before it is attached to the ROV.



No.	Task	Accomplished
1	Driver Electronics	
2	Communication	
3	ROV model	
4	Thruster model	
5	System simulation	
6	Controller design	
7	Implementation	

**Table 1.1:** Task list

Table 1.1 show the previously described tasks. This table will be used in Section 4-Discussion to evaluate the progress of the project at the conclusion of it.

## Chapter 2

# Problem Analysis

In the following section the requirements of the pitch functionality will be analysed. The necessities for the project will be identified and the analysis will lead to a list of components needed and a list of tasks to fulfil in order to create the thruster controller.

### 2.1 Cougar XT ROV

The Cougar is a work class ROV, and will be the subject which the pitch controller will be fitted. The standard configuration of the ROV allows for full 3-D movement within the water column[2] but with no possibility to change the pitch angle of the vehicle.

The ROV is fitted with six thrusters, two thrusters is positioned vertical to create ascend and descent thrust and the last four are vectored horizontally to allow the ROV to move freely in the plane. The controller of the ROV manages the thrust of all six thruster to manoeuvre as commanded.

The Cougar is connected to the topside through an umbilical, this carries power and communication to the ROV. The system has three separate power levels which are all provided through this umbilical.

- 440 VAC single phase for general ROV equipment[3]
- 500 VDC Thruster power supply[3]
- 660 VAC three phase separate tooling power supply[3]

Communication with the ROV is achieved through a fibre optic interface and inside the electronics pod of the system two optical MUX' is located, and these provide the following interfaces for communication.

- 8 x RS-232/422/485[3]
- 2 x 10/100 Mbps Ethernet[3]
- 4 x analogue video channels[3]

In the standard configuration of the Cougar, all video interfaces are used but only four RS-232 and one Ethernet interface is available at the bulkhead of the ROV. The remaining are theoretically available but it would be necessary to open the electronics pod of the ROV to connect the interfaces to output ports.

## 2.2 Electronics Pod

To implement the pitch controller it is necessary to first of all be able to control the speed and direction of the thruster. For this project a SM-7 is used. This is the same thruster that the Cougar uses. The design of the controller is based on information in the manual for the Cougar system, which has two thruster controller boards, these set both direction and speed for the individual thrusters. According to the handbook, the direction is set by a 24 VDC signal, *DIR+* and *DIR-*, where the polarity of the signal decides the direction of the thruster. The speed is by a 12 V PWM signal[3] referenced to the directional 0 V line.

### 2.2.1 Control Unit

As the thruster must be controlled from the surface, some communication between the topside and the control unit must be established. This is dependent upon the interfaces available from the ROV, as described in Section 2.1-Cougar XT ROV, either Ethernet or a serial interface is available. However, as the Ethernet interface is a relative high bandwidth interface, which is not needed to send fairly short data packages to the controller, a serial interface is chosen. In addition, this is done as there are more of these available, compared to the Ethernet interface. Finally, the serial interface is much simpler to implement as most microcontrollers has some communication facilities which can be used for serial communication.

The control unit also switches a digital signal, which will control the direction of rotation for the thruster. The challenge is that the voltage of the digital signal is 24 VDC which no microcontroller can achieve directly. The same issue is present with the PWM signal of 12 VDC. Most microcontrollers work between 3 VDC and 5 VDC. Therefore, to achieve the necessary voltage to control the thruster, a driver circuit must be created which convert the signals to the correct levels.

Finally, the control unit must be fast enough to maintain a control algorithm, the speed requirements for this is hard to foresee, as it is dependent on both the the amount of control elements, as well as the feedback. For example, if an analogue sensor has to be sampled to generate feedback for the controller, the speed of the ADC will limit how often a feedback signal can be generated.

There are many different microcontrollers available. Many (if not all) will be able to fulfil the stated requirements, so the choice of microcontroller is not easy. However, as a lot of work already lies in modelling the the ROV system and DC thruster, keeping the controller simple is an advantage. With this in mind, one of the most simple microcontrollers to use is an Arduino. The reason is two fold; First all circuit elements needed to allow the controller to function as specified has already been taken care of, this also includes managing the stability of the  $V_{cc}$  input for the controller[4]. Secondly, the programming language for the Arduino vastly simplifies the utilisation of functions for

the controller as the registers comes pre-initialised as functions are used. And if greater control is desired, the controller can be written to directly in C or even assembly. For this reason, an Arduino was chosen, and to keep the pod small, an Arduino nano is used, as it has a much smaller footprint than an Arduino Uno and the same performance is achieved.

### 2.2.2 Driver Board

As previously described, the 5 VDC direction and speed signals from the control unit must be converted to the correct 24 VDC directional and 12 VDC PWM speed signal for the thruster. The direction control signal is relatively simple as it needs to switch on and off without any significant speed requirements, relays switching a 24 VDC signal with a control voltage of 5 VDC will be used for this. As the controller has a limited current output, maximum 40 mA per I/O pin[4], a transistor will be used to switch the relay, this way the current to drive the coil in the relay is drawn from the power supply and not the Arduino itself.

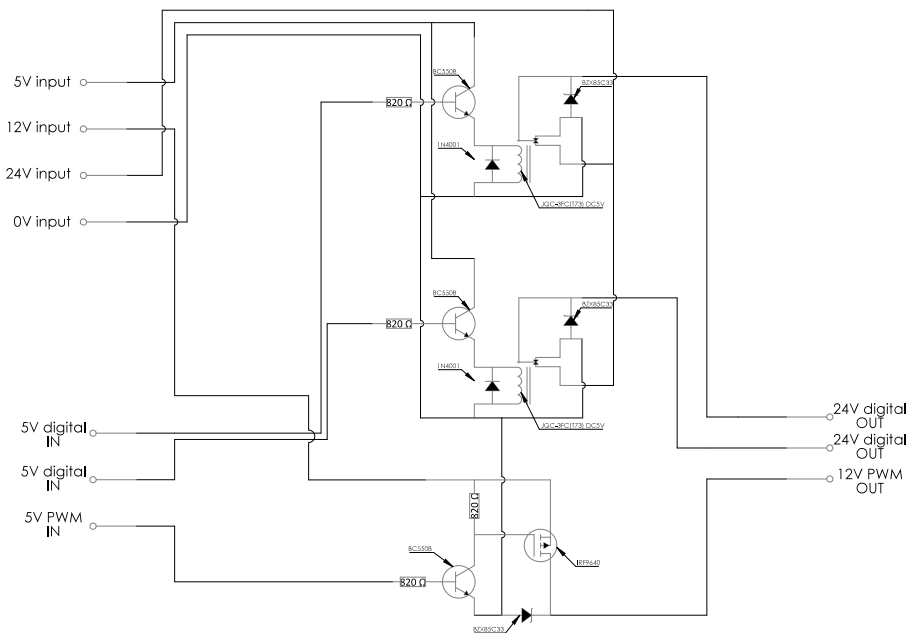


Figure 2.1: Driver circuit schematics

Converting the 5 VDC PWM to 12 VDC is not practical using a relay, as it needs a switching frequency of at least 50 Hz[3], which regular magnetic relays are not able to do. Instead, a PNP type MOSFET is used to switch a signal from a 12 VDC power line, as seen in Figure 2.1. This MOSFET will open at a  $V_{GS} = -10 V$  meaning the voltage at the gate of the MOSFET will have to be at least 10 V lower than the voltage at the source. To ensure the MOSFET can be fully closed, a driver circuit is implemented to control the gate voltage of the MOSFET in the same manner as is implemented for the relays. The way it works is that the voltage at the base of the transistor exceeds the  $V_{BE(on)}$  of 0.7 V the gate of the MOSFET is driven to the 0 V line, meaning that  $V_{GS} = 0 V_{gate} - 12 V_{source} = -12 V_{GS}$  this means the output is 12 V. Changing the voltage

at the base of the transistor to  $0\text{ V}$  means the voltage at the gate of the MOSFET is driven to  $12\text{ V}$  which leads to  $V_{GS} = 12\text{ V}_{gate} - 12\text{ V}_{source} = 0\text{ V}_{GS}$  so the output of the MOSFET is  $0\text{ V}$ .

The direction output for the thruster is generated by two digital signals from the Arduino, and as previously mentioned is generated by two relays which output either  $0\text{ VDC}$  or  $24\text{ VDC}$ . These relays are driven by similar transistors as used to control the MOSFET for the PWM output, as can be seen in Figure 2.1. Diodes are placed in parallel with the coils of the relays to ensure that the current generated by the coils when they are switched can be dissipated in these.

All outputs from the driver board is connected to the  $0\text{ V}$  line through Zener diodes to ensure if any voltage exceeds  $33\text{ VDC}$  this is shortened to the  $0\text{ V}$  line. The diodes are the same as the ones Saab Seaeye uses for their own controller[3]. These diodes will help protect the thruster against voltage spikes on the control lines. To protect the control unit,  $820\ \Omega$  resistors are placed in series with the control signals to ensure that the maximum current that can be drawn from each pin is limited. This limit can be calculated using ohm law.

$$i_{max} = \frac{5\text{ V}}{820\ \Omega} = 6.1\text{ mA}$$

The current drawn will be less than this in reality, but this will be the maximum current if a short circuit occurs, which could happen if a transistor burns out. A pull-up resistor is placed from the gate of the MOSFET to the  $12\text{ V}$  line, this limits the current that will run across the transistor when it is closed.

The maximum rated current of the transistor is  $100\text{ mA}$ , however, it is generally a good idea to stay well below this level, as the closer to this rating, the more heat is dissipated within the transistor. In addition the value of the resistor decides how fast the MOSFET is turned on as well, as it has a small capacitance, this has to be *charged* before the MOSFET will open fully. This means if the resistance is too large, the output will rise more slowly. Sizing this resistor will always be a trade-off between rise-time of the MOSFET and heat dissipated in the transistor. As the PWM signal has a relatively low frequency of  $\approx 50\text{ Hz}$  a similar resistance as used previously was chosen, and the current through the transistor can be calculated in the same manner as before, which yields a current of  $14.6\text{ mA}$  when the transistor shorts to ground.

### 2.2.3 Pod Assembly

For the thruster control, three different voltage levels are required.  $24\text{ VDC}$ ,  $12\text{ VDC}$  and  $5\text{ VDC}$ , from the ROV  $24\text{ VDC}$  is readily available on any auxiliary port which the controller is connected to[3]. To get  $12\text{ VDC}$  and  $5\text{ VDC}$  for the control circuits, two PSUs are used.

- Mean Well SD-15B-05 converts  $24\text{ VDC}$  to  $5\text{ VDC}$
- Mean Well SD-15B-12 converts  $24\text{ VDC}$  to  $12\text{ VDC}$

Both PSU's implement short circuit and over voltage- and load protection, they also have an EMI filter which protect against unwanted noise from the switching regulators inside the supplies[5]. The  $12\text{ VDC}$  and  $5\text{ VDC}$  line is connected with a  $0.5\text{ A}$  fuse, and the

24 VDC line is connected with a 1 A fuse to protect against short circuits.

As the Arduino will communicate to the topside through an RS-232 interface, a converter circuit must be created. This is because the serial interface of the Arduino is TTL based, which works on a different voltage level than specified by the RS-232 standard, the voltage levels for each interface can be seen in Table 2.1.

Interface	Logical 0	Logical 1
ATmega328P[6]	0 V to 0.9 V	4.2 V to 5 V
RS-232 [7]	3 V to 25 V	-3 V to -25 V

Table 2.1: Communication voltage levels

To use the RS-232 interface with the Arduino, a driver circuit that transforms the voltage potential between the ATmega328P level and the RS-232 level is necessary. A ready made component has been acquired which handles the conversion, it is manufactured by RSS-Systems and is able to convert 3 V or 5 V TTL or CMOS based signals into RS-232 level signals[8] and is powered by the 5 V line.

The pod is connected to the ROV by two cables, one connects to an auxiliary port, AUX3, which supplies 24 VDC power and the RS-232 communication interface. The other connection is the tooling power supply port, which has control signals and 500 VDC power supply for the thruster motor. The control signals of this remain disconnected.

The thruster is connected to the electronics pod through one cable, and the 500 VDC is connected directly to the tooling power supply, the signals generated by the Arduino and driver board replaces the control signals which would normally also come from the tooling power supply port.

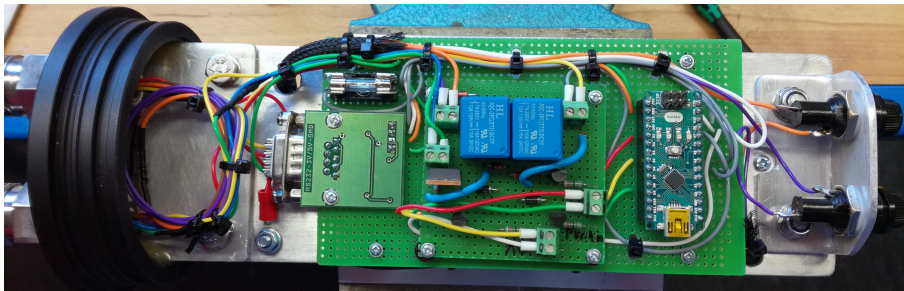


Figure 2.2: Assembled electronics pod

All electronics is combined in a water proof tube which is sealed with a O-ring. The assembled pod is shown in Figure 2.2, the Arduino, driver circuit and RS-232 to TTL converter are mounted on top of an aluminium plate, below the plate the two DC-DC PSUs are mounted.

## 2.3 ROV Dynamic Model

The goal of creating a dynamic model for the ROV is to correlate the force generated by thruster into the resultant movement of the ROV. As the objective of the project is to

implement pitch functionality, this dynamic model only needs to describe the movement in this direction.

The approach used to create the model is to identify the relevant forces acting upon the ROV. By doing this, the model will describe the steady state of the system, and the thrust needed to maintain any desired pitch angle can be calculated.

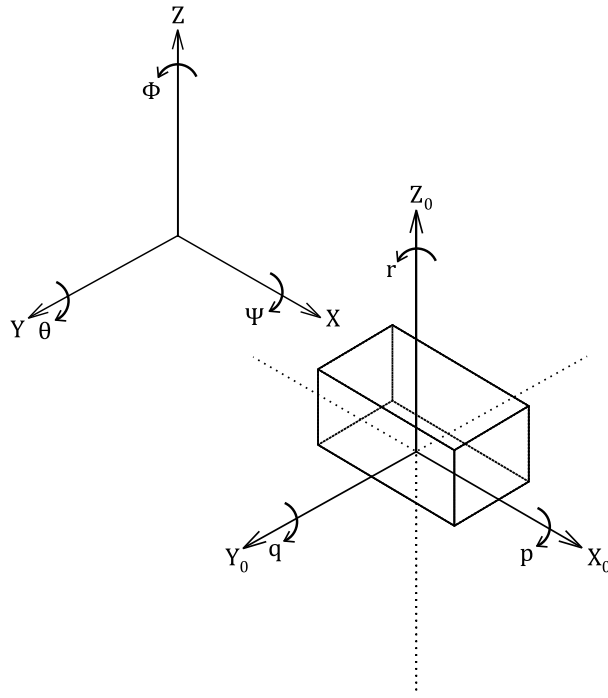


Figure 2.3: ROV reference frame

Figure 2.3 shows two reference frames which will be used, the first frame ( $X$ ,  $Y$  &  $Z$ ) is the global frame. This frame is used to describe the position and orientation of the ROV, relative to a fixed point and orientation.

$$\eta = \begin{bmatrix} \eta_1 & \eta_2 \end{bmatrix}^T = \begin{bmatrix} X & Y & Z & \phi & \theta & \psi \end{bmatrix}^T \quad (2.1)$$

Equation 2.1 is the combination of the two vectors that describe the position and heading of the ROV.  $\eta_1$  is the vector pointing to the centre of the ROV i.e. the position of it, while  $\eta_2$  is the vector that describes the heading of the ROV relative to  $X$ ,  $Y$  and  $Z$ .

The second frame in Figure 2.3 ( $X_0$ ,  $Y_0$  &  $Z_0$ ) is the local frame. This is located at the centre of the ROV, described by  $\eta_1$ , and is oriented with the  $X_0$  axis positioned along  $\eta_2$ . The local frame describes the linear and angular velocities of the ROV system.

$$v = \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T = \begin{bmatrix} X_0 & Y_0 & Z_0 & p & q & r \end{bmatrix}^T \quad (2.2)$$

As was done for the position and heading, the angular and linear velocities can be described by two vectors,  $v_1$  and  $v_2$ , this is shown in Equation 2.2. These velocities are combined in a vector describing all velocities of the system,  $v$ .

As the objective is to describe only the pitch of the ROV, neither the position vector  $\eta_1$  nor the linear velocity vector  $v_1$  is of interest. It is assumed that the controller in the ROV

will handle this. Likewise, for the vectors  $\eta_2$  &  $v_2$  only the terms relating to the pitch of the ROV.

$$\eta_{ROV} = \begin{bmatrix} 0 & \theta & 0 \end{bmatrix}^T \quad (2.3)$$

$$v_{ROV} = \begin{bmatrix} 0 & q & 0 \end{bmatrix}^T \quad (2.4)$$

All terms not relating to the pitch angle or velocity in Equation 2.3 and Equation 2.4 has been set to zero. From this, the vector  $\eta_{ROV}$  describes the pitch angle of the ROV relative to a fixed reference frame while the vector  $v_{ROV}$  describes the angular velocity of the pitch of the ROV.

In the following, the forces acting on the ROV will be described in order to include them in the model for the system.

### 2.3.1 Inertia Tensor

The equation used to describe the rotational motion of a body is similar to Newtons equation of motion, given in Equation 2.5 below[9]. For linear motion, Newtons equation identifies the fact that a given mass will have a *resistance* to being accelerated. For rotational motion, the inertia tensor describes the mass distribution in an object, which will describe how an object *resists* acceleration in varying degrees between different directions.

$$F = m \cdot a \quad (2.5)$$

Describing rotational motion requires, as mentioned, the use of the inertia tensor. The general formula for describing the rotational motion of rigid bodies can be seen in Equation 2.6. In this equation the inertia tensor ( $I$ ) is a mathematical description of the mass distribution in the object. In a similar manner as for linear motion, when a torque is applied to a rigid body, it will have a varying acceleration in different directions.

This variation arises due to the way the mass is distributed in the body. This is similar to what Newton tells of linear acceleration, namely that if the mass is increased it will take a greater force to achieve a given acceleration. Similarly if the mass is increased in a rigid body, it will take a greater torque to achieve a given acceleration. Additionally, if the mass is distributed non-uniformly, the acceleration will also occur non-uniformly.

$$\tau = I \cdot \frac{d\omega}{dt} \quad (2.6)$$

In Equation 2.6,  $\tau$  is a three-element vector describing the torque around the  $x$ ,  $y$  and  $z$  axis,  $\frac{d\omega}{dt}$  is the rotational acceleration around the same axes and finally  $I$  is the inertia tensor, which is a  $3 \times 3$  matrix. Equation 2.6 can be written explicitly, as shown in Equation 2.7.

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \cdot \begin{bmatrix} \frac{d\omega_x}{dt} \\ \frac{d\omega_y}{dt} \\ \frac{d\omega_z}{dt} \end{bmatrix} \quad (2.7)$$



If the object analysed is symmetric around all axes and the mass is distributed evenly, Equation 2.7 simplifies significantly as the off diagonal terms in the inertia tensor will be zero [10]. Determining the inertia tensor of the Cougar analytically is a very complex task, so to save time, some simplifications are made to get an approximation of it, which will hopefully be sufficiently accurate.

In order to get the approximation, there are two ways that are possible, one way is to treat the Cougar as a box, with equivalent dimensions and mass as the ROV. With this simplification, calculating the inertia tensor is simplified as only the diagonal elements will be non-zero.

Which means Equation 2.7 can be simplified to the following expression.

