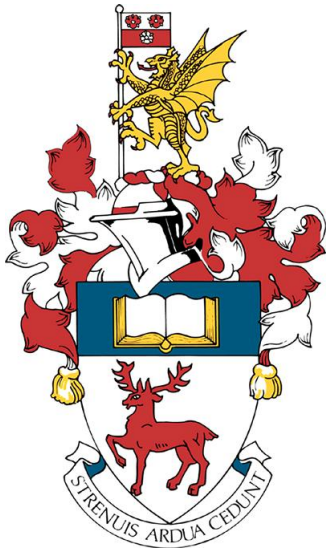
UNIVERSITY OF SOUTHAMPTON

UAV Smart Wing - active control of wing lift to suppress short-period oscillations

Author:
Bhagyesh GOVILKAR

Supervisor:
Dr. T. Glyn THOMAS



This report is submitted in partial fulfillment of the requirements for the MEng Aeronautics and Astronautics, Faculty of Engineering and the Environment, University of Southampton

Word Count : 9254
April 24, 2018

Abstract

Flight trajectory control has been discussed in several papers over the past few decades. Controlling the trajectory of flight has many benefits. A stable platform is easier and safer to operate. Any data being collected in flight will also be of higher quality because it is not being affected by external disturbances. Perhaps the greatest benefit of being able to control the flight path is that rejecting disturbances reduces the wear and tear on the aircraft. Failure due to fatigue is less likely as a result. The frequency of routine inspections can be reduced which cuts the downtime of the aircraft. The eventual "write-off" is also postponed. The economic value of such research is therefore very high.

In this report, a controller will be designed which can effectively reject disturbances in the wing lift. This will give the user better control over the aircraft and the operation of the aircraft will be made safer. A simulation model will be designed in Simulink which will attempt to approximate the plant before it is built. Consequently, the simulation optimised parameters will be applied to the actual system and will be tested in a wind tunnel. Any corrections that must be made to the parameters post testing will be outlined. Finally, a comparison will be made between the final parameters and the simulation parameters to realise any flaws within the model. This will be helpful to design a similar control system without a plant ready for testing.

Declaration

I, Bhagyesh Govilkar declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research. I confirm that:

1. This work was done wholly or mainly while in candidature for a degree at this University;
2. Where any part of this thesis has previously been submitted for any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. None of this work has been published before submission.

Acknowledgements

I would like to acknowledge my supervisor, Dr T. Glyn Thomas who provided me with guidance every step of the way and supported my ideas which enabled me to complete this project.

I would also like to thank my coursemate and friend Aaron Byrne, who provided assistance in building the parts required for this project...

List of Abbreviations

SPO	**S**hort **P**eriod **O**scillation
PID	**P**roportional **I**ntegral **D**erivative
I2C	**I**nter-**I**ntegrated **C**ircuit
SPI	**S**erial **P**eripheral **I**nterface bus

Physical Constants

Density of air	$\rho = 1.225\,\mathrm{kg/m^3}$ (at sea level at $15^\circ C$)
Density of water	$\rho_w = 1000\,\mathrm{kg/m^3}$
Gravitational acceleration	$g = 9.81\,\mathrm{m/s^2}$ (at the surface of earth)

List of Symbols

ω_n	natural frequency	Hz
ζ	Damping ratio	
K_p	Proportional gain	
K_i	Integral gain	
K_d	Derivative gain	
L	Lift	N
D	Drag	N
ρ	Damping ratio	kgm^{-3}
V	Airspeed	ms^{-1}
S	Planform Area	m^2
C_L	Coefficient of Lift	
C_p	Coefficient of Pressure	
Δh	Difference in fluid height m	
p_0	Stagnation/Total pressure	Pa
p	Static pressure	Pa
δ, θ	Flap deflection angle	degrees
α	Angle of Attack	degrees
N	Size of sample window	
f	Sample rate	Hz
$w_{1,2,3}$	Lift estimator parameters	
t_i	Integral/reset time	
t_d	Derivative time constant	
τ, T	Time constant	s

Contents

Chapter 1

Introduction

Unmanned Aerial Vehicles are becoming increasingly mainstream. With faster and more efficient ways of manufacturing such as 3D printing gaining momentum, this will only serve to bolster the emerging market.

There are several contemporary and potential applications of UAVs: they are widely used in agriculture to give farmers a detailed picture of the health of their farm, the amount of resources such as pesticides required and they can help cut down on labour costs; consumers and photographers are keen on buying products from UAV manufacturers such as DJI because these devices enable people to capture images/videos from extreme heights. The defence sector are using UAVs mostly to carry out surveillance and reconnaissance activities, a handful are even being used for offensive operations such as the MQ-9 Reaper. Commercial services such as Amazon Inc. have been known to conduct research in this technology. The company has an ambitious goal: using UAVs to transport goods to their customers.