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \cdot \begin{bmatrix} d\omega_x \\ d\omega_y \\ d\omega_z \end{bmatrix} \quad (2.8)$$

Finding the diagonal elements of the inertial tensor can be done using the equations given below [10], where the  $x$ -axis is the length of the ROV, the  $y$ -axis is the width of the ROV and the  $z$ -axis is the height of the ROV.

$$I_{xx} = M \int_V (y^2 + z^2) dV \quad (2.9)$$

$$I_{yy} = M \int_V (x^2 + z^2) dV \quad (2.10)$$

$$I_{zz} = M \int_V (x^2 + y^2) dV \quad (2.11)$$

The above method will yield a crude result of the inertia tensor, and will not describe the fact that the mass is unevenly distributed inside the Cougar frame. To get a more accurate number, without overcomplicating it, a very simple model of the Cougar is drawn in SolidWorks, which can calculate the inertia tensor of a 3-D model.

The idea of the model is to split the Cougar into two boxes, one emulating the buoyancy foam at the top of the ROV, while the other box emulates the weight of the rest of the ROV. The length and width of the boxes will be equal to the cougar, and their height will be calculated as a function of the measured density of each component.

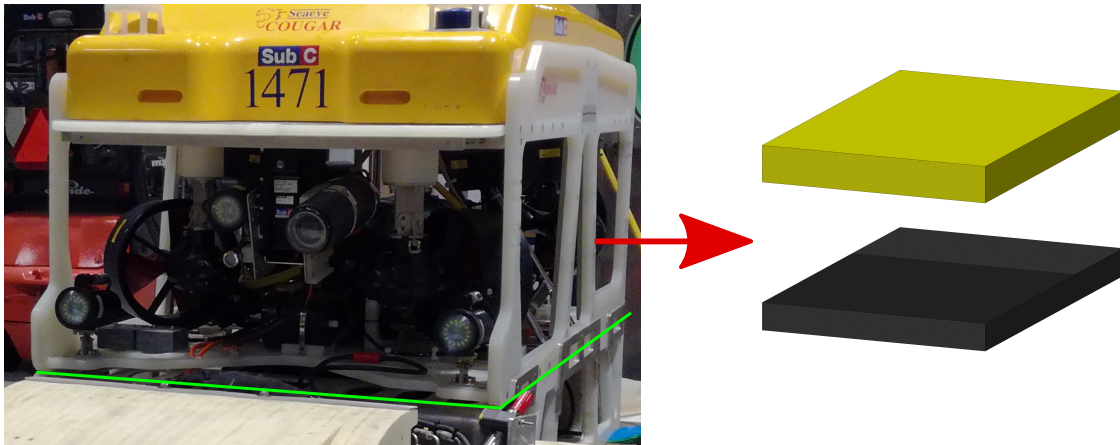


Figure 2.4: Cougar system simplification

Figure 2.4 shows the concept of the simplification. On the left is a picture of the Cougar. The green line marks the bottom of the ROV, below the line is a skid, this is unrelated to the current project and is not included in any calculations.

On the right hand side of Figure 2.4 is a picture of the simplified Cougar model. The idea behind the simplification is to calculate the density of the buoyancy foam and the rest of the ROV. Each are then modelled as a box, the distance from the top of the yellow block to the bottom of the grey block is equal to the height of the ROV. The calculation of densities and heights of the blocks are described in Appendix A.1.2-Foam volume & density.

As can be seen on Figure 2.4, most of the components of the ROV is positioned low in the frame, while the buoyancy foam is at the top. This is the reason the simplification should give a better approximation than treating the cougar as a single box. If greater accuracy is needed, it should be possible to split more components of the cougar up and positioning equivalent boxes in their relative positions of the frame.

Hopefully, the most simple model is sufficient to create a system model and implement a controller that is able to maintain a pitch angle. But this is an area where greater accuracy can be achieved if later experiments shows this is necessary.

Using SolidWorks it is possible to get the inertia tensors directly from the program, and using this function, the inertia tensors are given by the following.

$$I = \begin{bmatrix} 68.78 & 0 & 0 \\ 0 & 111.95 & 0 \\ 0 & 0 & 110.95 \end{bmatrix} [kg \cdot m^2] \quad (2.12)$$

By applying the inverse of the inertia tensor on both sides of Equation 2.8, the angular acceleration can be evaluated as shown in Equation 2.13.

$$\begin{bmatrix} d\omega_x \\ d\omega_y \\ d\omega_z \end{bmatrix} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \begin{bmatrix} 0.014539 & 0 & 0 \\ 0 & 0.008933 & 0 \\ 0 & 0 & 0.009013 \end{bmatrix} [kg \cdot m^2]^{-1} \quad (2.13)$$

Equation 2.13 is the result of multiplying the inverse of the inertia tensor on both sides of the equation. Using this equation, it is possible to calculate the resulting angular acceleration on the ROV due to given torque.

### 2.3.2 ROV Torque

Due to the way the Cougar is built, two forces generate a torque when the cougar is pitched. The buoyancy foam pulls the ROV upwards while the rest of the Cougar pulls downwards, when the ROV is level, these forces cancel and the ROV has a stable orientation. However, if the ROV is pitched (or rolled for that matter) these two forces, will generate a torque upon the ROV which will cause it to swing back towards a steady state. This means, even if the ROV is perfectly weighted to be of equal mass as the water it displaces, it has a pitch and roll angle to which it will return.

This implies that the ROV system is a stable system, and when undisturbed will maintain the same pitch and roll angle. Further, if it is subjected to a disturbance it will also return to this state. This also means when the thruster is implemented to maintain a pitch angle, it will have to run continuously as it will have to work against this torque.

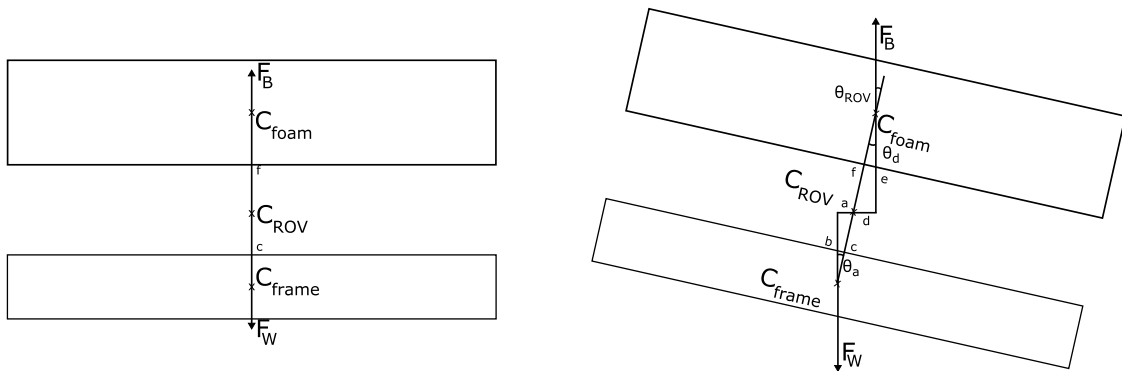


Figure 2.5: ROV moment of torque

Figure 2.5 illustrates the setup of the ROV, the top block being the buoyancy foam where  $C_{foam}$  is the centre of mass of the foam block. The bottom block emulates the weight of the rest of the ROV where  $C_{frame}$  is the centre of mass for this.  $C_{ROV}$  is the centre of mass for the whole system.

When the ROV is at balance, as seen left in Figure 2.5, all centre of mass' is aligned on a vertical axis, and the buoyancy force  $F_B$  is equal to the force exerted by the weight of the rest of the ROV, denoted  $F_W$ .

When the ROV is at an angle, the forces  $F_B$  and  $F_W$  is still of an equal magnitude, however, they form a torque acting upon the ROV frame. The concept is illustrated on the right in Figure 2.5. Two torques are generated when the ROV is pitched. The first torque ( $\tau_d$ ) is a result of the buoyancy of  $F_B$ , the second torque ( $\tau_a$ ) is a result of the weight  $F_W$ . The magnitude of the torques depend on the angle of the ROV, as they are a function of the horizontal distance between the centre of gravity of the ROV and the centre of the acting force.

$$\tau = r \cdot F \cdot \sin(\phi) \quad (2.14)$$

Generally, torque is calculate by Equation 2.14 [11]. Where  $F$  is the force applied,  $r$  is the distance from the axis of rotation to the point where force is applied and  $\phi$  is the angle between the vector from the point of force application to centre of rotation and the vector of the force itself. In Figure 2.5 the angle  $\phi$  for the two torque cases is between sides  $b$  and  $c$  for the torque generated by  $F_W$  and between sides  $e$  and  $f$  for the torque generated by  $F_B$ . Based on this, two functions can be formulated which describe the torque as a function of the pitch angle of the ROV.

$$\tau_B(\theta_{ROV}) = d \cdot F_B \cdot \sin(\theta_{ROV}) \quad (2.15)$$

$$\tau_W(\theta_{ROV}) = a \cdot F_W \cdot \sin(\theta_{ROV}) \quad (2.16)$$

In Equation 2.15 and Equation 2.16 the distance  $d$  and  $a$  both change as a function to the angle  $\theta_{ROV}$ . Using trigonometric calculus (See figure 2.5) the functions can be rewritten as below.

$$\tau_B(\theta_{ROV}) = \sin(\theta_{ROV}) \cdot f \cdot F_B \cdot \sin(\theta_{ROV}) \quad (2.17)$$

$$\tau_W(\theta_{ROV}) = \sin(\theta_{ROV}) \cdot c \cdot F_W \cdot \sin(\theta_{ROV}) \quad (2.18)$$

In Equation 2.17 and Equation 2.18 both  $f$  and  $c$  are constant (which can be seen in Figure 2.5). For the model, they are approximated using the simplified Cougar model described in Appendix A.1. By knowing the location of the centre of mass' for each block and the whole Cougar system,  $f$  and  $c$  is given by the distance between each.

In Section A.1.3-Centre of Mass' the location of all the mass centres are calculated, relative to the same point they are given by the following.

$$\begin{aligned} C_{frame}^x &= 0.748 \text{ m} & C_{frame}^y &= 0.059 \text{ m} \\ C_{foam}^x &= 0.748 \text{ m} & C_{foam}^y &= 0.711 \text{ m} \\ C_{ROV}^x &= 0.748 \text{ m} & C_{ROV}^y &= 0.385 \text{ m} \end{aligned}$$

From this it is possible to calculate the lengths  $f$  and  $c$  in Figure 2.5 by the following.

$$c = C_{ROV}^y - C_{frame}^y \Rightarrow c = 0.326 \text{ m} \quad (2.19)$$

$$f = C_{foam}^y - C_{ROV}^y \Rightarrow f = 0.326 \text{ m} \quad (2.20)$$

Inserting this into Equations 2.17 & 2.18 the last unknown is the forces acting upon each. First, calculating the buoyancy of the foam block, requires the use of Archimedes' principle [12], as shown below.

$$F_B = \rho_{water} \cdot V_{foam} \cdot g \quad (2.21)$$

Using the same density of water and volume of foam as found in Appendix A.1.2 and the gravitational acceleration to be  $9.82 \text{ m/s}^2$  [13].

$$F_B^{foam} = 998.2 \text{ kg/m}^3 \cdot 0.2366 \text{ m}^3 \cdot 9.82 \text{ m/s}^2 \Rightarrow F_B^{foam} = 2319.23 \text{ N} \quad (2.22)$$

In the same manner, the buoyancy of the frame can be calculated using the volume of the frame that was found in Appendix A.1.1.

$$F_B^{frame} = 998.2 \text{ kg/m}^3 \cdot 0.177 \text{ m}^3 \cdot 9.82 \text{ m/s}^2 \Rightarrow F_B^{frame} = 1735.01 \text{ N} \quad (2.23)$$

Next it is necessary to calculate the gravitational forces acting upon both the foam and frame. This can be calculated using Newtons second law [9], see Equation 2.5.

Applying the equation, the gravitational force acting on the foam and frame can be calculated, using the weight of the frame and the foam as found in Appendix A.1.1 and Appendix A.1.2. For the mass of the frame, the weight of the lead ballast is added. After this, the resultant forces  $F_B$  and  $F_W$  (See figure 2.5) is equal to the difference between the buoyancy and the gravitational force.

$$F_g^{frame} = (237 \text{ kg} + 78 \text{ kg}) 9.82 \text{ m/s}^2 = 3093.3 \text{ N} \quad (2.24)$$

$$F_g^{foam} = 105 \text{ kg} \cdot 9.82 \text{ m/s}^2 = 1031.1 \text{ N} \quad (2.25)$$

Based on this, the forces  $F_B$  &  $F_W$  in Equation 2.17 and Equation 2.18 can be calculated by the difference between the buoyancy and weight.

$$F_B = F_B^{foam} - F_g^{foam} \Rightarrow F_B = 1288.13 \text{ N} \quad (2.26)$$

$$F_W = F_B^{frame} - F_g^{frame} \Rightarrow F_W = -1358.29 \text{ N} \quad (2.27)$$

As the ROV is balanced, the forces calculated in Equation 2.26 and Equation 2.27 should be of equal magnitude but in opposite directions. There is a difference however of  $\approx 70 \text{ N}$ . This is a relatively small difference, and there can be several reasons for the deviation. First of all, the ROV will never be perfectly balanced. Additionally, the scale used to measure the ROV weight only measures to the nearest  $\text{kg}$  which causes some deviation of results. Keeping all this in mind, the difference of  $\approx 70 \text{ N}$  equates to  $\approx 7 \text{ kg}$  which seems very reasonable. Inserting the results into the equation for torque, the torque of the ROV is given by the following.

$$\tau_B(\theta_{ROV}) = 419.93 \text{ N m} \cdot \sin^2(\theta_{ROV}) \quad (2.28)$$

$$\tau_W(\theta_{ROV}) = -442.80 \text{ N m} \cdot \sin^2(\theta_{ROV}) \quad (2.29)$$

### 2.3.3 Drag

Two effects causes drag on the ROV. One is waves and currents, these create a drag force upon the ROV which *pushes* the ROV. This effect is ignored because it is assumed it will mainly affect the ROVs positioning, which the pitch controller will not handle.

The second cause of drag occurs when the ROV accelerates through the water. This results in a drag force working against the movement. Describing the drag of the ROV, or any other arbitrary object, is a complex challenge, depending upon the shape, area and the direction of flow relative to the shape [14].

$$F_D = \frac{1}{2}\rho C_{ds}Su^2 \quad (2.30)$$

Equation 2.30 [15] is the general formula for calculating the drag force of an object. In the expression  $S$  is the area of the object,  $u$  is the fluid velocity and  $C_{ds}$  is the drag coefficient.

The drag coefficient is a lumped parameter, which combines several parameters into a single number. Calculating the coefficient for a complex shape, for example an ROV, is very difficult and is often identified experimentally rather than analytically [14].

To simplify it the calculation, the ROV will be analysed as a plate being pitched around a centre pivot point, this makes it much easier as the drag coefficient of *simple* shapes is well documented. To get the coefficient of a plate, recommendations by DNV are used. Using these recommendations, the drag coefficient is dependant upon the ratio between the length and width of the plate [15]. The table used to find the coefficient for the plate has been included in Appendix B-DNV Drag Coefficients.

By Equation 2.30, the drag force also depends on the velocity of the fluid. As previously stated wave and current induced drag will be ignored. This means, the fluid is assumed to be stationary, and the velocity of the fluid will be equal to the angular velocity of the ROV as it pitches.

Therefore, the fluid velocity changes relative to the centre of rotation. So to get an approximation for the drag of the ROV, the rectangular plate emulating the ROV will be split into two half plates, with the velocity changing as a function of the distance from the centre of rotation.

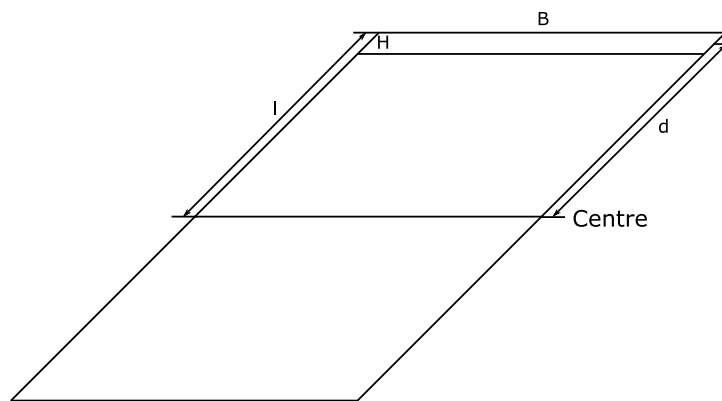


Figure 2.6: Drag of the Cougar

Figure 2.6 illustrates the method in which the drag of the Cougar will be analysed. The plate will have the same total length and width as the Cougar. The drag will be described as a sum of sections of the plate with the dimensions  $B \times H$ . The fluid velocity will then be calculated from the angular velocity ( $\omega$ ) multiplied with the distance from the centre of the ROV to the plate section.

This means, that by decreasing the length  $H$  of each plate section, the approximation for the drag should be improved, by getting a more accurate expression of the fluid velocity.

By applying the recommendation established by DNV in the document DNV-RP-H103 [15] to calculate the drag coefficient and drag force (See Appendix B-DNV Drag Coefficients) the drag coefficient,  $C_{Ds}$ , is based on the ratio between the width and length of the plate. From this, the following relation is true.

$$\lim_{H \rightarrow 0} \frac{B}{H} = \infty \Rightarrow C_{ds} = 1.90 \quad (2.31)$$

Equation 2.31 will be true as long as  $H \ll B$ . This means Equation 2.30 can be rewritten to the expression shown in shown in the following expression.

$$F_D = 1/2\rho C_{ds}(B \cdot H)(\omega \cdot d)^2 \quad (2.32)$$

Splitting the plate into a number of smaller sections Equation 2.32 can be formulated as a sum of the smaller sections as in Equation 2.33 below.

$$F_D = 1/2\rho C_{ds}(B \cdot H) \sum_{i=1}^n (\omega \cdot d_i)^2 \quad (2.33)$$

The equation shown above yields the force resulting from the drag of the ROV as it pitches. As the model of the system is based on a sum of torques yielding an acceleration, the drag must be converted to an equivalent torque. Calculating the torque generated by a force is done by the expression in Equation 2.14. As the drag caused by the pitching of the ROV is perpendicular to the ROV, the expression can be simplified to Equation 2.34 because the angle  $\phi$  is  $90^\circ$  constantly.

$$\tau = r \cdot F \quad (2.34)$$

By applying the torque equation to Equation 2.33 the drag of the cougar as a torque can be calculated as shown below in Equation 2.35.

$$\tau_D(\omega) = 1/2\rho C_{ds}(B \cdot H) \sum_{i=1}^n ((\omega \cdot d_i)^2 \cdot d_i) \quad (2.35)$$

### 2.3.4 System Model

The model of the system is based on a balance of forces, namely the torque generated by the Cougar ROV as it is pitched. This identifies the torque which the thruster needs to generate in order to maintain a desired pitch angle. With this, the balance can be described by two equations. The torque generated by the ROV at a certain angle, and the torque generated by the thruster to achieve a given angle.

$$\tau_{ROV}(\theta_0) = \tau_B(\theta_0) + \tau_W(\theta_0) \quad (2.36)$$

Equation 2.36 describes the torque generated by the weight and buoyancy caused by the ROV itself at a given angle. The thruster needs to generate the torque that this equation describes in the opposite direction. This is shown in Equation 2.37.

$$\tau_{SM-7}(\theta_{set}) = (\tau_B(\theta_{set}) + \tau_W(\theta_{set})) \cdot -1 \quad (2.37)$$

When the ROV is stationary, Equation 2.36 describes the torque of the ROV. However, when the ROV is accelerated, a drag will arise which works against the movement. This drag torque is a function of the angular velocity of the ROV.

$$\tau_D(\omega) \quad (2.38)$$

By adding Equation 2.38 and Equation 2.37 to Equation 2.36 the torque for the whole system is given by the following.

$$\tau_{net} = \tau_{ROV}(\theta_0) + \tau_{SM-7}(\theta_{set}) + \tau_D(\omega) \quad (2.39)$$

By basing the model on a torque balance, it is possible to identify the thrust needed to maintain a given angle for the ROV. The drag has no effect on the torque required to maintain a pitching angle, however, it does limit the velocity which the system can achieve as it will have an increasing effect as the velocity increases.