With this increasing usage of UAVs, there is a strong need for a control system that can give the user a stable, reliable and robust aircraft. The aim is to reduce the frequency of UAVs crashing, increase the life-span and reduce the maintenance costs.

In this project, I will be looking specifically at fixed wing UAVs and to increase the speed of the response to disturbances in the lift. A fast response may be helpful in events such as a "short-period oscillation". This is one of the two longitudinal dynamic modes that occur on every aircraft, the other being a "phugoid oscillation". Short-period oscillations are characterised by the rapid decay and high frequency. A typical SPO can settle within a second and occurs at 1-2Hz. A pilot can induce a SPO by a sharp pitch input. They can also naturally occur due to a gust (impulsive increase/decrease in airspeed and/or angle of attack).

A UAV control system that is not tested in gusts is vulnerable to growing instability. It can act to reinforce the oscillation instead of cancelling them. Even if the system is proven to not introduce dynamic instabilities, the peak loading on the wing during a SPO event can test the structural limits of the aircraft making structural failures more likely. While this is not economically ideal, a more pressing concern is the health and safety risk created by such an event.

In this project, my goal will be to design a control system that minimises the effects of rapidly changing external conditions such as the airspeed in a gust in order to make UAVs safer to operate and also more economical.

1.1 Literature review

The knowledge we currently have and the applications of this knowledge need to be established first to identify the areas in which we lack an understanding and where improvements and optimisations can be made. In this chapter, I intend to do exactly that. By the end of this chapter, it will be clear that there has been little development in this area.

1.1.1 Phugoid suppression

It is worth looking at how we deal with phugoid oscillations first since suppressing phugoid oscillations has been well documented. This is achieved through the use of a Pitch Attitude Controller and has been described by Etkin [2]. More recently, a pitch-rate feedback control system was experimented with using an integral-separated PID controller [4]. In this controller, an error threshold is set below which the integral term in the PID calculation will be eliminated. This essentially makes it a PD controller until the pre-set threshold is crossed. The advantage of this is to limit the instability that the integral term brings but retain its tendency to remove steady-state errors. In both cases, the control system did well to suppress Phugoid oscillations and managed to weaken SPOs but relied on gyroscopes or other attitude sensors. This meant that the control system took action AFTER the flight dynamics caused a deviation in the pitch rate/angle from the nominal. This model is currently the state-of-the-art, but includes an intrinsic lag between the onset of the oscillations and the corrective action of the controller.

There has been little development in designing a control system that detects the disturbance at the source and takes action BEFORE the flight dynamics can cause an appreciable change in attitude. In this project, I will aim to achieve a control system that does not wait for a large change in attitude to occur before acting. This implies that the airflow around the wing will be controlled and the control of the flight trajectory is merely a consequence.

1.1.2 Aeroelastic control in wind turbines

Wind turbine blades are comparable to aircraft wings since both are lift generating devices and both run into similar issues. I looked at wind turbines and how active control systems are being used to improve performance, safety and robustness of the blades. These are highly relevant to what I am trying to achieve in this project because all these studies fall under the category of flow control and to control the aerodynamic forces on a wing is also controlling the flow of air past the wing.

There are multiple studies that look at aeroelastic control of wind turbine blades using trailing edge flaps and load sensors for input. There were two issues these studies dealt with: to suppress flutter[7], an aeroelastic phenomenon in which a structure in a fluid flow undergoes simple-harmonic motion; and to delay fatigue of the blades[8] by reducing the cyclic loading. The first issue is a safety concern while the second issue is about the economic sustainability of the wind turbines. Replacing large turbine blades can be difficult

and expensive because of their size. These turbines can also be located in challenging environments like the North Sea to complicate the logistics even further. Therefore, a control system that can elongate the life of a single blade is highly sough-after.

These studies have shown great success. Experiments showed that under gusting environments, settling times can be reduced by 50%. In turbulence, the standard deviation of the load oscillation was reduced by 30%. In theory, this problem is quite similar to managing longitudinal dynamic modes in aircraft and the benefits are also similar. In both cases, simple harmonic motion is causing excessive loading on the wing or blade and reducing the life of the structure. Positive feedback can cause a disaster and is potentially a safety hazard. Performance of the system is degraded by these oscillations and this has an effect on the operational costs.