Finding the angular acceleration caused by a given torque can be done using Equation 2.6. By inserting the torque of the system, Equation 2.39 into that equation, the instantaneous acceleration to a given torque can be calculated, as in Equation 2.40.

$$\tau_{net}I^{-1} = \frac{d\omega}{dt} \quad (2.40)$$

By integration of Equation 2.40, the angular velocity can be calculated.

$$\omega = I^{-1} \int (\tau_{net})dt \quad (2.41)$$

As the torque from the ROV changes as the ROV rotates, the angle  $\theta_0$  changes, the calculation of acceleration and velocity is only accurate for very small time periods. By knowing the angular velocity, it is possible to calculate the angle by multiplying the angular velocity with the elapsed time.

$$\theta = \omega \cdot \Delta t \quad (2.42)$$

By recalculating the system torque, acceleration, velocity and angle for very small time periods, it is possible to approximate the system behaviour as it pitches. By including the previously calculated velocity in Equation 2.41 and the previously calculated angle in Equation 2.42, the system can be described as below.

$$\omega_1 = I^{-1} \int_0^{\Delta t} (\tau_{ROV}(\theta_0) + \tau_{SM-7}(\theta_{set}) + \tau_D(\omega))dt + \omega_0 \quad (2.43)$$

$$\theta_1 = \omega_1 \cdot \Delta t + \theta_0 \quad (2.44)$$

These two equation can be inserted into the vectors describing the ROV angle (Equation 2.3) and ROV angular velocity (Equation 2.4).

$$\eta_{ROV} = \begin{bmatrix} 0 \\ \omega_1 \cdot \Delta t + \theta_0 \\ 0 \end{bmatrix} \quad (2.45)$$



$$v_{ROV} = \begin{bmatrix} 0 \\ I^{-1} \int_0^{\Delta t} (\tau_{ROV}(\theta_0) + \tau_{SM-7}(\theta_{set}) + \tau_D(\omega)) dt + \omega_0 \\ 0 \end{bmatrix} \quad (2.46)$$

## 2.4 Thruster Test

Creating the model of the system also requires a model which can correlate the output to the thruster and the resulting thrust generated by the *SM-7* thruster. There are two primary methods which can be used to get a model of the thruster, *white box* or *black box* modelling.

In *white box* modelling, the whole system is described mathematically, in case of an electric thruster it would mean describing the DC motor mathematically first, and after this is achieved, describe the interface from the revolutions of the DC motor to water moved which generates thrust.

In *black box* modelling, the thruster is viewed as an unknown system, and through experiments it is aimed to get a describing function. With the thruster for example, an experiment can be setup, where the thrust generated is plotted against the PWM signal input to the thruster. From these data points a function can be sought that approximates the result of any given duty cycle.

To create a model of the thruster *black box* modelling is the chosen method, this is done for two reasons. First of all several important parameters needed to model the thruster mathematically is unknown and several experiments would be needed in order to get a model of the thruster. This is further complicated by the fact that the *SM-7* is not rated to operate above water for extended periods of time. Secondly, the hope is that, by using the *black box* method, creating an experiment that compares PWM input to thrust output, a model can be acquired faster to save time.

In the following section the test to acquire the input output data needed to model the thruster will be described. The setup to perform the test will be described and the preliminary results will be shown. A simple function to calculate the required PWM to get a given torque will be done using excel, by correlating the PWM input as a function of the output thrust. The data acquired in the following section will then be used in the solution to model the thruster using the Matlab toolbox *System Identification*.

### 2.4.1 Modelling Method

In order to get an accurate model of the thruster, a series of experiments must be performed. The ultimate goal of these is to get a transfer function which describes the thruster behaviour. The idea is to identify the relation between the thrust generated by the *SM-7* as a function of the PWM duty cycle being input. Figure 2.7 illustrates the concept.

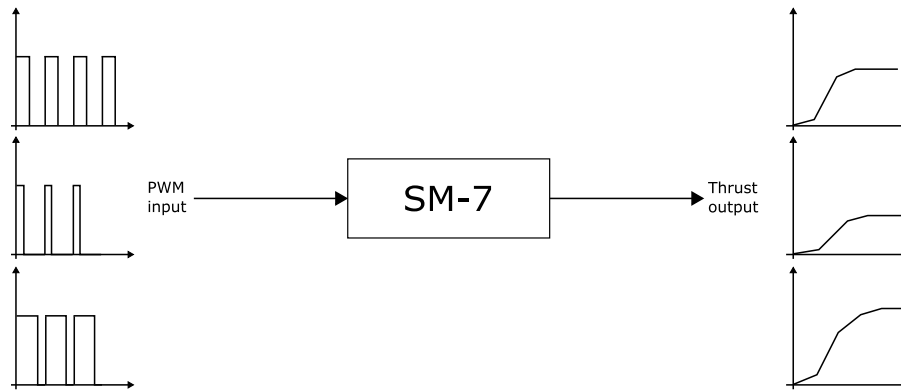


Figure 2.7: System identification method of *SM-7* thruster

By measuring the force generated by the thruster for different PWM values, see Figure 2.7, the relation between the input and output can be analysed. The thruster is analysed as a *SISO* system (Single-Input and Single-Output). By performing a set of experiments, inputting PWM signals of varying duty cycles and measuring the generated force, it will be possible to use this data to estimate a model of the *SM-7*.

Correlating the generated thrust to the input is relatively simple if the steady state behaviour is the only information needed. For the thruster, the goal is to describe the transient behaviour which makes the issue slightly more complex. The main issue is ensuring that the sampling of the force generated by the *SM-7* occurs at a sufficiently high rate. Finding the minimum sampling rate required in order to analyse the transient response is difficult without knowing the system, as it is dependent upon the *speed* of the system.

To monitor the thrust during the test a load cell was used which had a sampling rate of 10 samples per second. The reason for using the cell was that it came calibrated with a controller, so the output could be read on analog 4 – 20 *mA* output channel. Secondly it was hardware which had been used on a previous experiment so there were some older written code that could be reused.

#### 2.4.2 *SM-7* Test

Collecting the data required for system identification as described in Section 2.4.1-Modelling Method, a test stand is necessary. This will hold the *SM-7* while different PWM duty cycles are output, meanwhile a load cell will monitor the generated thrust. Figure 2.8 shows the conceptual idea of the test bench.

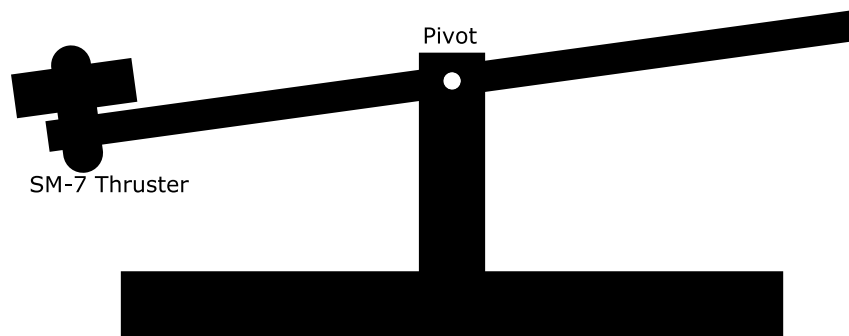


Figure 2.8: SM-7 test bench concept

The functionality of the test bench is two-fold, first of all it will be used to collect the data necessary to create a model of the thruster. Secondly, the setup will also be used to emulate the ROV when it comes to testing the functionality of the full control system. This is done to ensure, that if the controller performs poorly, it will not harm the Cougar ROV.

The setup illustrated in Figure 2.8 consists of a lever of the same length as the ROV. The lever is mounted to a base plate via a pivot point in a vertical beam. This point acts as the centre of mass of the ROV, and is the rotational joint which the thruster pitches around.

For the thruster test, the opposite end of the lever, where the thruster is not mounted, is tied down with a load cell in between. This means when the thruster generates upwards thrust, the rope tied to the lever will be extended, and the load cell will give a reading of the force. By sampling the load cell it is possible to generate datasets that contain the force generated as a function of time, and these can then be compared for different PWM duty cycles.

To perform the test, a simple java program is written which implements serial communication with two Arduinos. The first Arduino is used to sample the load cell, monitoring the current thrust level, and relays the readings to the test PC. The second Arduino is the one which is implemented to control the thruster, as described in Section 2.2. The program allows the user to both set direction and *PWM* value for the thruster, while the program logs the readings from the load cell alongside the current output for the thruster and a time stamp.

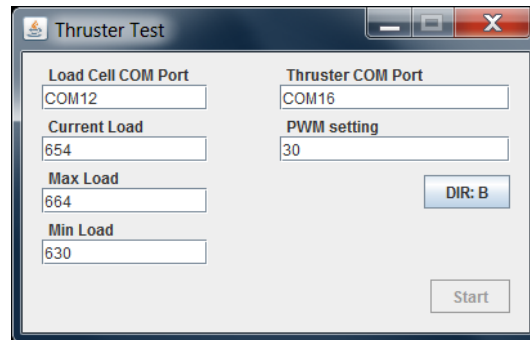


Figure 2.9: Thruster test control program

The program for the test incorporates a simple UI, shown in Figure 2.9. This enables entering of the COM-ports which the Arduinos are connected to. When the "start"-button is pressed, the program opens and maintains the two ports. Current, maximum and minimum readings of the load cell is presented live. This is the raw data, meaning it is the ADC reading, a number between 0-1023. To control the thruster, a text field allows the user to enter a number between 0 and 100, equivalent to the wanted PWM value. Pressing the *enter*-key sends the set value to the thruster controller. If an unrecognised character is entered, the value is set to 0 and this is transmitted. Any value outside the range will be set to the nearest number inside the range. Finally, the button "DIR:B" in Figure 2.9 sets the direction of the thruster, if pressed it will transmit a command to switch the direction to forwards, and the button label changes to "DIR:F".

After the "start" button is pressed, the program logs all the readings from the load cell along with the output states to the thruster controller. The log is written in a text file and the data entries are structured as below.

*Time Loadcell<sub>RAW</sub> Loadcell<sub>KG</sub> PWM<sub>%</sub> Direction*

Entries are created every time a reading from the load cell is received. The microcontroller monitoring the load cell is setup to create approximately 142 samples per second, the code can be seen in Appendix C.2. This is a theoretical sampling rate based on the average time for the *ATMega328P* microcontroller to perform an *ADC* reading [6], and as such it is slightly lower in reality. However, as the load cell used has a sampling rate of 10 samples per second, it is more than sufficient.



Figure 2.10: Thruster test stand

Figure 2.10 shows the test stand with the thruster mounted on the right. On the left, a chain is visible, this is connected to a load cell and then to a fixed point. The whole setup is lowered into a tank with water and the thruster is tested at several different PWM values. To get both directions of the thruster modelled, it is pulled out of the tank, and the thruster is turned around, and the test is redone with the new direction of the thruster.

PWM	Steady Load (F / B)
20 %	$\approx 1 \text{ kg} / \approx 1 \text{ kg}$
30 %	$\approx 5 \text{ kg} / \approx 4 \text{ kg}$
40 %	$\approx 15 \text{ kg} / \approx 9 \text{ kg}$
50 %	$\approx 20 \text{ kg} / \approx 14 \text{ kg}$
60 %	$\approx 27 \text{ kg} / \approx 19 \text{ kg}$
70 %	$\approx 34 \text{ kg} / \approx 30 \text{ kg}$
80 %	$\approx 47 \text{ kg} / \approx 46 \text{ kg}$
90 %	$\approx 57 \text{ kg} / \approx 60 \text{ kg}$
100 %	$\approx 61 \text{ kg} / \approx 61 \text{ kg}$

**Table 2.2:** Preliminary results from test

Table 2.2 contains the preliminary results from the test. These results are used to create a simple correlation between torque and PWM value. As the measurement from the test is the force generated, to get the torque the thruster generates depends upon the distance from the thruster to the centre of rotation, that is, the ROV's centre of mass along with the angle of the force [16]. Due to the simplification of the Cougar model (described in Section 2.3.1), the centre of mass is the centre of the ROV. Therefore, the distance from the thruster to the centre of rotation is half the Cougar length (1515 mm [2]) plus the distance from the thruster bracket to the centre of the thruster (Measured to be  $\approx 9 \text{ cm}$ ).

$$\tau = F \cdot r \Rightarrow \tau = F \cdot 0.7665 \text{ m} \quad (2.47)$$

Assuming the thruster is mounted vertically on the Cougar frame, the torque can be calculated as shown in Equation 2.47. By using the steady state force values of the thruster from Table 2.2 and calculating the resulting torque using Equation 2.47, a correlation of PWM values as a function of torque is created. Figure 2.11 below shows this, for both forwards and backwards directions.

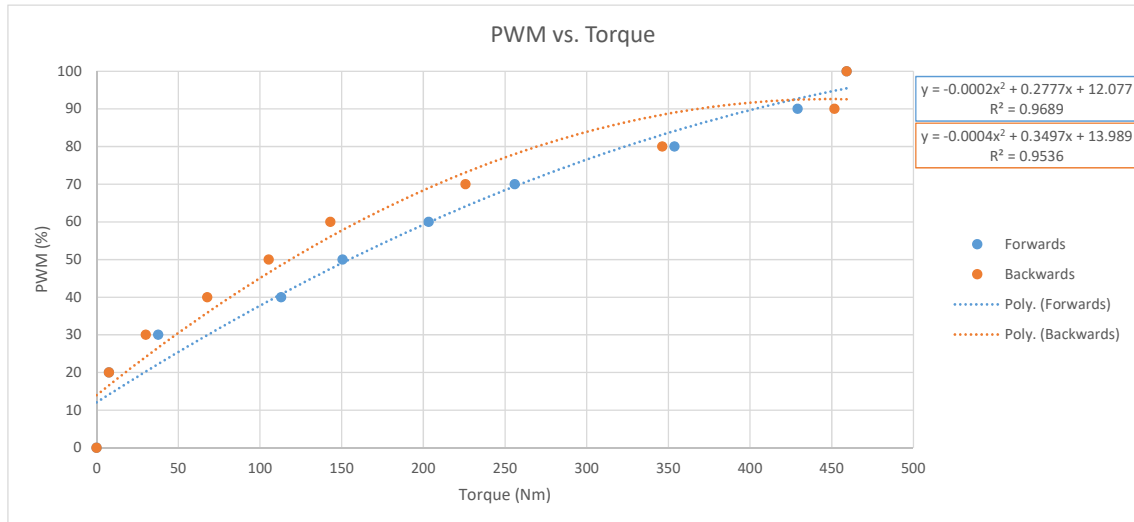


Figure 2.11: Thruster torque vs. PWM value

The correlation between the output PWM value and the resulting torque, plotted in Figure 2.11 will be used later to for the control logic to *choose* PWM values.

## 2.5 Conclusion

In the preceding chapter, the project was identified, to implement pitch functionality upon a Cougar ROV which has no facilities allowing pitch, roll or yaw movement. In Section 2.1 the Cougar ROV was described along with the available interfaces. Here an RS – 232 interface was identified as the best candidate for communication.

As the thruster used was one from the spares kit of the Cougar, the SM-7 , some electronics had to be created that allows control of the thruster. In Section 2.2 the required electronics was identified and created. This led to a pod containing the required electronics to control the speed and direction of the thruster along with circuits the allow an Arduino microcontroller to communicate upon an RS – 232 interface.

To implement a controller, a simple model describing the Cougar ROV pitch movement were found in Section 2.3. Here the relevant physical forces acting upon the ROV was identified and a mathematical expression was derived which describes the pitch movement of the ROV due to a torque.

Finally, the a mathematical description of the thruster behaviour is needed for the system simulation and control setup, and in Section 2.4, it was decided to use a methodology called *black box* modelling. From this, a series of experiments were performed in order to correlate the force generation of the thruster due to different inputs. In the following chapter, the thruster data will be used to identify a model of the thruster. Finally, a model for the ROV and thruster system will be created and simulated and a controller will be designed.

## Chapter 3

# Solution

In the following chapter, the pitch control for the ROV will be analysed and implemented. Initially, the data acquired in the thruster test in Section 2.4 will be used to find a transfer function for the *SM-7* thruster. With this and the dynamic model of the ROV found in Section 2.3, the system can be simulated and analysed. Finally, a controller is created that will aim to increase the system response time while avoiding oscillations and potential steady state offsets of the angle.

### 3.1 *SM-7* Model

As previously described, black box modelling methods will be used to model the *SM-7* thruster. In the following section the data acquired in Section 2.4, will be used to identify a system model of the *SM-7*.

To achieve this, the *System Identification* toolbox in Matlab is used. This implements several algorithms that can assist in modelling of systems based on input and output data.

Before starting the system identification process, the data from the test is analysed and several steps are taken to preprocess the data. This is done in order to ensure the model is as accurate as possible with the data acquired.

#### 3.1.1 Data Preprocessing

Figure 3.1 below show the data from the experiment with the thruster pushing forward. The output, thrust, has been converted to the equivalent torque it would create while mounted on the ROV frame. In this data set, the data from all the experiments is combined into a single set.

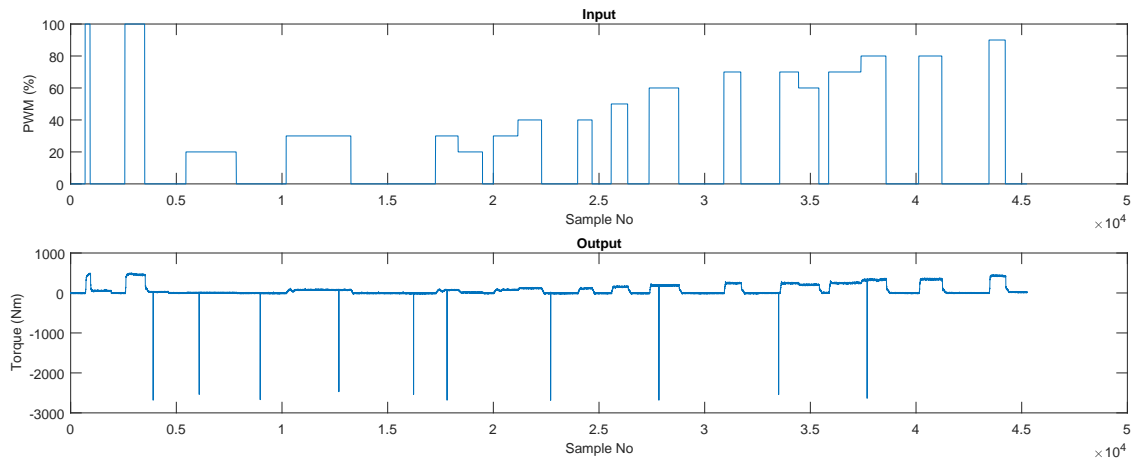


Figure 3.1: Input/output data from thruster test

Looking at Figure 3.1 it is clear that there are a few data points that lie far from the mean values (below  $-2000\text{ Nm}$ ). As this is way outside the range of what the thruster can generate, and in the wrong direction considering the test setup, these points must be either noise or communication errors. For this reason, these outliers are removed before starting system identification. Figure 3.2 shows the result of removing the outliers, the input/output relation of the thruster is more clear.