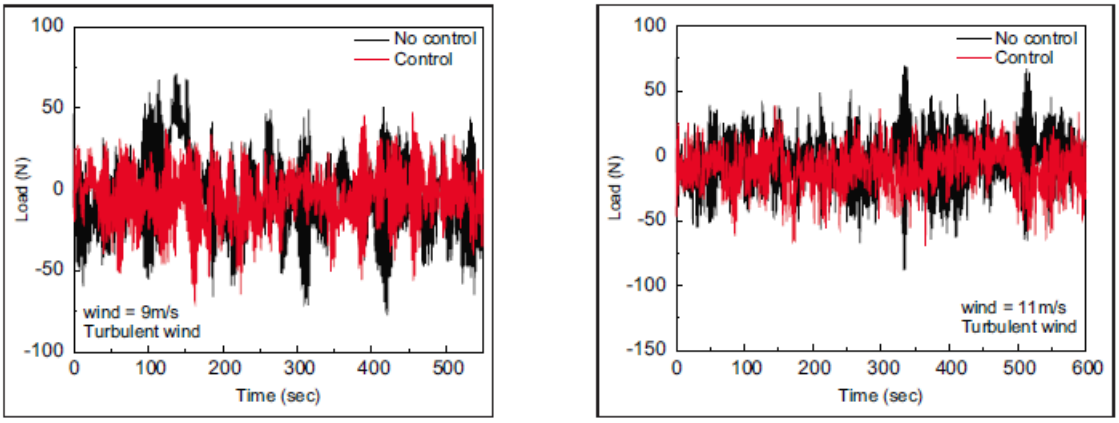


FIGURE 1.1: The load variation has shown decrease in both tests [8]

This review of literature will help me set realistic objectives for this project. I should be a able to achieve similar results to the study discussed above.

1.1.3 Aims and objectives

I will aim to design a control system that manages to suppress the rapid change in lift production as may be seen by the wing through a gust.

The control system should also advance the settling time of the system compared to an existing control system that settles an SPO in 0.263 seconds. I will aim to settle the lift within 50% of that time.

The amplitude of the oscillation in lift should be smaller in the gust with the control system active than it would be without it.

Lastly, I will attempt to use first principles modelling to determine the optimal PID gains and design the control system. Then, based on wind tunnel data, I will use heuristic techniques driven by collected data to further improve the performance. Eventually, I will use trial and error to fine tune the controller gains for my plant. Comparing the gains determined by Simulink to the optimal gains determined by testing will verify whether the first principles model was valid. If not, I will be exploring the possible flaws in the model. This will be helpful for designing a control system for UAVs whose plants have not yet been constructed.

Chapter 2

Theory

In this section, relevant theory that has been well established will be outlined. It may be helpful to first understand to understand why SPOs occur and how control of wing lift can help solve the problem.

2.1 Flight dynamics

This section will outline flight dynamics very briefly to help understand the mechanism of longitudinal dynamic modes such as a SPO.

I will use the longitudinal equations of motion as a starting point :

$$\begin{bmatrix} \Delta\dot{u} \\ \dot{w} \\ \dot{q} \\ \Delta\dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\mathring{X}_u}{m} & \frac{\mathring{X}_w}{m} & 0 & -gcos\theta_0 \\ \frac{\mathring{Z}_u}{m-\mathring{Z}_{\dot{w}}} & \frac{\mathring{Z}_w}{m-\mathring{Z}_{\dot{w}}} & \frac{\mathring{Z}_q+mU_\infty}{m-\mathring{Z}_{\dot{w}}} & \frac{-mgsin\theta_0}{m-\mathring{Z}_{\dot{w}}} \\ \frac{1}{I_y}\left[\mathring{M}_u + \frac{\mathring{M}_{\dot{w}}\mathring{Z}_u}{m-\mathring{Z}_{\dot{w}}}\right] & \frac{1}{I_y}\left[\mathring{M}_w + \frac{\mathring{M}_{\dot{w}}\mathring{Z}_u}{m-\mathring{Z}_{\dot{w}}}\right] & \frac{1}{I_y}\left[\mathring{M}_q + \frac{\mathring{M}_{\dot{w}}(\mathring{Z}_q+mU_\infty)}{m-\mathring{Z}_{\dot{w}}}\right] & -\frac{M_{\dot{w}}mg\sin(\theta_0)}{I_y(m-Z_{\dot{w}})} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ w \\ q \\ \Delta\theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta\mathring{X}_c}{m} \\ \frac{\Delta\mathring{Z}_c}{m-\mathring{Z}\dot{w}} \\ \frac{\Delta\mathring{M}_c}{I_y} + \frac{\mathring{M}_{\dot{w}}}{I_y}\frac{\Delta\mathring{Z}_c}{m-\mathring{Z}\dot{w}} \\ 0 \end{bmatrix} \quad (2.1)$$

The derivations of these equations are fairly straightforward but quite lengthy and can be found in virtually any flight mechanics/dynamics textbook [2]. The dynamics of flight (in the longitudinal direction) are heavily influenced by physical and geometric parameters such as the mass and the second moments of inertia around the y-axis (lateral direction) which influences the rotational kinematics of the aircraft. They are also influenced by the aerodynamic derivatives (e.g. $\mathring{X}_{u,\dot{w}}$, $\mathring{Z}_{u,w,q,\dot{w}}$, $\mathring{M}_{u,w,q,\dot{w}}$).