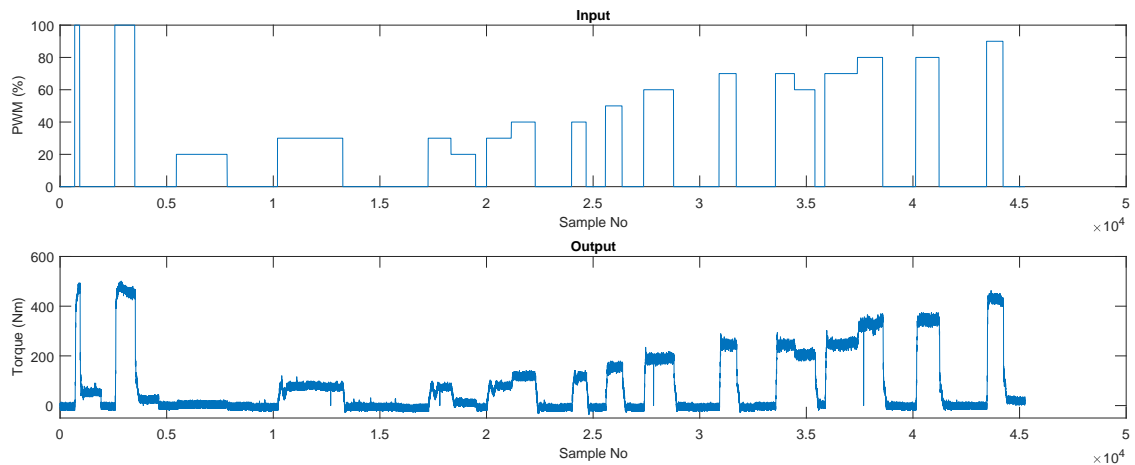


Figure 3.2: Input/output data from thruster test with outliers removed

Looking at the output data in Figure 3.2, it is clear that there is a significant amount of noise in the data. To improve the system identification, this data can be filtered, so that high frequent noise is removed. This has the downside of slightly skewing the data, and there is the risk if the data is filtered too severely, the transient behaviour of the thruster will be lost.

Two methods that can be applied to remove the noise is; create a low pass filter or implemented a moving average filter. The moving average is the most simple, while the low pass is a bit more complex but can have a better attenuation of unwanted frequencies.

When implementing a low pass filter (or any filter for that matter), analysing the data



for the frequencies that are present in the signal is necessary, as this will help establish the pass and stop bands. To analyse the signal, a spectrum analysis is performed on the output data. This is based on the *Fast Fourier Transform* (FFT), using this, the frequency content of the output data [17] can be plotted. When performing frequency analysis of signals, there are two factors to be wary of. The first factor is the sampling frequency, as it is not possible to identify signals of a frequency higher than  $1/2$  sampling rate, which is referred to as the Nyquist Frequency [18]. From this arise the second issue, which is aliasing. Aliasing is the tendency of signals with frequencies above the Nyquist Frequency to appear as a lower frequent signal when being sampled [19]. Avoiding these issues is challenging. One generally ensures sampling frequencies of a sufficiently high rate compared to the signals being worked with. However, as described in Section 2.4, the load cell used to generate the output data has a sampling rate of 10 samples per second, which means while getting the transient behaviour of the thruster is difficult, it is also likely that high frequent noise will appear as lower frequencies.

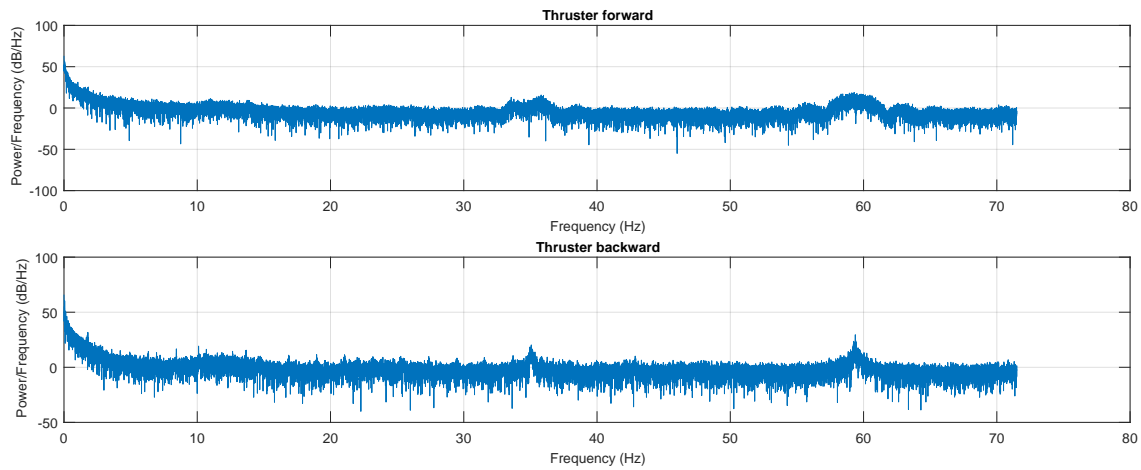


Figure 3.3: Frequency density of output data

Figure 3.3 shows the spectrum analysis of the output data of the thruster in both directions. This shows a clear dominance of frequencies below  $\approx 5$  Hz. For both output data sets it also looks as if there are two frequency bands which also has some power,  $\approx 33$  Hz to 38 Hz and  $\approx 57$  Hz to 63 Hz. With this in mind, the data can be passed through a low pass filter, which attenuates frequencies above the stop band of the filter.

Using the Filter Design toolbox in Matlab, a low pass filter can be created relatively easy. As the previously described frequency bands is most likely noise, a low pass filter is created with the following design parameters.

- $F_{pass} = 5$  Hz and  $F_{stop} = 25$  Hz
- $F_s = 150$  Hz and  $A_{stop} = 60$
- Design method: FIR equiripple

The above settings result in a filter of 14<sup>th</sup> order. It has a linear group delay of 7 samples on the output of the filter. One advantage of the FIR filter is that it has a common group delay, meaning that all frequencies get delayed an equal amount of samples.

Alternatively to using the low pass filter, the data can be passed through a moving average filter. The function of this is similar to the low pass filter in that it will attenuate high frequent signals while affecting low frequent ones less. It works by computing the average of the the current sample summed with a number of previous samples. This means that a 10 samples moving average, will compute the average of the current sample and the 9 previous samples and pass the result as the output, Equation 3.1 below shows the expression for the filter [20].

$$y(i) = \frac{1}{N} \sum_{j=0}^{N-1} x[i + j] \quad (3.1)$$

By increasing the factor  $N$  in the moving average filter, more samples are used to calculate the average and the amplitude decreases. By increasing the samples in the the filter however, the step response also get affected more severely. Finding the exact number of samples to use for the moving average filter is largely a trial-and-error process. For the data set, a 50 step moving average filter does a good job of reducing the noise while maintaining the transient response of the data.

To evaluate which filter to use on the data, it is passed through both the low pass filter and the moving average filter. Figure 3.4 below show the raw data, the data through the low pass and the data through the moving average.

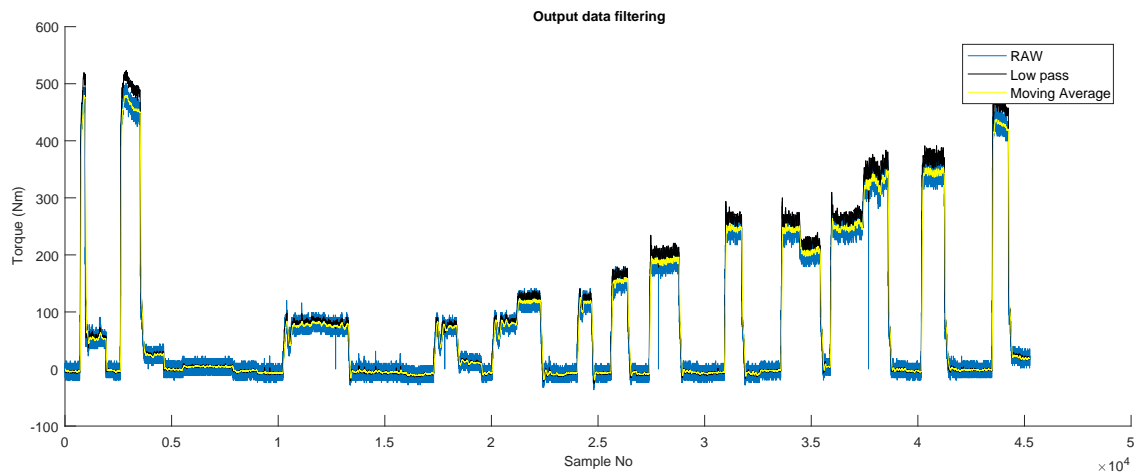


Figure 3.4: Filtered output data

Comparing the two data sets in Figure 3.4, it is clear that the moving average (the yellow curve) has removed significantly more of the noise than the low pass filter (the black curve). Furthermore, the moving average filter also seems to follow the trend of the original data (the blue curve) better than the low pass filter does. For this reason the data is passed through the moving average filter before being applied in the system identification toolbox in Matlab.

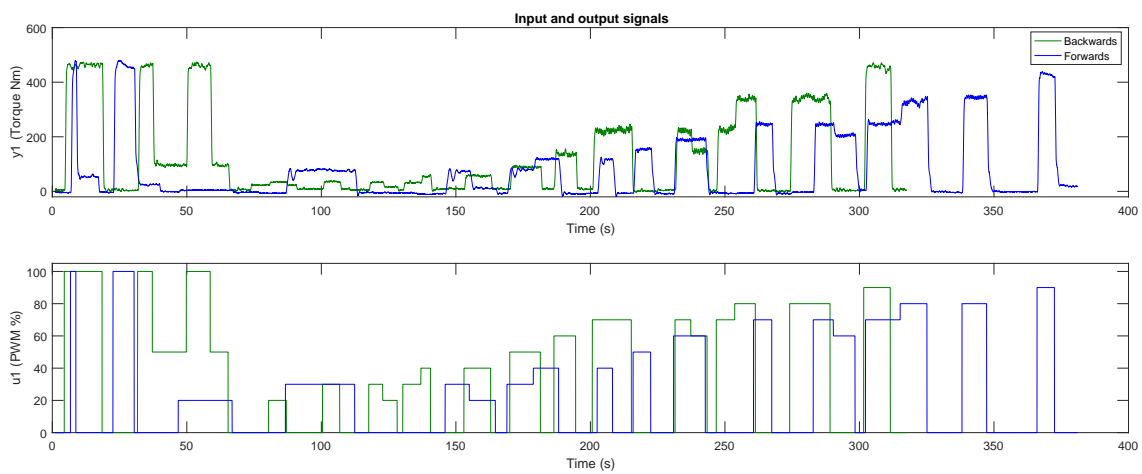
### 3.1.2 Model Identification

To used the system identification toolbox in Matlab, two data sets are required, the input and the output data. Additionally, it is necessary to know the sample time. The theoret-

ical sample time of the data is  $\approx \frac{1s}{142\text{ samples}} = 0.007\text{ s/samples}$ . This is based on the time the manufacturer of the microcontroller claims an ADC reading takes added with  $2\text{ ms}$  delays per reading included in the microcontroller code (See Appendix C.1).

To get a more accurate sample time, it was calculated from the thruster test by adding the time for all the tests, and dividing it in the total samples. By doing this, the average sample time is found to be  $0.0084\text{ s/samples}$ , which yields  $\approx 119$  samples per second.

With the sample time, the data sets for the forwards and backwards directions imported in the system identification tool box. Figure 3.5 show the time plot of both data sets and illustrate the input and output relation between the PWM signal and the resulting torque.



**Figure 3.5:** System identification Input/Output data

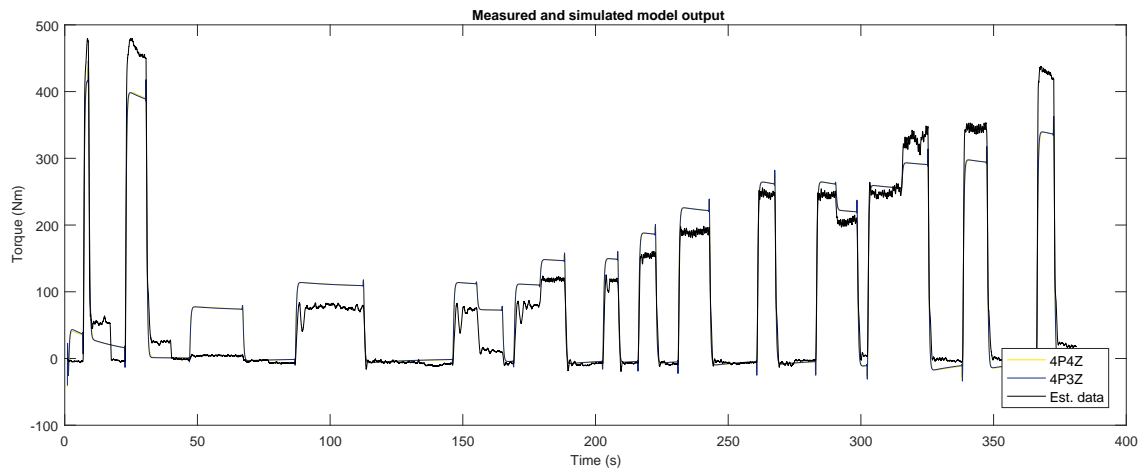
The next step is to estimate a transfer function for the two directions. This involves *guessing* the number of zeroes and poles of the function, knowing the type of system will help in approximating the order of the function. By changing this, the order of the equation is changed, and while including a higher order might increase accuracy, it is not given that it will. It is also imperative to remember that increasing the order will increase the computational requirement of the model. This is significant if the model is used at run time to predict system behaviour. In the following steps, only the forward direction will be covered. At the end however, a function for the backwards will also be shown.

First the system identification toolbox is used to find the best fitting transfer function for the pole and zero combinations shown in Table 3.1.

Poles / Zeroes	Estimation Fit (%)
2 / 1	71.93 %
3 / 2	71.96 %
4 / 3	72.90 %
5 / 4	72.64 %
4 / 1	71.97 %
4 / 2	72.8 %
4 / 4	72.92 %

**Table 3.1:** Transfer function combination

Looking at the result of these initial functions, the result is far from satisfactory. The estimation fit should be above 80 %, preferably even in the nineties. However, this is the estimation for the full data set, across the full input range.



**Figure 3.6:** Model identification for full input range

Figure 3.6 is the curve for the two best fitting functions, plotted against the output data. Looking at this, it is observed that the functions has trouble following in the upper and lower end of the range. This makes sense, as the thruster generates very little thrust at 20 % PWM, and the difference in thrust generated at 90 % and 100 % is close to zero. Appart from this, the thrust to PWM is also not linear, which also makes it difficult to achieve a good fit.

One way to solve this is, to extract the data of the thruster functioning between 30 % and 80 % input. In this range, it seems as if the thruster to PWM ratio is closer to being linear. By doing this, the fit is improved to  $\approx 77.5\%$ . Narrowing the range further to between 40 % and 80 % yields a fit of  $\approx 83.3\%$  while a range of between 30 % and 70 % gives a fit of  $\approx 86.5\%$ .

The trade off when narrowing the range is that while the function get more accurate when working within the range, as the input to the function moves outside the range, the error increases substantially. One way to avoid the issue is, to ensure the input to the function stays inside the range that has been used, meaning some of the potential of the thruster is lost. Alternatively, it can be accepted that the function is inaccurate

when working outside the specified range, with the goal that the controller will be able to handle the potential deviations.

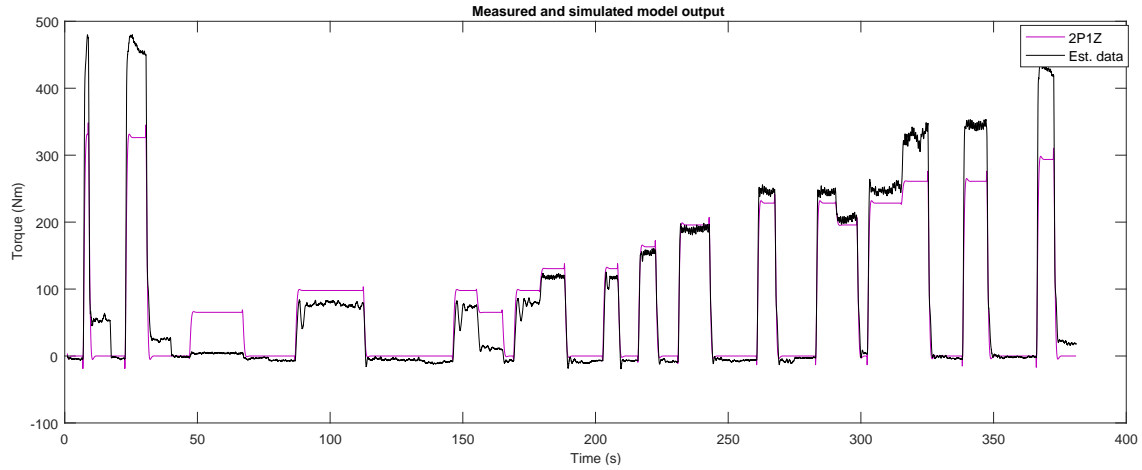


Figure 3.7: Output of model estimations based on 30 % to 70 % range

Figure 3.7 show the output of a transfer functions with 2 poles and 1 zeros fitted to the 30 % to 70 % PWM input range. Comparing Figure 3.7 to Figure 3.6 the difference can be seen. While the function is a vast improvement on predicting the output inside the 30 % to 70 % input range, when the input is outside this range, the prediction becomes much worse than it was previously.

The transfer function plotted in Figure 3.7 has a fit to estimation data of 86.47 % which is a significant improvement to the 72.92 % found on the full range. The transfer function for the thrust is given in Equation 3.2 below.

$$\frac{-4.261 s + 33.3}{s^2 + 5.117 s + 10.21} \quad (3.2)$$

The same procedure was done for the backwards direction. However, the behaviour of the thruster in the reverse direction seems to be slightly more erratic. This made getting an accurate model (above 60 %) difficult.

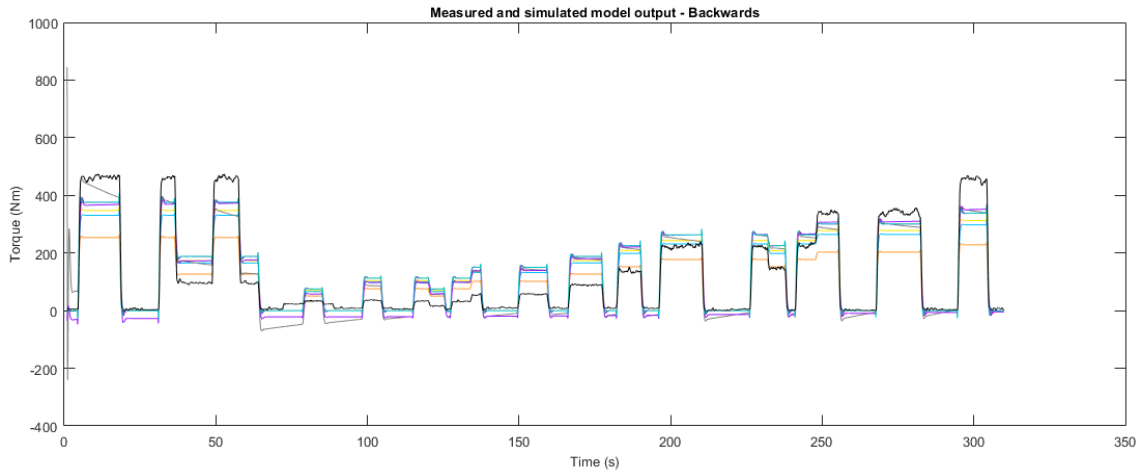


Figure 3.8: Output of modelling attempts for reverse direction

Figure 3.8 show a combination of the best models achieved. Except for one, all have a fit to estimation data in the early 60 % range. These estimates are the result of both the full range, limited range and even with an increase in filtering of the thrust data.