These equations can be rewritten in the form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}+\mathbf{B} \quad (2.2)$$

Where x is the state vector, **A** is the system matrix (constant) and B is a vector of control forces and moments which can be considered 0.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \quad (2.3)$$

This is a first order ODE which means that the solutions are in the form of

$$\mathbf{x} = \mathbf{x}_0 e^{\lambda t} \quad (2.4)$$

$$\lambda \mathbf{x}_0 = \mathbf{A}\mathbf{x}_0 \tag{2.5}$$

$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{x}_0 = 0 \tag{2.6}$$

Non-trivial solutions occur when $det(\mathbf{A} - \lambda \mathbf{I}) = 0$. Solving this equation for any given aircraft will give a quartic equation (4th order polynomial) which when solved will give two pairs of complex conjugates. One of the pairs corresponds to the phugoid mode while the other one corresponds to the SPO.

For example, we can consider the Navion aircraft [1]:

$$\mathbf{A} = \begin{bmatrix} -0.09148 & 0.04242 & 0 & -32.17 \\ 10.51 & -3.066 & 152 & 0 \\ 0.2054 & -0.05581 & -2.114 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.7}$$

Computing the eigenvalue of this matrix: The matrix produces two pairs of complex

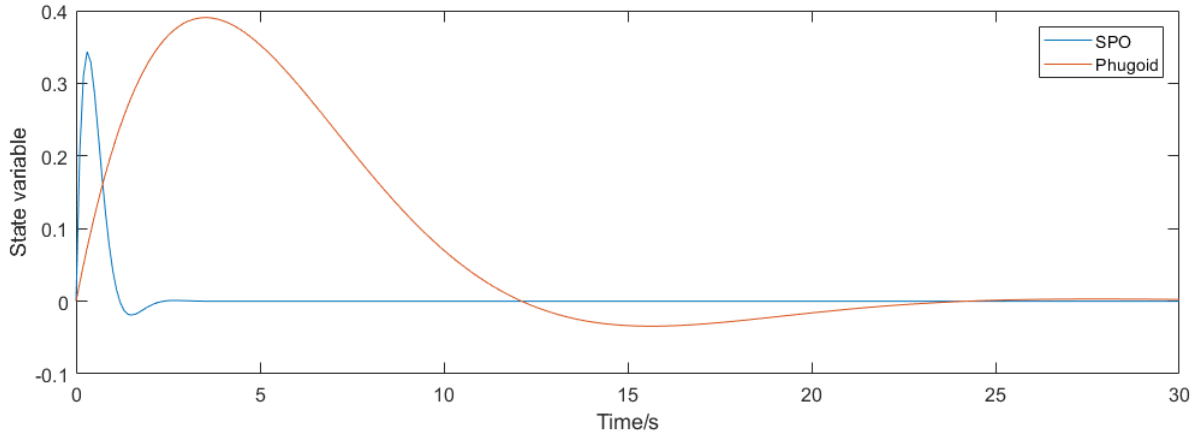


FIGURE 2.1: MATLAB output

conjugate eigenvalues. The first pair $\lambda_{1,2} = -2.4352 \pm 2.6461i$ is the SPO mode. The decay rate (the real part) is much greater than the other pair $\lambda_{3,4} = -0.2006 \pm 0.2593i$ and so is the frequency (coefficient of i).

We considered the control vector **B** to be 0 here. In other words, this only applies if no control forces are applied. However, when my control system is active, vector **B** will not be 0. The controller will adjust this vector (specifically the third element) in order to oppose any disturbance including a dynamic mode. From equation 2.1, we can see that the aerodynamic derivatives relating to the pitching moment on the aircraft affect the pitch rate ($\dot{q}$). Pitching moment is influenced mostly by the aerodynamic contributions from the wing and the tailplane. Thus, controlling lift is a way of mitigating longitudinal dynamic modes.

2.2 SPO regulations and qualification

The United States of America's Federal Aviation Authority regulates SPOs in FAR Part 23.181 [5] as follows: Any short period oscillation not including combined lateral-directional

oscillations occurring between the stalling speed and the maximum allowable speed appropriate to the configuration of the airplane must be heavily damped with the primary controls -

(1) Free; and

(2) In a fixed position.