It is clear that the functions has difficulty following the low thrust generated in the lower range, overestimating the thrust. Likewise, in the high thrust range they have an equal difficult time following the actual behaviour. It follows the same tendency as was also observed for the forward direction (See Figure 3.7). For the backwards direction it is even worse. Unfortunately, the time table prevents solving this issue currently, it is noted that the model for the backwards direction of the thruster has to be reformulated as a fit of 63.19 % was the best one achieved, shown in Equation 3.3.

$$\frac{-5.449s + 43.29}{s^2 + 4.758s + 11.52} \quad (3.3)$$

As there could not be found a better fit for the backwards direction, for the system simulation and controller design, the model found for the forward direction will be used to describe both the forward and backwards direction. It is done for two reasons: firstly the thrust to PWM ratio of both directions is quite near to each other (See Table 2.2 in Section 2.4.2) and lastly the fit is so poor it will have to be re-evaluated.

At this point, there is little more than can be done. Improving the fit can be done a number of ways. One possibility is using a load cell with a higher sampling rate, as it would be possible to get at better description of the transient response of the thruster, while allowing for improved filtering of the data without harming the transients. Building upon this, an anti-aliasing filter could be created for the load cell sampler. This is essentially a *low pass* filter, which attenuates frequency above the *stop band*. By implementing the filter in hardware, and placing it between the *ADC* of the sampler and the load cell high frequency noise is removed [21]. This has the advantage of functioning in continuous time (on analog signals) when implementing it in the hardware, without being subject to issues arising from the Nyquist frequency.

An alternate solution to the poor estimate is to seek a non-linear model, as there are proven non-linearities in the thruster behaviour, clearly seen in the low PWM and high PWM ranges (e.g. Figure 3.5). The system identification toolbox has facilities for this,

and a quick test was done to get a *Hammerstein-Wiener* model for the backwards data. Without any adjustment, a fit to estimation data for the full set of 95.62% was achieved.

However, as the time is limited and it is an area in which lies outside the scope of the current project, this function will not be used for the remainder of the report, but it is an area that could be investigated as it seems to be a promising method to improve the fit without doing more experiments.

## 3.2 System Simulation

With the transfer function of the *SM-7* and the model describing the ROV pitch, the next phase is to combine the blocks and simulate the system. To do this, Simulink, an extension to Matlab, is used. This allows the modelling and simulation of the system using a combination of predefined blocks and the functions which has been found to describe the thruster and the ROV.

There is several different approaches than can be used to build the system, but the one chosen here is to split the problem into three over-all blocks; one block describes the thruster behaviour, another describes the ROV torque and finally the dynamics of the ROV, meaning the actual displacement (or pitch) is calculated in the third block. Figure 3.9 below show the a sketch of the concept.

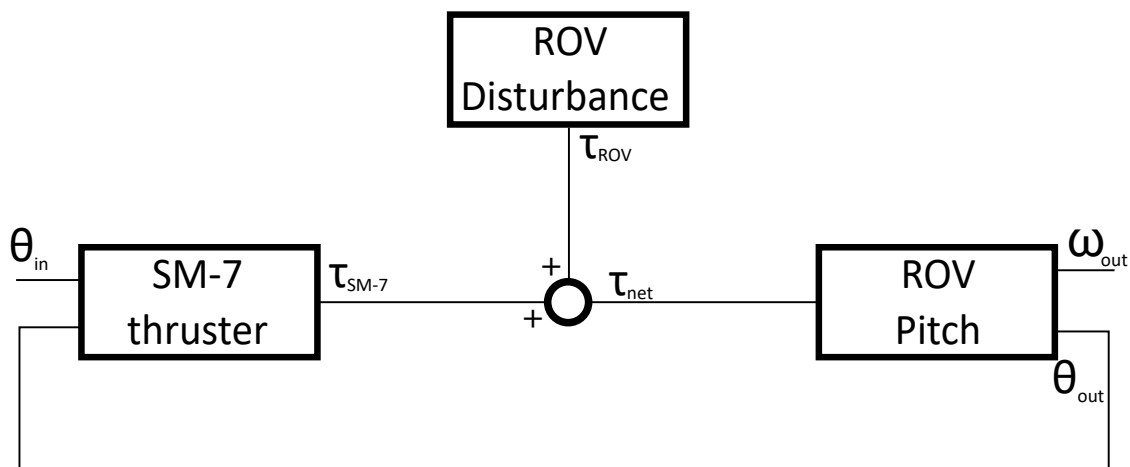


Figure 3.9: System model design

The *SM-7* block in Figure 3.9 is based on the transfer function for the thruster found in Section 3.1. This block takes two inputs: the current set point (in radians) and the actual angle (also in radians). Using these two pieces of information, the required torque will be calculated. By using the function to calculate the PWM for a given torque, found in Section 2.4.2, the input to the transfer function can be found.

Describing the torque that the ROV generates as the pitch angle is changed, is modelled as a *disturbance* (see Figure 3.9). This block describes the current torque that the ROV causes which works towards orienting the ROV back to  $0^\circ$ , identified in Section 2.3.1. Additionally, the torque generated by the drag of the ROV, depending on the angular velocity,

is also included in this block. The description for this was found in Section 2.3.3.

The final part of the system is the ROV block. This block takes the combined torque of the thruster and the disturbance ( $\tau_{net}$ ) and uses the system model (see Section 2.3.4) to calculate first the angular velocity ( $\omega$ ) and based on this the angular displacement of the ROC ( $\theta_{out}$ ). By building these blocks in Simulink it is possible to evaluate the behaviour of the system.

An important factor to remember when building the system is the reference direction. This determines the sign ( $\pm$ ) of the angles, velocities and torques of the system, in relation to which direction they prescribe.

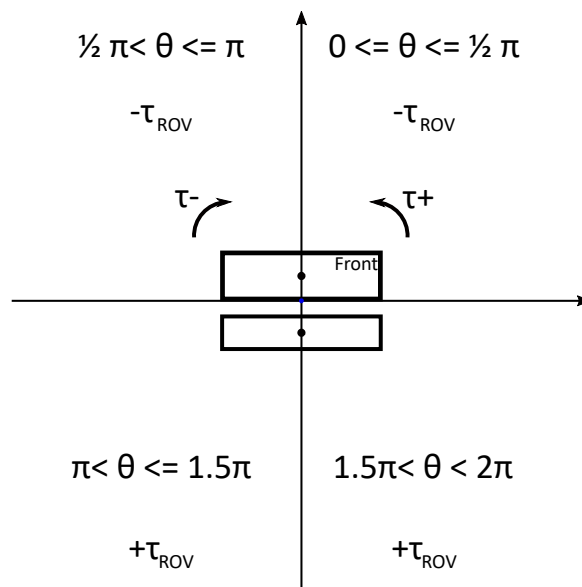


Figure 3.10: System reference point

Figure 3.10 above show a sketch of the ROV placed in a coordinate system. The angles for each quadrant is included and the torque,  $\tau_{ROV}$ , indicates the sign of the torque generated by the ROV when the ROV has an angle within the interval of the quadrant. In a similar manner, the arrows indicate the direction which the ROV will pitch if a torque is applied with a matching sign. For example, this means if a pitch angle of  $\frac{\pi}{7} \approx 26^\circ$  is wanted, a positive torque has to be applied. As can be seen in Figure 3.10, the torque of the ROV will be of negative direction, meaning it is working against the forced pitch.

### 3.2.1 Thruster Block

The thruster block describes the logic that will lay the foundation for building the controller for the SM-7 thruster. This is based on two inputs, as seen in Figure 3.9. The set point,  $\theta_{in}$  and the feedback from the system,  $\theta_{out}$ , based on these two the block calculates the PWM value that is needed to generate the thrust to ensure the pitch angle reaches the correct level. By using the transfer functions for the thruster that was found previously, the block emulates the behaviour of the actual thruster, meaning when a con-



troller is developed it is possible to tune the controller before fitting it to the actual system.

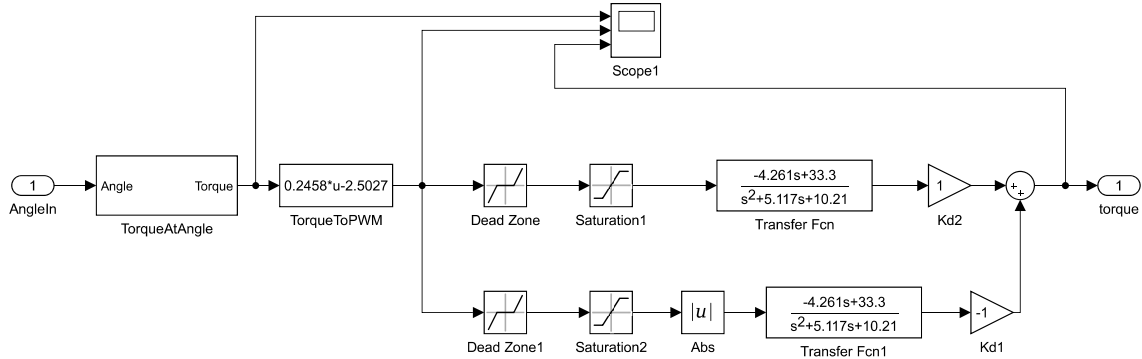


Figure 3.11: Thruster system block

Figure 3.11 show the Simulink model for the thruster. The first step for the thruster block is to convert the angle input ( $\theta_{in}$ ) to a torque. This calculation is based on the formula for the torque of the ROV found in Section 2.3.2. So from this the block first calculates the torque that the ROV generates at the set angle, and from this follows the required torque to maintain that angle.

After this the required PWM value to maintain the torque is calculated, this is based on the fit of the PWM vs. Torque curve found in Section 2.4.2. After this step the PWM value is passed through a *Dead Zone* block. This causes any value between 0 and 20% to be floored to 0%. The reason is that the *SM-7* does not run below this duty cycle. After this step, the value is passed through a *Saturation* block, which causes values outside the 0 to 100% range to be rounded to the nearest value.

Then, the value is input to the transfer function of the thruster, which will generate an output torque in relation to the model found for the thruster. This torque value is output from the block and then follows the logic in Figure 3.9.

The two branches in Figure 3.11 after the *TorqueToPWM* function, handles the direction of the thruster. The bottom branch creates the same *Dead Zone* and *Saturation* parameters as described above, but for negative PWM values. The *Abs* block transforms the value to a positive one and parses it to the transfer function.

### 3.2.2 Disturbance Block

The disturbance system block calculates the torque generated by the ROV as a results of being pitched. This is built around the function describing the ROVs tendency to return to a neutral pitch angle (See Section 2.3.2). In addition to this, the drag of the ROV as a function of the velocity adds another torque force. This was described in Section 2.3.3.

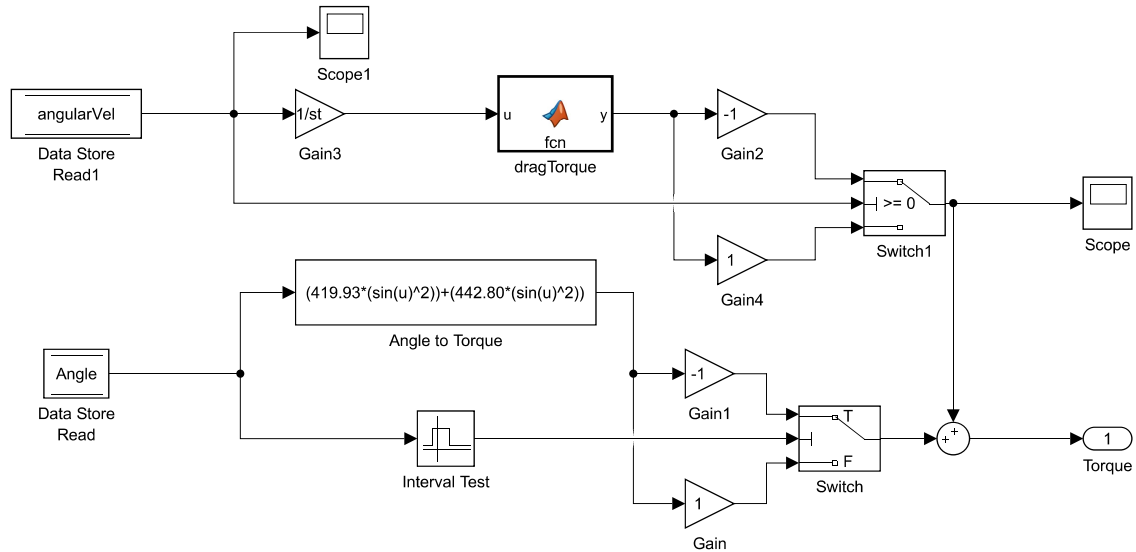


Figure 3.12: Disturbance system block

The disturbance system block, shown in Figure 3.12, reads the current angle ( $\theta_{out}$ ) and angular velocity ( $\omega$ ) of the ROV and uses this to calculate the torque caused by the ROV during pitching. The torque generated by the mass distribution of the ROV frame, pitching the ROV towards  $0^\circ$ . In the model, the formula used to calculate the torque of the ROV is used.

$$\tau_{pitch} = 419.93 \text{ N m} \cdot \sin^2(\theta_{ROV}) + 442.80 \text{ N m} \cdot \sin^2(\theta_{ROV}) \quad (3.4)$$

Equation 3.4 is used to evaluate the torque at any angle. As  $\sin^2$  cannot yield a negative number, this equation will always yield a positive torque, so to ensure that the direction of the torque satisfies the actual situation, a *switch* block is implemented in the system in Figure 3.12. This ensures that depending on the result of the *Interval Test* block, the output of the torque function is either passed through a positive or negative gain.

The *Interval Test* block checks the current angle of the ROV compared to the sign the torque must possess, given the direction established in Figure 3.10. This means, for an angle,  $\theta_{out}$  between  $0$  and  $\pi$  the torque is negative, resulting in a clockwise acceleration. Likewise, for an angle above  $\pi$  and below  $2\pi$  the torque is positive, resulting in an anti-clockwise acceleration. Depending on which interval the angle is within, the *Switch* block is either set or reset.

Additionally, the drag is calculated based on the current angular velocity of the ROV. This drag is converted to the equivalent torque it generates, as detailed in Section 2.3.3. The velocity is first passed through a gain of  $\frac{1}{\text{sampletime}}$ . This is done as the angular velocity is calculated as  $\text{rad}/\text{sampletime}$  while the drag equation requires  $\text{rad}/\text{s}$ . After this the drag is passed through a *Gain* block of  $\pm 1$ . Which gain is applied depends on the direction of the angular velocity, which is controlled by the *Switch* block.

The result of the drag calculation is added to the result of the torque force of the ROV and the combined torque is output as  $\tau_{ROV}$ .

### 3.2.3 ROV Block

The ROV system block is the plant of the system. It takes the torque ( $\tau_{net}$ ) of the system as the input and from this calculates the angular velocity ( $\omega$ ) and the angular displacement ( $\theta$ ) of the ROV.

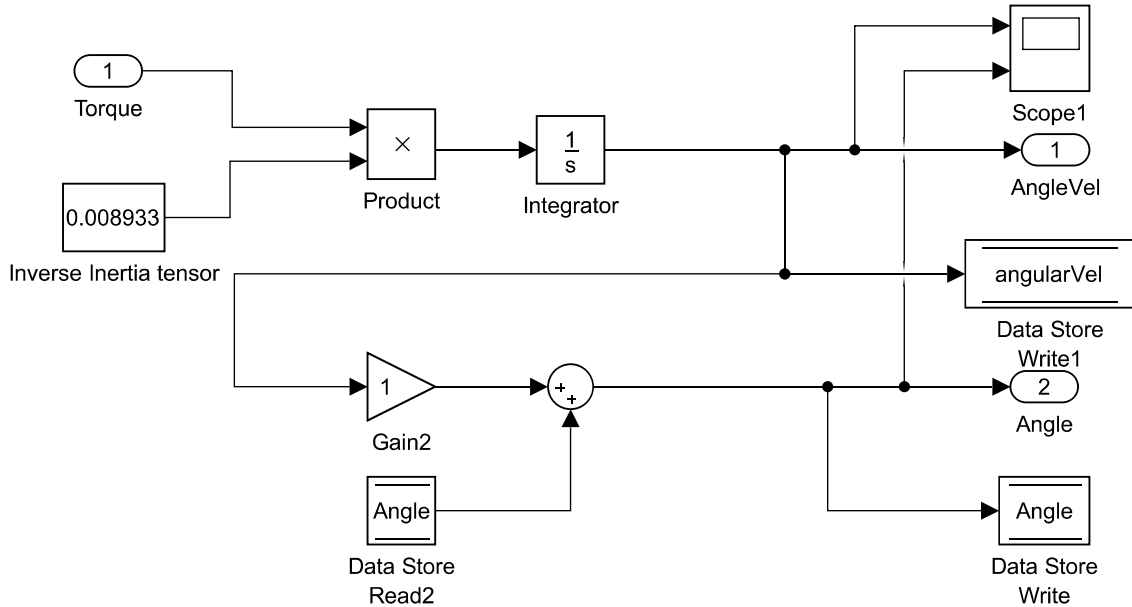


Figure 3.13: ROV system block

The ROV block in Figure 3.13 is the Simulink circuit of how the ROV pitch block in figure 3.9 is created. The method is based on the equations identified in Section 2.3.4 to describe the pitch of the ROV. The calculation is divided into a two step process, first being calculating the angular velocity. From this calculation of angular velocity, the angular displacement can be calculated, this is assuming a constant velocity across the sample time.

$$v_{ROV} = \begin{bmatrix} 0 \\ I^{-1} \int_0^{\Delta t} (\tau_{ROV}(\theta_{net})) dt + \omega_0 \\ 0 \end{bmatrix} \quad (3.5)$$

Calculating the angular velocity for the ROV block in Figure 3.13 uses Equation 3.5. First the torque is multiplied with the inverse inertia tensor through the *Product* block. The result of this is integrated in the  $\frac{1}{s}$  block. As Simulink integrals work by accumulating the values from previous integration steps, the old velocity ( $\omega_0$ ) does not need to be added to the result explicitly. The angular velocity is then saved to a memory block using a *Write* block, which allows it to be used for calculating the drag (See Section 3.2.2).

$$\eta_{ROV} = \begin{bmatrix} 0 \\ \omega_1 \cdot \Delta t + \theta_0 \\ 0 \end{bmatrix} \quad (3.6)$$

The second step is calculating the angular displacement ( $\theta_{out}$ ) this is based on Equation 3.6. In Figure 3.13 the output from the integrator block (being angular velocity) is passed through a *gain* block. This gain equates to multiplying the velocity with the  $\Delta t$  in Equation 3.6. As the velocity is calculated as  $\frac{rad}{sample\ time}$ , this gain is 1. The former velocity is added to this result, which is then saved in another *Write* block.

### 3.2.4 System Evaluation

With the system shown in Figure 3.9 defined in the Simulink environment, a test of the system were performed to see how it behaves across the input range before creating a controller for the system. A question to consider is what requirements there are of a controller. It is conceivable that the straight forward open loop control proportional scheme is sufficient for the problem.

To investigate the system, a simple test in Simulink is performed. This is this is done by inputting different angle set points and analysing the output angle. By recording the time it takes to reach a steady state and potential steady state offset, the requirements for the controller can be established.

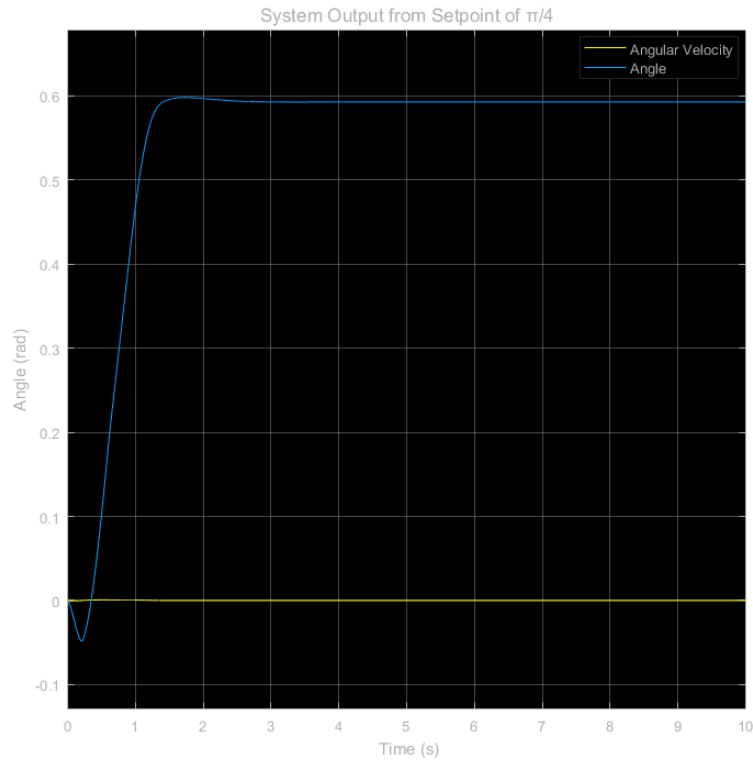
Input angle	Output angle	Deviation	Steady state time
$18^\circ / \frac{\pi}{10}^c$	$-3.5^\circ / -0.0615^c$	$21.5^\circ$	$\approx 2.6\ s$
$20^\circ / \frac{\pi}{9}^c$	$4^\circ / 0.0703^c$	$16^\circ$	$\approx 2.8\ s$
$22.5^\circ / \frac{\pi}{8}^c$	$11.3^\circ / 0.1979^c$	$11.2^\circ$	$\approx 3.0\ s$
$25.7^\circ / \frac{\pi}{7}^c$	$14.4^\circ / 0.2518^c$	$11.3^\circ$	$\approx 3.0\ s$
$30^\circ / \frac{\pi}{6}^c$	$19.5^\circ / 0.3405^c$	$10.5^\circ$	$\approx 3.0\ s$
$36^\circ / \frac{\pi}{5}^c$	$25.7^\circ / 0.4486^c$	$10.3^\circ$	$\approx 3.0\ s$
$45^\circ / \frac{\pi}{4}^c$	$34.0^\circ / 0.5927^c$	$11.0^\circ$	$\approx 3.0\ s$
$60^\circ / \frac{\pi}{3}^c$	$37.7^\circ / 0.6583^c$	$22.3^\circ$	$\approx 3.0\ s$
$90^\circ / \frac{\pi}{2}^c$	$37.7^\circ / 0.6583^c$	$52.3^\circ$	$\approx 3.0\ s$
$-18^\circ / -\frac{\pi}{10}^c$	$-5.9^\circ / -0.1022^c$	$12.1^\circ$	$\approx 2.5\ s$
$-20^\circ / -\frac{\pi}{9}^c$	$-9.6^\circ / -0.1670^c$	$10.4^\circ$	$\approx 3.0\ s$
$-22.5^\circ / -\frac{\pi}{8}^c$	$-13.1^\circ / -0.2284^c$	$9.4^\circ$	$\approx 3.0\ s$
$-25.7^\circ / -\frac{\pi}{7}^c$	$-16.9^\circ / -0.2955^c$	$8.8^\circ$	$\approx 3.0\ s$
$-30^\circ / -\frac{\pi}{6}^c$	$-21.5^\circ / -0.3751^c$	$8.5^\circ$	$\approx 3.0\ s$
$-36^\circ / -\frac{\pi}{5}^c$	$-27.3^\circ / -0.4770^c$	$8.7^\circ$	$\approx 3.0\ s$
$-45^\circ / -\frac{\pi}{4}^c$	$-35.3^\circ / -0.6169^c$	$9.7^\circ$	$\approx 3.0\ s$
$-60^\circ / -\frac{\pi}{3}^c$	$-37.9^\circ / -0.6622^c$	$22.1^\circ$	$\approx 3.0\ s$
$-90^\circ / -\frac{\pi}{2}^c$	$-37.9^\circ / -0.6622^c$	$52.1^\circ$	$\approx 3.0\ s$

Table 3.2: Test cases system without controller

Table 3.2 show the result of inputting different angular set points in the system model. As would be expected with using the same function for the thruster directions, the switch in direction has little impact. The small deviation that is present is due to the fact the model describes a small negative thrust generation at low PWM values.

By looking at the data in Table 3.2, it seems that the time delay to reach a steady state level is fairly consistent of  $\approx 3.0$  seconds. It is also worth noting that there is a steady state

offset consistently throughout, but it is smallest for angles between  $\pm 11^\circ \rightarrow 45^\circ$ .



**Figure 3.14:** Output of system at  $\frac{\pi}{4}$  input

Figure 3.14 show the output of the system with an input of  $\frac{\pi}{4} = 45^\circ$ . The output is calculated across a 10s time period. The tendency of the system in Figure 3.14 was seen across the full input range in Table 3.2. There is a tendency of a small overshoot, which is expected as the generated thrust will accelerate the ROV, and the torque working against the thruster will have to counteract the velocity that the acceleration has caused before the ROV settles into a steady state.

With this knowledge of the system, the following factors can be used to evaluate the effectiveness of the controller that has to be added to the system.

- Steady state offset
- Overshoot
- Settling time

The current overshoot of the system is relatively small, and adding a controller has a great chance of increasing this, as the controller will work towards accelerating the ROV faster towards the settle point. Steady state offset of the system is one factor a controller will be able to fix, implementing an *I*-controller will accumulate the error between the input and output, meaning it will increase or decrease the thrust until the error is zero [22].

### 3.3 Controller Design

In the following, the controller for the thruster will be added to the Simulink model, specifically, the controller is added to the thruster block (described in Section 3.2.1). The controller will be based on the *PID* control scheme, so before developing the controller, the type of controller that can solve the problem is evaluated.

Controller Response	Rise time	Overshoot	Settling time	Steady state error
$K_p$	Decrease	Increase	Small change	Decrease
$K_i$	Decrease	Increase	Increase	Eliminate
$K_d$	Small change	Decrease	Decrease	No change

**Table 3.3:** *PID*-gains output effect [23]

Evaluating which controller to apply can be done by comparing the effects of the three gains: Proportional, integral & derivative (Shown in Table 3.3) with the requirements of the controller. One important note of the derivative gain is that it works by amplifying the *rate of change* of the input error, a very powerful control method. It is however, very likely that if it is implemented in a system with noise on the feedback line, the controller will amplify the noise return, and the result is a very erratic and unstable system.

For that reason the *d*-gain will not be included in any of the controllers designed, as the potential for an unstable system is not an acceptable trade off for the benefits of including the gain.

With this in mind, two controllers will be created and compared. First a simple *P*-controller will be created. As per Table 3.3, this should decrease the steady state error and the rise time, at the expense of a increase in overshoot.

Secondly, a combined *PI*-controller will be created, and this should eliminate the steady state error. The challenge will be to limit the overshoot, as it has the potential of increasing the settling time of the system.

Before building a controller, the feedback system and control variable has to be decided. While the feedback variable is the angle of the ROV, which is subtracted from the set point, yielding an error term describing the angle offset from the set point. There are several ways the error term can be used to create a control signal.

One method would be to recalculate the error term to a torque difference, which would be added output of the block that calculate the torque requirements at the set point (See Section 3.2.1 for more details).

Alternatively, the torque calculated from the error term could be converted to a PWM value, and this PWM value would be added to the one calculated from the setpoint, increasing the PWM for the motor in relation to the error term.

A final method tried is to convert the error term to an angle in degrees, and adding this to the PWM output. This essentially multiplying the error with a gain of  $\frac{180}{\pi}$  before passing it through the controller and adding it to the PWM.

The three approaches described was all tested during the controller setup. Calculating a PWM value from the error term has the disadvantage, that the torque to PWM function is not linear, so a torque calculated does not scale across the full PWM range. This meant the controller had a difficult time maintaining the error term at zero.

Using the angle directly to control the thruster works very well. Converting it to degrees essentially ensures that even small angles gets corrected by the controller.

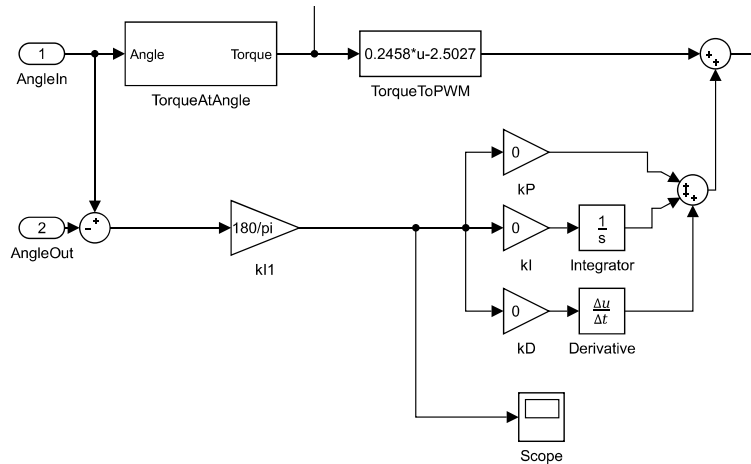


Figure 3.15: Thruster with *PID*-Controller

Figure 3.15 show how the *PID*-controller scheme is included for the thruster block (Based on the same block shown in Figure 3.11). By changing the value of the *Gain*-blocks ( $k_P$ ,  $k_I$  &  $k_D$ ) the controller can be tuned. For the following the gain  $k_D$  will remain at zero.

To get a controller for the system, first a *P*-controller will be developed, and the gain that seems to give the best rise time without getting too much overshoot. The  $k_P$  value found for the *P*-controller will then be used in the *PI*-controller where different values for the  $k_I$  will be tried, which will lead to a *PI*-controller that can maintain the pitch angle of the ROV.

### 3.3.1 *P*-Controller

As described, the tuning method for the controller is to first find the  $k_P$  gain that has to most promising effect on the control of the ROV. To do this the  $k_P$  *gain*-block (See Figure 3.15) is tested with different values and the output angle is saved.

The controller is tuned with an input of  $\frac{\pi}{5}$ . Using this input has two reasons; First the angle is sufficiently large that the controller will use the maximum PWM value. Secondly, the angle is still inside the range that the thruster can maintain (See Table 3.2).

For the proportional gain, the gain is incremented in steps of 2 within the interval  $2 \rightarrow 20$  and the output data is plotted next to each other to make it easier to compare the effect of the increasing gain.

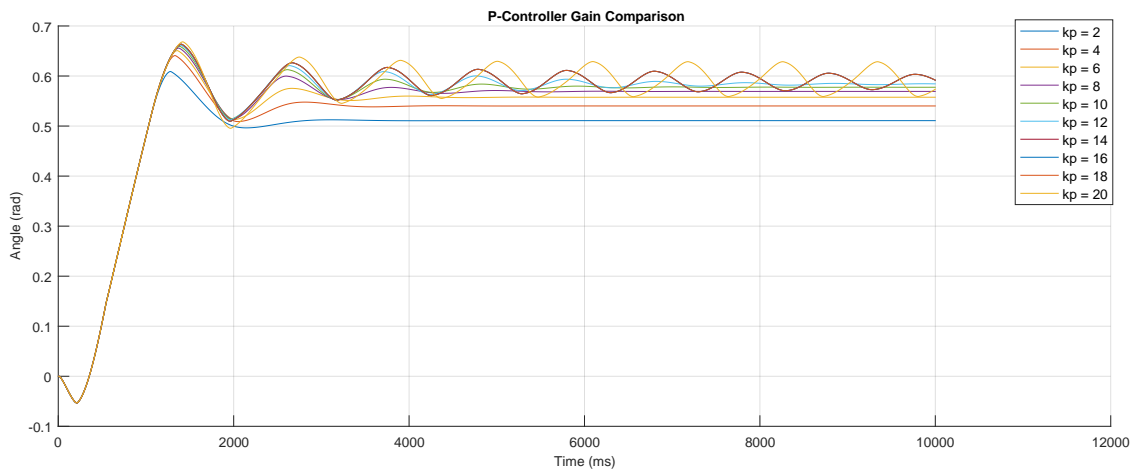


Figure 3.16: Output of system with a  $P$ -controller at  $\frac{\pi}{5}$  input

Figure 3.16 show the output curve of the  $P$ -controller. As expected of the  $kP$  gain, a significant increase in the overshoot is observed as the gain value increases. This also leads to a oscillating behaviour of the output, as the controller will have a tendency to over compensate.

Evaluating the steady state offset of the controller can be done by comparing the output values to the set point of  $\frac{\pi}{5} \approx 0.63$ . The controller performs as expected here as well, as the steady state error is reduced for an increased  $kP$ , but none of them manage to eliminate the offset.

Finally, the rise time is also expected to be reduced for an increasing  $kP$  gain. Looking at Figure 3.16, the decrease in rise time can be identified by the tendency of the outputs; for an increasing  $kP$  gain yields a steeper angle before changing direction, this is also what leads to the increased overshoot.

Comparing the different gains in Figure 3.16, the optimum gain seems to be in the vicinity of  $kP = 2$ . While the value has a relatively large steady state offset, it seems the one with the fastest settling time. By adding an integrator to this  $P$ -controller, the steady state offset should be fixed.

### 3.3.2 $PI$ -Controller

Finding the value for  $kI$  gain follow the same method as described for the  $kP$  gain, but this will be incremented in steps of 0.5 in the interval  $0.5 \rightarrow 4$ . The reason for this smaller step size is that the integral is an accumulator, meaning it can easily cause significant overshoot as it will accumulate a large error that has to be compensated.

The test follow the same input as before,  $\frac{\pi}{5}$  and the value found of the  $kP$  gain is used. The output curves is once again plotted next to each other to compare the behaviour.



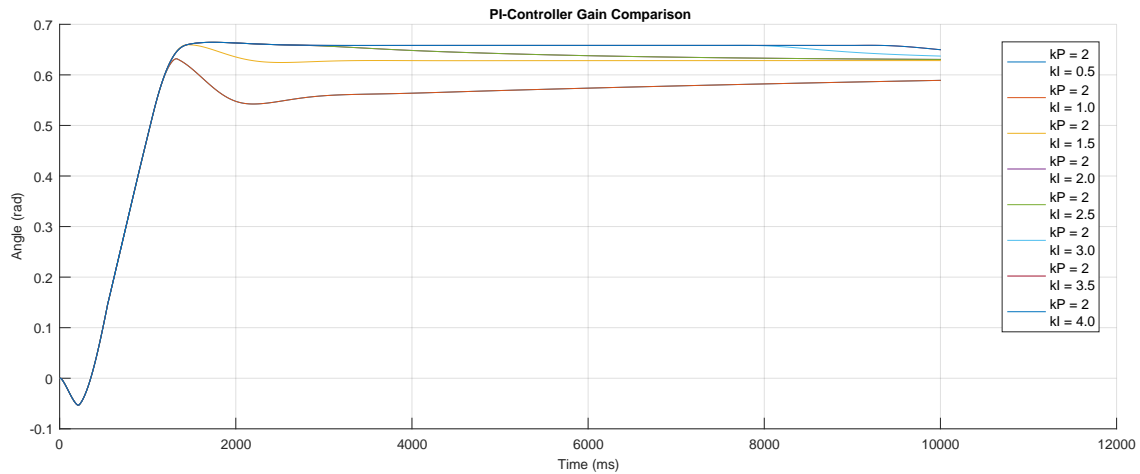


Figure 3.17: Output of system with a  $PI$ -controller at  $\frac{\pi}{5}$  input

Figure 3.17 show the output from the different  $kI$  gain values. By comparing the output curve to the one for the pure  $P$ -controller, the effect of the  $kI$  gain is clear. First of all, it is clear that every output has either reached the set point or are very near it with a tendency of approaching the point.

The output of the  $PI$ -controller also have a clear tendency of overshooting the set point, and any  $kI$  gain above 1.5 seems to make the output remain above the set point for an extended period of time. However, below a gain of 1.5 the output has a tendency to oscillate around the set point. Therefore, choosing the gain for the  $kI$  is a trade off between a fast settling time, with oscillating behaviour, or a longer settling time, but with a more steady approach towards the set point.

The  $kI$  gain has helped reduce the settling time of the system and eliminated the steady state offset. This makes it invaluable together with the  $kP$  gain. Looking at the output curves. The choice of gain is based on reduced oscillation of the output, with an accepted trade off of the settling time, for this the best gain value for the integral term is in the vicinity of 2.5. This seems to be the value that gives the fastest settling time and the most stable system behaviour.

### 3.4 Conclusion

With the  $PI$ -controller created in Section 3.3.2, the final objective is to compare the result of the system with a controller as opposed to no controller. This is done by subjecting the Simulink model with the  $PI$ -controller to the same input range as was done in the system evaluation in Section 3.2.4 Table 3.2.

Input angle	Output angle	Deviation	Steady state time
$18^\circ / \frac{\pi}{10}^c$	$18.0^\circ / 0.3140^c$	$0.0^\circ$	$\approx 4.5\text{ s}$
$20^\circ / \frac{\pi}{9}^c$	$20^\circ / 0.3490^c$	$0.0^\circ$	$\approx 3.6\text{ s}$
$22.5^\circ / \frac{\pi}{8}^c$	$22.9^\circ / 0.3993^c$	$0.4^\circ$	$\approx 3.9\text{ s}$
$25.7^\circ / \frac{\pi}{7}^c$	$25.7^\circ / 0.4490^c$	$0.0^\circ$	$\approx 4.0\text{ s}$
$30^\circ / \frac{\pi}{6}^c$	$30.0^\circ / 0.5240^c$	$0.0^\circ$	$\approx 4.0\text{ s}$
$36^\circ / \frac{\pi}{5}^c$	$36.3^\circ / 0.6340^c$	$0.3^\circ$	$\approx 9.2\text{ s}$
$45^\circ / \frac{\pi}{4}^c$	$37.7^\circ / 0.6583^c$	$7.3^\circ$	$\approx 2.8\text{ s}$
$60^\circ / \frac{\pi}{3}^c$	$37.7^\circ / 0.6583^c$	$22.3^\circ$	$\approx 2.8\text{ s}$
$90^\circ / \frac{\pi}{2}^c$	$37.7^\circ / 0.6583^c$	$52.3^\circ$	$\approx 2.8\text{ s}$
$-18^\circ / -\frac{\pi}{10}^c$	$-18^\circ / -0.3141^c$	$0.0^\circ$	$\approx 3.6\text{ s}$
$-20^\circ / -\frac{\pi}{9}^c$	$-20^\circ / -0.3491^c$	$0.0^\circ$	$\approx 3.7\text{ s}$
$-22.5^\circ / -\frac{\pi}{8}^c$	$-22.5^\circ / -0.3993^c$	$0.0^\circ$	$\approx 4.1\text{ s}$
$-25.7^\circ / -\frac{\pi}{7}^c$	$-25.8^\circ / -0.4495^c$	$0.1^\circ$	$\approx 4.6\text{ s}$
$-30^\circ / -\frac{\pi}{6}^c$	$-30.1^\circ / -0.5250^c$	$0.1^\circ$	$\approx 5.5\text{ s}$
$-36^\circ / -\frac{\pi}{5}^c$	$-36.4^\circ / -0.6350^c$	$0.4^\circ$	$\approx 8.5\text{ s}$
$-45^\circ / -\frac{\pi}{4}^c$	$-37.9^\circ / -0.6622^c$	$7.1^\circ$	$\approx 2.5\text{ s}$
$-60^\circ / -\frac{\pi}{3}^c$	$-37.9^\circ / -0.6622^c$	$22.1^\circ$	$\approx 2.5\text{ s}$
$-90^\circ / -\frac{\pi}{2}^c$	$-37.9^\circ / -0.6622^c$	$52.1^\circ$	$\approx 2.5\text{ s}$

Table 3.4: Test of system controller

Table 3.4 show the result of testing the controller through the same input range as the original system evaluation test. From this table it is clear that the objective of eliminating the steady state error is achieve, as long as the angle is within the maximum achievable with the thruster ( $\approx 37.7^\circ$ ).