United Kingdom's Civil Aviation Authority regulates SPOs in a very similar manner and can be found in CAP482 S181.

The quality of handling is largely determined by pilot opinion. This is visualised by pilot opinion charts such as the one shown below.

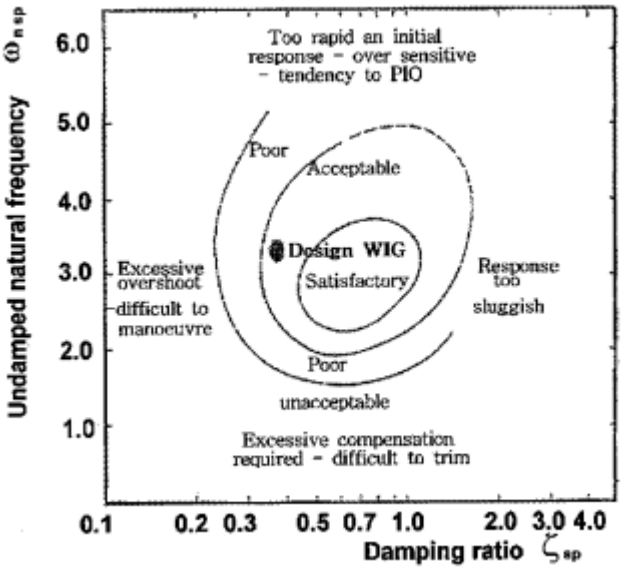


FIGURE 2.2: Short-period pilot opinion contours [3]

From SPO approximation [6] we can determine the natural frequency and damping ratio:

$$\omega_n = \sqrt{\mathring{Z}_w \mathring{M}_q + \mathring{M}_w \mathring{Z}_q} \tag{2.8}$$

$$\zeta = \frac{\mathring{M}_q + \mathring{Z}_w}{\omega_n} \tag{2.9}$$

Therefore, simply by adjusting the aerodynamic derivatives, it is possible to conform to the regulations. The regulations do not make the existence of a control system with a SPO dedicated model mandatory. This may explain the lack of development in this field. Thus, to fill this gap I will be developing a system that will focus on minimising the impact of SPOs. It can be difficult to test a control system directly on an aircraft because flight tests are more expensive, more complicated to run and pose some health and safety risks. It will be much more convenient and safer to run wind tunnel tests first. Therefore, the system will not be tested in an actual SPO. However, it is safe to assume that if the wing lift is being damped by the control system, a SPO will be suppressed during flight because the change in pitch angle is a result of the increased wing lift as discussed in the last section.

2.3 Trailing edge flap

Flaps are typically used as high-lift devices deployed in conditions where the airspeed is low. Increased lift is desirable to minimise the distance to take off or to achieve slower and steeper approaches during landing because extending flaps will lower the stall speed of the aircraft. The principle of operation is quite straightforward. Deflecting the flap downwards will increase the C_L and deflecting it upwards will decrease it. This in turn will change the lift produced:

$$L = \frac{1}{2}\rho V^2 S C_L \tag{2.10}$$

For a plain flap, the increase in lift can also be understood by realising that the airspeed under the wing will be slowed down when the flap is deflected downwards. Therefore, some of the dynamic pressure will be converted to static pressure increasing the pressure under the wing and consequently the lift. Deflecting the flap upwards will have the opposite effect: the speed at the top of the of the airfoil will be slowed down which will increase the static pressure here. This increased pressure acts opposite to the direction of positive lift and thus will reduce the size of the lift or it could produce negative lift.

A plain flap can both increase and decrease lift which will be useful to regulate lift through a gust.

2.4 PID controller

While PID controller (or it's variations) is not the only controller in existence, it is by far the most commonly used controller in industrial applications. The reason for it's popularity is most likely because it is easy to implement and gives the user the ability to fine tune the performance of the control system. For example, if the user desires a fast but less robust (allows overshoot) system, he/she can increase the integral gain.

By adjusting the gains, you can change the rise time, peak time, settling time, percentage overshoot and the steady state error (if applicable).

A PID output is the sum of the proportional, integral and derivative outputs. The proportional output is proportional to the error. The integral output is proportional to the integral of the error with respect to time. And finally, the derivative output is proportional to the rate of change of error. The following equation describes the output of a PID controller in the time domain.

$$output = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{2.11}$$

Or in the laplace domain:

$$\mathcal{L}[output] = K_p + \frac{K_i}{s} + K_d s \tag{2.12}$$

Chapter 3

Instrumentation

3.1 Computer and programming language

At the moment, there are two options for an embedded system: Arduino and Raspberry Pi.