One important note is that this was all done with the same transfer function for the thruster in both directions, as a description for the backwards direction of the thruster was not achieved. This means while the current control works for the ROV and thruster setup currently, it will be necessary to find a solution for the backwards direction, to get a better description of this.

Apart from this, the model of the system works and it was possible to identify a *PI*-controller that is able to maintain the pitch angle of the ROV. With an improved fit for the backwards direction of the thruster, it should be possible to use the controller for the *SM-7* thruster.

Building upon this, the controller could be improved, and one direction that could be considered is *Adaptive* control schemes. If done correctly, this control scheme can assist in reducing the impact of poorly modelled phenomenon, as it would allow the controller to adjust the gains based on the feedback, for example scaling the gain blocks based on the change rate of the feedback.

## Chapter 4

# Discussion

The following section will aim to to overview of the status of the project at the time of its finalisation. The success and failures will be highlighted and a few comments for the the reason will be shared and a brief description of what needs to be done for a full scale system implementation will be made.

Before getting too deep into this, a quick overview of the status of the project is a good idea. The task list from Section 1.1-Project Description is presented below in Table 4.1.

No.	Task	Accomplished
1	Driver Electronics	✓
2	Communication	✓
3	ROV model	✓
4	Thruster model	≈
5	System simulation	✓
6	Controller design	✓
7	Implementation	✗

Table 4.1: Project progress

Looking at table 4.1, it is clear that the majority of the tasks has been completed. However, two tasks remain incomplete. The incomplete tasks is the thruster model and the implementation of the control system.

While it was succeeded in finding a model of the thruster, this was done in Section 3.1, it was only achieved for a single direction of the thruster. However, for the backwards direction the fit of the transfer functions found using the System Identification toolbox was simple too poor to be acceptable. The reason why the fit was so poor is fairly hard to know. It was observed that the formulas generally has a difficulty following the thrusters behaviour in the extremes of the input range.

This was not bigger surprise as the system works best for Linear Time Invariant systems (LTI). And as was found, the thrusters behaviour was non linear in the *extreme* input range. A fact that is abundantly clear when looking at its thrust to PWM ratio (see Table 2.2 in Section 2.4.2), as the thrust output has little change in the 80 % → 100 % PWM input range, and the below 20 % PWM the thruster generates no thrust.

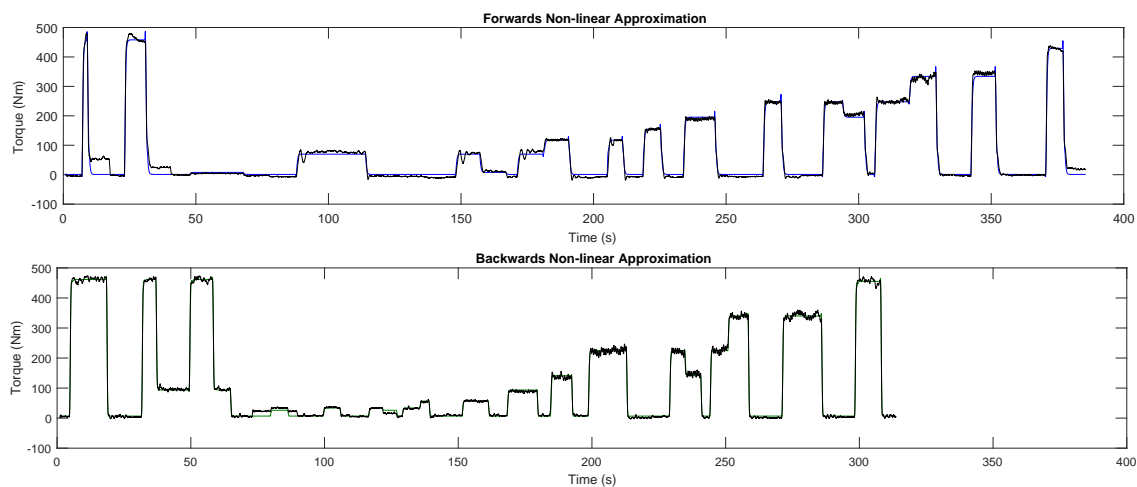
This cannot be the only reason for the poor fit of the backwards direction compared to

the forward direction, as the output/input relation is quite similar. And for the forward direction a fit to estimation data of 86.47% was achieved by approximating the model in the range of 30 → 70% range. By creating the fit inside this range, the risk is that the model will be worse when working outside the range, compared to a poorer fit across the whole range. The hope is however, that the thruster will generally perform inside the range where it is tuned to.

Working below the 30% is unlikely, as the thruster creates relatively little thrust there. The issue is then, that the model might overestimate the thrust generated below thirty, and the effect is a controller that *thinks* it creates a larger thrust than is actually being generated. Looking at Figure 3.7 in Section 3.1.2, it can be seen that this is what will happen at values below 30%. This is where a controller based on a feedback should hopefully catch the error and increase the output to the thruster.

The situation becomes worse in the other end of the spectrum, as can be seen in the same figure, the controller will have a tendency to underestimate the thrust generated at PWM values above 70%. This is a situation with potentially worse consequences, as the system could be destabilised as the controller will have a tendency to overshoot the set points worse than it is *tuned* to. The issue can be overcome by limiting the output of the controller, which will also mean that significant capabilities of the thruster is lost.

Alternatively to this, as a new model has to be sought for the backwards direction, the same could be done for the forward direction. It was found that a non-linear model seemed to be much better at describing the behaviour of the thruster.



**Figure 4.1:** Non-Linear model output vs. measured

Figure 4.1 show the output curve of two non-linear models, these were approximated based on the full range data set of the forwards and backwards direction. Looking at the curve, it is clear the model has a much better ability to follow the behaviour of the thrust in the low and high input ranges. Both models we approximated as *Hammerstein-Wiener Models*, with all settings at the Matlab *default*. The reason for not using the non-linear modelling is that improvement of the non-linear model was observed too late, and it is an area outside the scope, it would require quite an addition to the report to apply the correct methodologies and not simply *blind modelling*, which is very much what was done for the output of Figure 4.1.

With the lack of a model of the thruster for both directions, it follows that implementation of the controller was not achieved on the full system, as this would be hazardous at best. However, a controller was developed, which was tested with the same model for both directions, based on the assumption that the input/output relation of the thruster was similar for both directions. This led to the development of a *PI*-controller for the system, which was able to maintain the pitch angle of the system. Theoretically.

While a simple *PI*-controller was tested and seems to be quite sufficient to maintain the pitch angle (See Table 3.4 in Section 3.4), there is still room for improvement. The controller has a tendency to overshoot slightly and approach the set point from a too high value, this is something that could be sought to be improved. It might not be achievable with a *PI*-controller, so different control schemes could be investigated to build upon the *PI*-controller.

One path that could be investigated is to create an *adaptive* control algorithm. If this was done right, it could be used to make the control system more robust towards changes in operating conditions. This can for example be achieved by looking at the feedback change rate, and alter the controller gains depending upon the expected change rate vs. the actual one.

The dynamic model of the Cougar system which was required to test and tune the controller was implemented successfully. It was possible to create a mathematical description of the Cougar as it pitches. And combined with the model of the thruster the whole pitching manoeuvre was simulated using the Simulink toolbox in Matlab. Recalling that this model was simplified in some regards (See Section 2.3.1 & Section 2.3.3) it is worth noting that how close the behaviour observed through the model lies to reality is unknown.

For example, wave and current induced loads were ignored as they are generally moving in a direction that does not directly change the pitch angle. This is an assumption that will not always be true, and one that can be particularly dangerous for high currents while the ROV is pitching.

The drag was also simplified drastically to being treated as a sum of small rectangles, how close or how far from the reality this really is, is difficult to know, probably it is quite far, as there is a lot more to the ROV than a simple rectangle. With this being said it yielded a dampening effect which ensures velocity is lost when no torque is applied.

One area that was never touched upon, was how the control from the topside will be tied together with the Arduino in the electrical pod in the ROV. The most simple way would be to use the same method as for the thruster test, where a PC program uses a serial RS-232 interface to communicate with the Arduino. This could be built on the same program used for the thruster test; instead of setting a PWM value and direction, a simple angle would be input.

This would require the least amount of work as most of the requirements for this are already implemented. It does add the need for a computer to run the program, and space can be limited offshore, so another way to do is considered.

This entails building a small control unit, where a microcontroller will allow a user to set an angular value, could be via two buttons, incrementing and decrementing the

angle along with 2 7-segment displays to present the chosen value. The advantage of this is the control console for the thruster could be kept near the same control panel used for the ROV. This gave the benefit that the ROV pilot can operate both the ROV and the thruster, as it were one system.

With this being said, it can be said that implementing a pitch controller on the Cougar is possible, and while some questions still remain, what has been found in the project can lay the foundation for implementation of the system utilising the *SM-7* thruster for the task. As it was possible to describe the ROV's pitch angle mathematically, and the thruster model was identified using *black box* modelling methods (or half of it was). This was combined into a simulation that should behave similar to what the Cougar would in the real world, and from this a *PI*-controller was found.

## Chapter 5

# Conclusion

Comparing the Table 1.1 in Section 1.1-Project Description containing the list of tasks with Table 4.1 in Section 4-Discussion a overview of what was achieved is quickly achieved. The first thing is, that an implementation of the system was not achieved. With this being said, alot was done and a road map to get a full implementation if possible to get at this point.

The first issue that needed to be solved was to establish the control of the thruster and the communication to the top side. In Section 2.2 the electronics to achieved this was described. By using the technical documents of the Cougar XT system, a circuit was created which is able to create the same control signals as Saab Seaeye uses to control the SM-7 thruster. A microcontrolelr was chosen, which control the thruster and facilitate communication to the topside.

To develop the controller and analyse the system, a mathematical model of the ROV pitch manoeuvre was developed in Section 2.3. By identifying the forces acting upon the ROV as it changed the pitching angle and describing these an expression for the torque needed for any given angle was determined. By the use of the inertia tensor of the ROV system, the angular acceleration of the ROV system was described using the instantaneous torque. And from the knowledge of the angular acceleration, the displacement of the ROV was described. Finally, a drag force due to the angular velocity of the ROV was also included. The term were simplified significantly but gives a rough estimate of the actual drag which add a small dampening effect to the system.

To get a model of the thruster, a series of experiments were performed to identify the input/output relation, described in Section 2.4. This data was used in Section 3.1 to approximate a model using the Matlab toolbox *System Identification*. This effort was successful for the forward direction, while the backward direction yielded a very poor fit. However, by using the model of the forward direction for both directions, a model of the full ROV system was created in Section 3.2.

This model was used to build a *PI*-controller in Section 3.3. It was found the a *PI*-controller was able to control the system and maintain a steady state offset of  $\approx 0^\circ$  while working within the range of angle that the thruster allows.

## Chapter 6

# Perspective

In the following section, the potential for the thruster system will be discussed and the path of what needs to be done before the first incarnation of the pitch functionality can be tested will also be covered.

For the continued development of the pitch controller for the Cougar system, the first step that needs to be taken is to find the model of the thruster for the backwards direction. As it was not possible to get a very good description of the behaviour in this direction. As previously mentioned, there are two paths that can be tried, the first is to work with the data that has already been captured. Using this data a non-linear model of the thruster behaviour can be sought.

The advantage of doing this is, that there is no need to perform additional experiments in order to identify the thruster behaviour. This means an approximate function is achieved with less relative effort. Seeking a non-linear model was also found to improve the fit for the forward significantly, so this path has the promise of a much better estimation of the thruster behaviour.

The other possibility is to investigate the experimentation method, described in Section 2.4, and try and improve upon this. By improving the experiment, it is possible that better data could be used for the model estimation. By improving the experiment, it is meant to reduce the variables that can affect the result.

At the conclusion of the project there are a few areas that were found that could be improved. It was noticed that the data from the experiment did have significant noise, see Figure 3.2 in Section 3.1.1 as an example. One possible way to reduce this is to build an Anti-aliasing filter for the ADC which samples the load cell. The advantage of the hardware filter is that it functions on the analogue signals, meaning it is not subject to the same limitations that occur when working with filters in discrete time.

Another area that could be improved, is to find a load cell that can create measurements faster than 10 samples per second. While this will not help with the noise, it can improve the evaluation of the transient behaviour of the thruster. It would also make it easier to actually see the transient behaviour, as the step response of the thruster seems relatively fast.

With this, the next step is deploying the control system to the actual ROV, and here the electronics established in Section 2.2 are what would be used to control the thruster below the surface, but one area that has not yet been discussed is how the controller will



be handled by the topside. There are two possibilities, the first is to use a similar method as used for the thruster test a simple java program (used in Section 2.4). This has the advantage that no new hardware needs to be created and laptop or computer connected to the top side fibre optical MUX's would be able to control the thruster direction through a RS-232 interface.

The other possibility is to use a second Arduino (or any other microcontroller) to communicate with the thruster controller through the RS-232 interface. A circuit consisting of two 7-segment displays to show angle set point and two buttons to increase or decrease the angle could be implemented relatively easily. While requiring some more effort than a simple PC program, it has the advantage that it can be combined in a small control box and this box could be used by anyone quite intuitively.

One interesting factor which will not be known before the system is tested, is how the ROV will interact with the pitch control system. This is one factor which no amount of testing of the pitch system can identify. The risk here, is that the auto depth and auto heading systems of the ROV will notice something is incorrect as the ROV is angled. This could lead the ROV to behave in a manner that is not only unwanted, but could also be dangerous for the integrity of the ROV.

Even if the ROV controller behaves correctly, there is also the pitch controller which could react in unpredictable ways. As the ROV manoeuvres around the water column, the current and waves that move around the ROV can cause some swaying, and if the pitch controller misinterprets these motions, there is the possibility it will start to behave erratic. How these combinations will function, will not be found out, before the full system test is ready, it seems reasonable however, that a large basin of water is used, to reduce the chance anything critical breaks on the ROV.

A final consideration, by the equation of the torque generated by the ROV, the torque working against the pitching of the ROV. The maximum torque needed to pitch the ROV  $360^\circ$  can be found.

$$\tau_{360^\circ}^{\max}\left(\frac{\pi}{2}\right) = 419.93 \text{ Nm} \cdot \sin^2\left(\frac{\pi}{2}\right) + 442.80 \text{ Nm} \cdot \sin^2\left(\frac{\pi}{2}\right) = 862.73 \text{ Nm} \quad (6.1)$$

Equation 6.1 shows the maximum torque that the ROV generates as it is pitched. This peak is reached as the ROV is at  $90^\circ$ , from this, it can also be observed that the thruster needs to deliver at least a torque of  $862.73 \text{ Nm}$  to allow for full rotational pitching.

Comparing this with the SM-7 thruster, it was found through the experiments in Section 2.4 that the maximum force of the thruster is  $\approx 60 \text{ kg} \approx 589.2 \text{ N}$ . This means, using the calculation of force into torque, the distance the thruster should be from the centre of rotation can be found.

$$\tau = F \cdot r \rightarrow r = \frac{\tau}{F} \rightarrow r = \frac{862.73 \text{ Nm}}{589.2 \text{ N}} = 1.46 \text{ m}$$

This means, the thruster needs to be almost double the distance from the centre of the ROV if a full  $360^\circ$  pitch functionality is required. With this being said, it is probably a good safety feature, that no input to the thruster, can pitch the ROV in a full circle.

# References

- [1] NovAtel. *Attitude - Pitch/Roll/Yaw*. [Online]. Available from: <http://www.novatel.com/solutions/attitude/>. Accessed 8<sup>th</sup> January 2017.
- [2] Saab Seaeye. *Seaeye Cougar-XT*. [Online]. Available from: <http://www.seaeye.com/cougar-xt.html>. Accessed 13<sup>th</sup> October 2016.
- [3] Saab Seaeye Ltd. *Seaeye cougar technical manual book 2 of 2 - system 1471*. Product manual distributed with system, 2016.
- [4] Arduino. *Arduino Nano*. [Online]. Available from: <https://www.arduino.cc/en/Main/ArduinoBoardNano>. Accessed 24<sup>th</sup> October 2016.
- [5] Mean Well. *15W Single Output DC-DC Converter - SD-15 series*. [Online]. Available from: <http://www.meanwell.com/productPdf.aspx?i=51>. Accessed 25<sup>th</sup> October 2016.
- [6] Atmel. *ATmega328/P*. [Online]. Available from: [http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P\\_datasheet.pdf](http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_datasheet.pdf). Accessed 25<sup>th</sup> October 2016.
- [7] Ian Poole. *RS232 voltage levels & RS-232 Signals*. [Online]. Available from: [http://www.radio-electronics.com/info/telecommunications\\_networks/rs232/signals-voltages-levels.php](http://www.radio-electronics.com/info/telecommunications_networks/rs232/signals-voltages-levels.php). Accessed 25<sup>th</sup> October 2016.
- [8] RSS-systems. *SCHNITTSTELLEN TTL/RS232/RS485/CAN, 3V/5V/12V*. [Online]. Available from: <http://rss-systems.de/mcu-tools/mcu-bausaetze/ttl-rs232-rs485-can/index.php>. Accessed 25<sup>th</sup> October 2016.
- [9] The Physics Classroom. *Newton's Second Law*. [Online]. Available from: <http://www.physicsclassroom.com/class/newtlaws/Lesson-3/Newton-s-Second-Law>. Accessed 3<sup>rd</sup> November 2016.
- [10] J. Peraire & S. Widnall. *Lecture L26 -3D Rigid Body Dynamics: The Inertia Tensor*. [Online]. Available from: [https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16\\_07F09\\_Lec26.pdf](https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16_07F09_Lec26.pdf). Accessed 20<sup>th</sup> December 2016.
- [11] Maplesoft. *Rotation: Moment of Inertia and Torque*. [Online]. Available from: [http://www.maplesoft.com/content/EngineeringFundamentals/4/MapleDocument\\_30/Rotation%20MI%20and%20Torque.pdf](http://www.maplesoft.com/content/EngineeringFundamentals/4/MapleDocument_30/Rotation%20MI%20and%20Torque.pdf). Accessed 2<sup>nd</sup> November 2016.

- [12] Sarah Friedl. *Buoyancy: Calculating Force and Density with Archimedes' Principle*. [Online]. Available from: <http://study.com/academy/lesson/buoyancy-calculating-force-and-density-with-archimedes-principle.html>. Accessed 3<sup>rd</sup> November 2016.
- [13] Jon. *Tyngdeaccelerationen, g*. [Online]. Available from: [http://fysiklokalet.dk/index.phtml?con\\_id=55](http://fysiklokalet.dk/index.phtml?con_id=55). Accessed 4<sup>th</sup> November 2016.
- [14] Nasa. *The Drag Equation*. [Online]. Available from: <https://www.grc.nasa.gov/www/k-12/airplane/drageq.html>. Accessed 14<sup>th</sup> November 2016.
- [15] Det Norske Veritas. *Modelling and Analysis of Marine Operations*. [Online]. Available from: <https://rules.dnvgl.com/docs/pdf/DNV/codes/docs/2012-12/RP-H103.pdf>. Accessed 14<sup>th</sup> November 2016.
- [16] hyperphysics. *Torque Calculation*. [Online]. Available from: <http://hyperphysics.phy-astr.gsu.edu/hbase/torq2.html>. Accessed 25<sup>th</sup> December 2016.
- [17] National Instruments. *The Fundamentals of FFT-Based Signal Analysis and Measurement in LabVIEW and LabWindows/CVI*. [Online]. Available from: <http://www.ni.com/white-paper/4278/en/>. Accessed 27<sup>th</sup> December 2016.
- [18] Eric W. Weisstein. *Nyquist Frequency*. [Online]. Available from: <http://mathworld.wolfram.com/NyquistFrequency.html>. Accessed 27<sup>th</sup> December 2016.
- [19] National Instruments. *Aliasing*. [Online]. Available from: <http://zone.ni.com/reference/en-XX/help/370051M-01/cvi/libref/analysisconcepts/aliasing/>. Accessed 27<sup>th</sup> December 2016.
- [20] Steven W. Smith. *Moving Average Filters*. [Online]. Available from: [http://www.analog.com/media/en/technical-documentation/dsp-book/dsp\\_book\\_Ch15.pdf](http://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch15.pdf). Accessed 29<sup>th</sup> December 2016.
- [21] National Instruments. *What are Anti-Aliasing Filters and Why are They Used?* [Online]. Available from: <http://digital.ni.com/public.nsf/allkb/68F14E8E26B3D101862569350069E0B9>. Accessed 7<sup>th</sup> January 2017.
- [22] Industrial Controls. *Basics of PID Control (Proportional+Integral+Derivative)*. [Online]. Available from: <http://www.industrialcontrolsonline.com/training/online/basics-pid-control-proportionalintegralderivative>. Accessed 7<sup>th</sup> January 2017.
- [23] Control Tutorials for Matlab & Simulink. *Introduction: PID Controller Design*. [Online]. Available from: <http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID>. Accessed 7<sup>th</sup> January 2017.
- [24] The Engineering ToolBox. *Density and specific weight of water at temperatures ranging 0 - 100 C (32 - 212 F)*. [Online]. Available from: [http://www.engineeringtoolbox.com/water-density-specific-weight-d\\_595.html](http://www.engineeringtoolbox.com/water-density-specific-weight-d_595.html). Accessed 1<sup>st</sup> November 2016.
- [25] AmBrSoft. *Materials density table*. [Online]. Available from: [http://www.ambrsoft.com/CalcPhysics/Density/Table\\_2.htm](http://www.ambrsoft.com/CalcPhysics/Density/Table_2.htm). Accessed 2<sup>nd</sup> November 2016.

- [26] TutorVista. *Center of Mass Formula*. [Online]. Available from: <http://formulas.tutorvista.com/physics/center-of-mass-formula.html>. Accessed 3<sup>rd</sup> November 2016.

# Appendix A

## Calculations

### A.1 Simplified Cougar model

To get the inertia tensor, a simplified model, emulating the weight distribution of the real cougar, is drawn. The following contains the calculations used to create this model.

#### A.1.1 Volume & density of frame

The volume of the frame is used to create a block with an equal density to the ROV frame without buoyancy foam.

$$\begin{aligned} m_{air}^{ROV} &= 237 \text{ kg} & m_{water}^{ROV} &= 60 \text{ kg} \\ length_{frame} &= 1.495 \text{ m} & width_{frame} &= 1.006 \text{ m} \\ \rho_{water} &= 998.2 \text{ kg/m}^3 [24] \end{aligned}$$

First the water displacement of the frame is calculated. This is based on the ROV without lead.

$$M_{displaced} = m_{air} - m_{water} \Rightarrow m_{displaced} = 177 \text{ kg} \quad (\text{A.1})$$

The weight of the displaced water is converted to a displaced volume, based on the density of water.

$$V_{water} = \frac{m_{displaced}}{\rho_{water}} \Rightarrow V_{water} = 0.177 \text{ m}^3 \quad (\text{A.2})$$

The volume of the frame is equal to the volume of the water displaced. As the frame is simulated by a box, where the length and width is known, the height of it can be calculated.

$$height_{frame} = \frac{V_{frame}}{length_{frame} * width_{frame}} \Rightarrow height_{frame} = 0.118 \text{ m} \quad (\text{A.3})$$

The density of the frame can likewise be calculated.

$$\rho_{frame} = \frac{m_{air}}{V_{frame}} \Rightarrow \rho_{frame} = 1338.98 \text{ kg/m}^3 \quad (\text{A.4})$$

As all these calculations are based on the ROV without lead added, calculated mass of the lead in water from Equation A.8 is added to the mass of the ROV in air in order to get the density of the frame with lead on it.

$$\rho_{frame\ W/lead} = \frac{m_{air} + m_{water}^{lead}}{V_{frame}} \Rightarrow \rho_{frame\ W/lead} = 1740.85\ kg/m^3 \quad (A.5)$$

In Equation A.5 the density of the lead in water is used, so the calculation of the density for the frame is independent of the displacement of the lead, as this is already included in the calculation for the lead weight in water.

### A.1.2 Foam volume & density

In the following the volume of the foam and density of it is calculated. This is used in the simplified Cougar model.

$$\begin{aligned} m_{water}^{ROVW/Ofoam\ W/Olead} &= 60\ kg & m_{air}^{lead} &= 78\ kg \\ m_{air}^{ROVW/Ofoam\ W/Olead} &= 237\ kg & \rho^{lead} &= 11341\ kg/m^3\ [25] \\ m_{air}^{foam} &= 105\ kg \\ \rho_{water} &= 998.2\ kg/m^3\ [24] \end{aligned}$$

All calculations are based on the standard Cougar setup, however, as the ballast weight on the cougar had been removed during the project period, the weight of the ROV in water with the ballast attached and without foam is not known, it has to be calculated. The amount of lead was measured, and the added weight can be calculated as the following.

First the weight of the lead in water is needed. To do this, the displacement of the lead, and thereby the volume of it is needed.

$$V_{lead} = \frac{m_{lead}}{\rho_{lead}} \Rightarrow V_{lead} = 0.0069\ m^3 \quad (A.6)$$

$$m_{displacement} = V_{lead} * \rho_{water} \Rightarrow m_{displacement} = 6.87\ kg \quad (A.7)$$

$$m_{water}^{lead} = m_{air}^{lead} - m_{displacement} \Rightarrow m_{water}^{lead} = 71.13\ kg \quad (A.8)$$

From this, the weight of the cougar in water with lead, can be easily obtained and the displaced water of the cougar, with lead, both with and without foam can be calculated, from which the volume of the foam can be obtained.

$$m_{water}^{ROVW/lead\ W/Ofoam} = m_{water}^{ROVW/Olead\ W/Ofoam} + m_{water}^{lead} \Rightarrow m_{water}^{ROVW/lead\ W/Ofoam} = 131.13\ kg \quad (A.9)$$

$$m_{air}^{ROVW/lead\ W/Ofoam} = m_{air}^{ROVW/Olead\ W/Ofoam} + m_{air}^{lead} \Rightarrow m_{air}^{ROVW/lead\ W/Ofoam} = 315\ kg \quad (A.10)$$

$$V_{ROVW/lead\ W/Ofoam} = \frac{m_{air}^{ROVW/lead\ W/Ofoam} - m_{water}^{ROVW/lead\ W/Ofoam}}{\rho_{water}} \Rightarrow V_{ROVW/lead\ W/Ofoam} = 0.1842\ m^3 \quad (A.11)$$

The total displacement of the ROV is equal to total weight of the ROV with foam in air divided by the density of water, as at this point the weight of the water displaced is equal to the weight of the ROV.

$$V^{ROVW/lead\ W/foam} = \frac{m_{air}^{foam} + m_{air}^{ROVW/lead\ W/Ofoam}}{\rho_{water}} \Rightarrow V^{ROVW/lead\ W/foam} = 0.4208\ m^3 \quad (A.12)$$

From this the volume of the foam can be calculated.

$$V_{foam} = V^{ROVW/lead\ W/foam} - V^{ROVW/lead\ W/Ofoam} \Rightarrow V_{foam} = 0.2366\ m^3 \quad (A.13)$$

and the density of the foam can be calculated as well.

$$\rho_{foam} = \frac{m_{foam}}{V_{foam}} \Rightarrow \rho_{foam} = 443.79\ kg/m^3 \quad (A.14)$$

Finally, the height of the foam block can also be calculated.

$$height_{foam} = \frac{V_{foam}}{length_{foam} * width_{foam}} \Rightarrow height_{foam} = 0.157\ m \quad (A.15)$$

### A.1.3 Centre of Mass'

To model the moment of torque for of the Cougar, the axis of rotation of the cougar is needed, this will be the centre of mass of the Cougar. Further, the two forces treated, the weight of the ROV weighing it down, and the buoyancy modules dragging it upwards, both has a centre of mass, which is offset from the centre of mass of the complete ROV. These two centre of mass' is where those forces are acting upon, and as described in Section 2.3.1, the horizontal alignment between those three centres creates the moment of torque.

To calculate the centre of mass', a simplification of the cougar is created. First of all, only the torque around the  $Y_0$  axis, as illustrated in Figure 2.3, is of interest, as the model only describes pitching of the ROV. For this reason, the centre of mass' is calculated purely in 2-D. The concept is illustrated in Figure A.1.

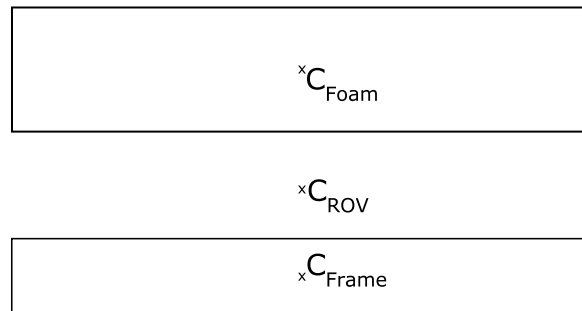


Figure A.1: Simplified model of Cougar

Both the bottom and top block (in Figure A.1) are treated as being of uniform density, and their length and width are equal to the dimensions of the ROV (the width is the axis which is being ignored). Due to treating the blocks as having uniform density, their centre of mass will be located in the centre of the shape, which means if the third axis was to be included it would be simple to calculate the location on this axis as well.

An important note is, that the system in will be treated as if submerged in water. This is important, because out of the water, the foam will not create buoyancy but instead also act as an increase in weight of the ROV. Further, the mass of the frame used is the mass of the frame as submerged in water, which was found in Equation A.9 to be  $m_{water}^{ROVW/lead\ W/Ofoam} = 131.13\text{ kg}$ . As the ROV is mutual buoyant the force exerted by the foam block is equal to the bottom block, but the force is in the opposite direction.

Calculating the centre of mass for both the foam block and frame block is needed as this is the centre of force, meaning it is where the force can be treated as being applied. As the density is uniform, the centre of mass is simply the geometric centre as well.

$$C_{foam}^x = length_{foam}/2 = 0.748\text{ m} \quad C_{foam}^y = height_{foam}/2 = 0.079\text{ m} \quad (\text{A.16})$$

$$C_{frame}^x = length_{frame}/2 = 0.748\text{ m} \quad C_{frame}^y = height_{frame}/2 = 0.059\text{ m} \quad (\text{A.17})$$

Both centres are calculated relative to there bottom left corner of each block. By implementing a reference frame at the bottom left corner of the frame block, the location centre of the foam block is shifted by the height of the ROV minus half the height of the block.

$$C_{ROV}^y = \frac{m_1 y_1 + m_2 y_2}{m_1 + m_2} \quad [26] \quad (\text{A.18})$$

Equation A.18 can be used to calculate the  $y$  location of the centre of mass of the whole ROV system. The  $x$  location of the ROV is equal to the two other blocks location. The mass of the buoyancy foam is treated as being equal to the frame.

$$C_{ROV}^y = \frac{m_{water}^{ROVW/lead\ W/Ofoam} C_{frame}^y + m_{foam} (ROV_{height} - C_{foam}^y)}{m_{water}^{ROVW/lead\ W/Ofoam} + m_{foam}} \quad (\text{A.19})$$

Where the mass of the frame and the height of the ROV is given by the following.

$$m_{water}^{ROVW/lead\ W/Ofoam} = 131.13\text{ kg} \quad ROV_{height} = 0.790\text{ m}$$

$$C_{ROV}^y = 0.385\text{ m}$$



# Appendix B

## DNV Drag Coefficients

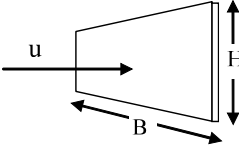
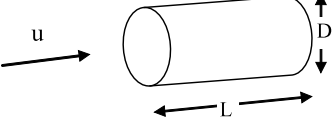
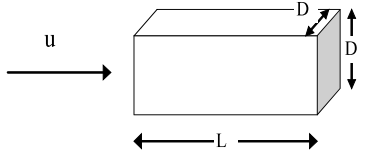
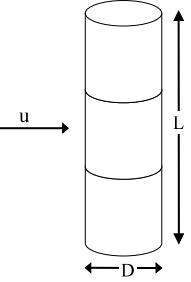
<b>Table B-2</b> Drag coefficient on three-dimensional objects for steady flow $C_{DS}$ . Drag force is defined as $F_D = \frac{1}{2}\rho C_{DS}Su^2$ . $S$ = projected area normal to flow direction [m <sup>2</sup> ]. $Re$ = $uD/\nu$ = Reynolds number where $D$ = characteristic dimension.			
Geometry	Dimensions	$C_{DS}$	
Rectangular plate normal to flow direction 	B/H		
	1 5 10 $\infty$	1.16 1.20 1.50 1.90	
		$Re > 10^3$	
Circular cylinder. Axis parallel to flow. 	L/D		
	0 1 2 4 7	1.12 0.91 0.85 0.87 0.99	
		$Re > 10^3$	
Square rod parallel to flow 	L/D		
	1.0 1.5 2.0 2.5 3.0 4.0 5.0	1.15 0.97 0.87 0.90 0.93 0.95 0.95	
		$Re = 1.7 \cdot 10^5$	
Circular cylinder normal to flow. 	L/D	Sub critical flow $Re < 10^5$	Supercritical flow $Re > 5 \cdot 10^5$
			$\kappa$
	2 5 10 20 40 50 100	0.58 0.62 0.68 0.74 0.82 0.87 0.98	0.80 0.80 0.82 0.90 0.98 0.99 1.00
		$C_{DS} = \kappa C_{DS}^{\infty}$ $\kappa$ is the reduction factor due to finite length. $C_{DS}^{\infty}$ is the 2D steady drag coefficient.	

Figure B.1: Table from DNV-RP-H103 [15]

# Appendix C

## Code

The following section contains the source code for the microcontrollers used during the project.

### C.1 Load cell monitor

---

```
1  int analogData;
2  int loadCellPin = A0;
3
4  void setup() {
5    Serial.begin(115200);
6  }
7
8  void loop() {
9    analogData = createSample(50);
10
11    Serial.println(analogData);
12
13    delay(2); //delay 2 ms
14  }
15
16  int createSample(int avgN){
17    unsigned long tempReading = 0L;
18    for(int i = 0; i < avgN; i++){\
19      tempReading += analogRead(loadCellPin); //Analog read takes ~0.0001 sec
20    }
21    return tempReading/avgN;
22  }
```

---

Listing C.1: Load cell monitoring code

### C.2 Thruster Test Controller

The following code is the arduino code used for the thruster during the test to acquire the thrust data in the test described in Section 3.1-SM-7 Model.

---

```
1
2  int PWM_PIN = 5;
3  int DIR_PIN = 2;
4  int DIRNOT_PIN = 9;
5  String rcv_data;
```

```

6  int pwm_Percent;
7  bool dir;
8
9  void setup() {
10   setPwmFrequency(PWM_PIN, 1024);
11   pwm_Percent = 100;
12   analogWrite(PWM_PIN, pwm_Percent * 2.55);
13   Serial.begin(9600);
14   pinMode(DIR_PIN, OUTPUT);
15   pinMode(DIRNOT_PIN, OUTPUT);
16   dir = false;
17   setDir();
18 }
19
20 void loop() {
21   while (Serial.available() > 0) {
22     int tempChar = Serial.read();
23
24     if (isDigit(tempChar)) {
25       rcv_data += (char) tempChar;
26     }
27
28     else if (tempChar == 'F' || tempChar == 'B') {
29       if (tempChar == 'F') {
30         dir = true;
31       } else if (tempChar == 'B') {
32         dir = false;
33       }
34       pwm_Percent = 100;
35       analogWrite(PWM_PIN, pwm_Percent * 2.55);
36       setDir();
37     }
38     else if (tempChar == '\n') {
39       if (rcv_data.length() > 0) {
40         pwm_Percent = 100 - rcv_data.toInt();
41         analogWrite(PWM_PIN, pwm_Percent * 2.55);
42       }
43       Serial.println("PWM: ");
44       Serial.println(pwm_Percent);
45       Serial.println("DIR: ");
46       Serial.println(dir);
47       rcv_data = "";
48     }
49   }
50
51 }
52
53 void setDir() {
54   if (dir) {
55     digitalWrite(DIRNOT_PIN, LOW);
56     digitalWrite(DIR_PIN, HIGH);
57   } else if (!dir) {
58     digitalWrite(DIR_PIN, LOW);
59     digitalWrite(DIRNOT_PIN, HIGH);
60   }
61 }
62
63
64 // -----

```

```

65 // PWM FREQUENCY PRESCALER
66 // -----
67 // Written by Kiwisincebirth 2014
68 // Revised by MacTester57 to allow correct PWM frequencies (confirmed with
69 // oscilloscope). January 2015
70 // - prescale variable: replaced uint8_t with uint16_t data type (fixes bug,
71 // which did not allow frequencies < 492Hz)
72 // - mode variable: replaced hex format with binary to make it more readable,
73 // if compared with bit tables in the 32U4 manual
74 // - added comments
75 //
76 // This function was extracted from:
77 // https://github.com/kiwisincebirth/Arduino/tree/master/libraries/PWMFrequency
78 void setPwmFrequency(int pin, int divisor) {
79     byte mode;
80     if (pin == 5 || pin == 6 || pin == 9 || pin == 10) {
81         switch (divisor) {
82             case 1: mode = 0x01; break;
83             case 8: mode = 0x02; break;
84             case 64: mode = 0x03; break;
85             case 256: mode = 0x04; break;
86             case 1024: mode = 0x05; break;
87             default: return;
88         }
89         if (pin == 5 || pin == 6) {
90             TCCR0B = TCCR0B & 0b11111000 | mode;
91         } else {
92             TCCR1B = TCCR1B & 0b11111000 | mode;
93         }
94     } else if (pin == 3 || pin == 11) {
95         switch (divisor) {
96             case 1: mode = 0x01; break;
97             case 8: mode = 0x02; break;
98             case 32: mode = 0x03; break;
99             case 64: mode = 0x04; break;
100            case 128: mode = 0x05; break;
101            case 256: mode = 0x06; break;
102            case 1024: mode = 0x07; break;
103            default: return;
104        }
105     }
106     TCCR2B = TCCR2B & 0b11111000 | mode;
107 }

```

Listing C.2: Thruster controller code

## Appendix D

### File Structure

Path	Content
.Files/Data/Backwards/*.txt	Raw data from backwards thruster test.
.Files/Data/Forwards/*.txt	Raw data from forwards thruster test.
.Files/Programs/loadCell/loadCell.ino	Arduino sketch used for the load cell monitor.
.Files/Programs/ThrusterController/SM7TestDriver.ino	Arduino sketch used for the thruster controller.
.Files/Programs/ThrusterControlClient/SUBC_TS.jar	Java application used to control the thruster during the test.
./Files/Matlab/Subssys_feedbackV2_NoController.slx	System simulation without controller for Simulink.
./Files/Matlab/Subssys_feedbackV2_NoController2015B.slx	Compatible for previous version.
./Files/Matlab/Subssys_feedbackV4_Controller.slx	System simulation with controller for Simulink.
./Files/Matlab/Subssys_feedbackV4_Controller2015B.slx	Compatible for previous version.
./Files/Matlab/readTestData.m	File that reads the raw data files from the thruster test and creates workspace variables of it

**Table D.1:** File table of *Data.zip* file