The Arduino uses an Atmel ATmega 2560 processor with a clock speed of 16MHz. The program for the Arduino is written in a subset of the C/C++ programming language on a desktop computer and copied to the onboard flash memory over USB typically. The Arduino IDE can be used to compile and write the executable to the flash memory. It also allows for serial communication with the device which is useful for debugging.

The Raspberry Pi 3 has a 4-core 1.2GHz Broadcom BCM2837 64bit CPU running at 1.2 GHz which is much faster than the Arduino Mega. The extra processing power allows the user to use the Raspberry Pi as a full fledged computer. The programs can even be written on the device itself. However, it will be easier to program on a desktop and push the code to the Raspberry Pi over WiFi using the Secure Shell (SSH) protocol.

The Raspberry Pi also gives the user a choice of programming languages to use unlike the Arduino. Python is most commonly used to code for the Raspberry Pi. However, it would be more suitable to use a compiled language like C/C++ rather than an interpreted language like Python. While C can be difficult and more time consuming than Python to write and debug, it does run faster. The table below [9] shows the results from a Mandelbrot Set test (a recursive algorithm) for Python 3 and C++ with the g++ compiler.

source	*secs*	*mem*	*gz*	*cpu*	*cpuload*
*Python*3	225.24	15736	688	899.25	100%100%100%100%
$C++g++$	1.64	27636	1002	6.51	100%99%100%99%

Speed is important in this project since the objective is to minimise the rise times and settling times. Thus C++ executed on the Raspberry Pi 3 is the most suitable option.

3.2 Pressure sensors

This is one of the most important design decision for this project because they determine the ability to detect a change in airflow and lift (process variable) which is essential for a closed-loop feedback system.

There are a variety of sensors to choose from. There are absolute pressure sensors that simply measure the gauge or absolute pressure of air at a single point. Then there are pressure sensors that measure differential pressure. These have two ports and the sensor will measure the difference in pressure the two ports are subjected to.

To calculate the lift on the wing, we will need to know the pressure distribution along the top and the bottom of the wing. A single differential pressure sensor can tell the difference in pressure at a certain chord position on the wing. For the same information, two absolute pressure sensors will be required. Furthermore, the airspeed in the wind tunnel will need to be measured. Using a differential sensor, one port can be subjected to the total/stagnation pressure from a pitot probe, and the second port will be subjected to the static pressure. The sensor will instantly give the difference between the two pressures which is the dynamic pressure. The airspeed can then be easily calculated using Bernoulli's principle:

$$V = \sqrt{\frac{2(p_0 - p)}{\rho}} \tag{3.1}$$

Thus, for convenience a differential type sensor will be used. TE connectivity makes a sensor called MS4524DO in both differential and absolute types. There is also the option to choose between the I2C and SPI data transmission buses.

There are merits to using both buses. The I2C bus is very common and many devices use it to communicate with a 'master'. The bus typically runs at 100kbps but can reach 400kbps on fast mode. It uses a data line (SDA) and a clock line (SCL) and is therefore sometimes called 'two wire interface' (TWI).

SPI on the other hand runs much faster at around 10Mbps. The interface requires a clock line (SCLK), a MISO line (Master in Slave out), sometimes a MOSI line (Master out Slave in) and a slave select pin.

The slave select pin allows communication with a specific device on the line. In this case, we will have four pressure sensors but we will need to talk to one sensor before moving on to the next.

I2C relies on the device address. Each I2C device is given a specific address which needs to be provided before data transmission can happen. All available I2C MS4525DO sensors have the address 0x28 which creates a conflict. The I2C bus will not know which sensor to communicate with because they're all at the same address. A 'switching' mechanism will be needed to make this option viable (discussed in 4.4).

The most suitable bus for this application is obviously SPI. However, there is a severe lack of availability of these sensors at a reasonable price. A compromise in speed had to be made by choosing the I2C sensor which are widely available. However, for my purpose the speed exceeds Nyquist criterion (test described in section 3.4) so the lower bus speed will not be an issue.

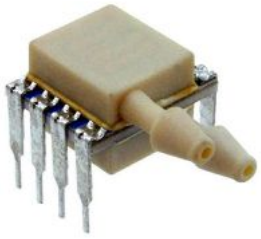


FIGURE 3.1: Differential type MS4525DO sensor

3.3 Sensor calibration

The pressure sensors mentioned in section 3.2 need to be calibrated against known values of pressures before they can be used. To achieve this, I used the principles of a manometer. The difference in pressure heads at the two ends of the tube is proportional to the difference in pressure.

$$\Delta P = \rho_w g \Delta h \tag{3.2}$$

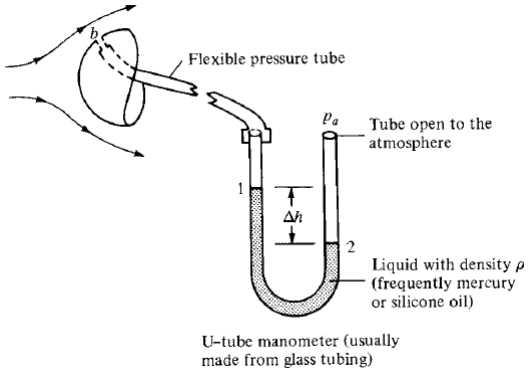


FIGURE 3.2: Diagram illustrating principle of manometer and how the fluid level changes when the ends are subjected to different pressures [11]

Height differences of -10cm,-5cm,0cm,5cm and 10cm were created by raising the level of water in a PVC tube. Both ends were connected to the ports of the sensors to give pressure differentials of $0Pa$, $\pm 490.5Pa$ and $\pm 981Pa$. The pressure sensors output the number of decimal counts. Results were collected for each sensor and are plotted below. These allowed me to convert the sensor output to pressure in pascals. The test results are shown in the figure 3.3 overleaf.

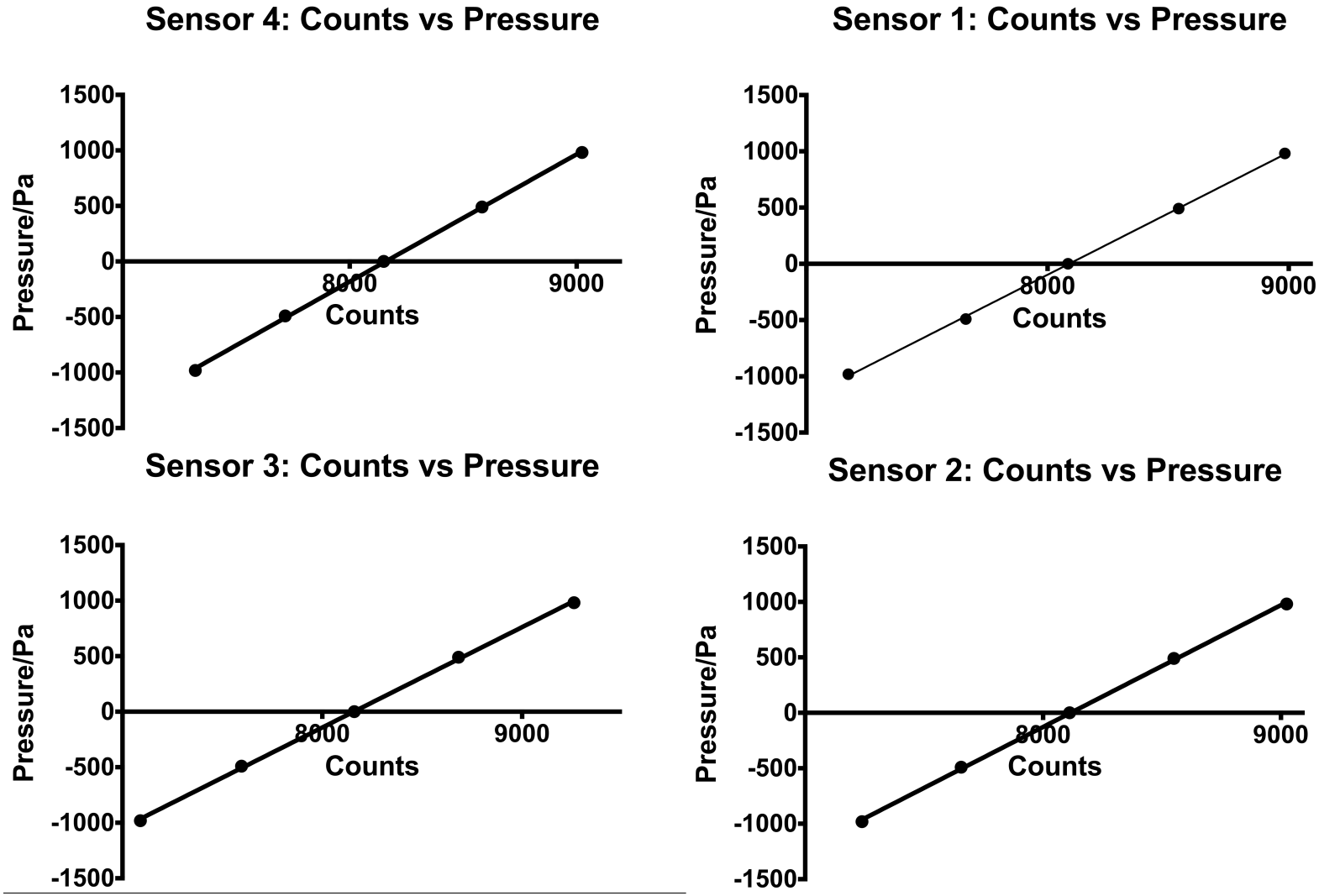


FIGURE 3.3: Counts vs Pressure

3.4 Data filtering

The aim of this section is to filter the data that comes from the pressure sensor so that the noise of values fed into the controller is minimised. The speed of sampling also needs to be considered especially since four pressure sensors are sharing a single I2C line. There are many choices for data filters.

A median filter takes a certain number of samples and outputs the median. This is useful when a sensor is known to output extreme values frequently. A common application of this filter is in ultrasonic range finders which work by sending out an ultrasonic pulse and receiving the reflection. The time between the pulse being sent and received is measured. The speed of sound at sea level is a constant. Thus, the distance can be calculated. Often, a sent pulse is not received and detected by the sensor which causes a timeout. In such an event, the sensor output can be extreme. A median filter however will completely disregard this value. In this project, the sensor is highly unlikely to output nonsensical values. Therefore, the use of median filter is not appropriate.

A simple average filter on the other hand may be much more useful. The operation is quite simple. A certain number of samples is taken and averaged. The intensity of noise is minimised by increasing the window (number of samples taken).

According to the Nyquist-Shannon theorem[12], "If a function x(t) contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart". In other words, the sampling frequency must be more than twice the SPO frequency. As mentioned earlier, SPOs occur at around 2Hz. Therefore, the sampling frequency must be at least more than 4Hz.

I measured the sample time as a function of the window size and the results are plotted below.

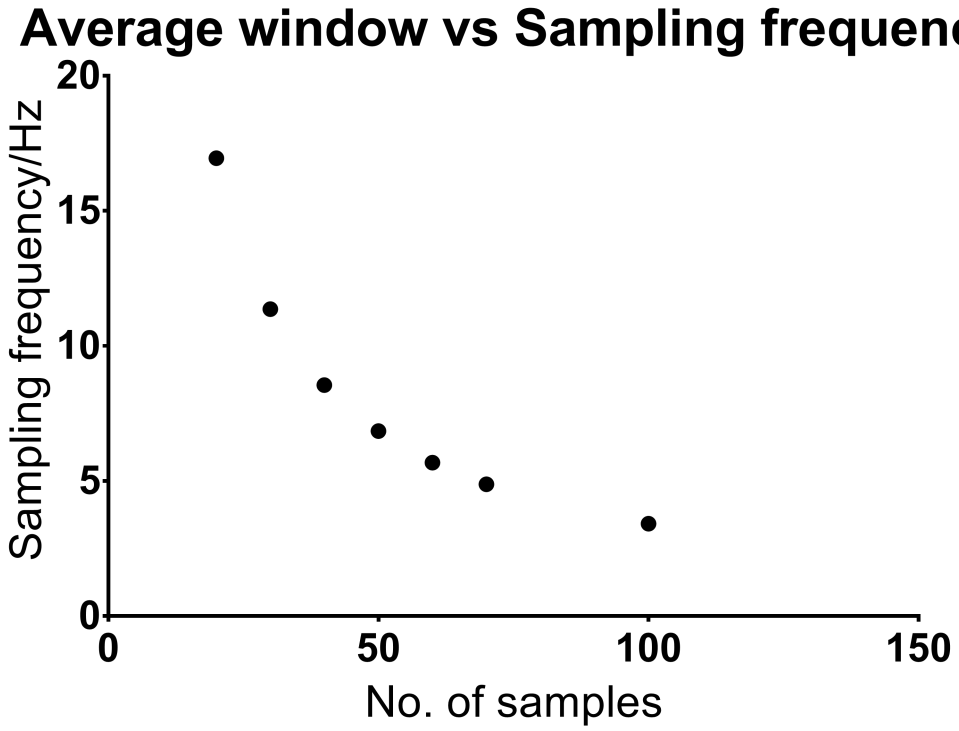


FIGURE 3.4: Window size vs sampling frequency

It seems that the sampling frequency is inversely proportional to the number of samples. This was confirmed by taking the log of both the frequency and the number of samples.

$$N \propto \frac{1}{f} \tag{3.3}$$

$$N = \frac{k}{f} \tag{3.4}$$

$$\log(N) = \log(\frac{k}{f}) \tag{3.5}$$

$$\log(N) = \log(k) - \log(f) \tag{3.6}$$

Plotting log(N) against log(f) gave the following result which proved the inversely proportional relation. The graphing software calculated the y-intercept to be 2.525. This is also

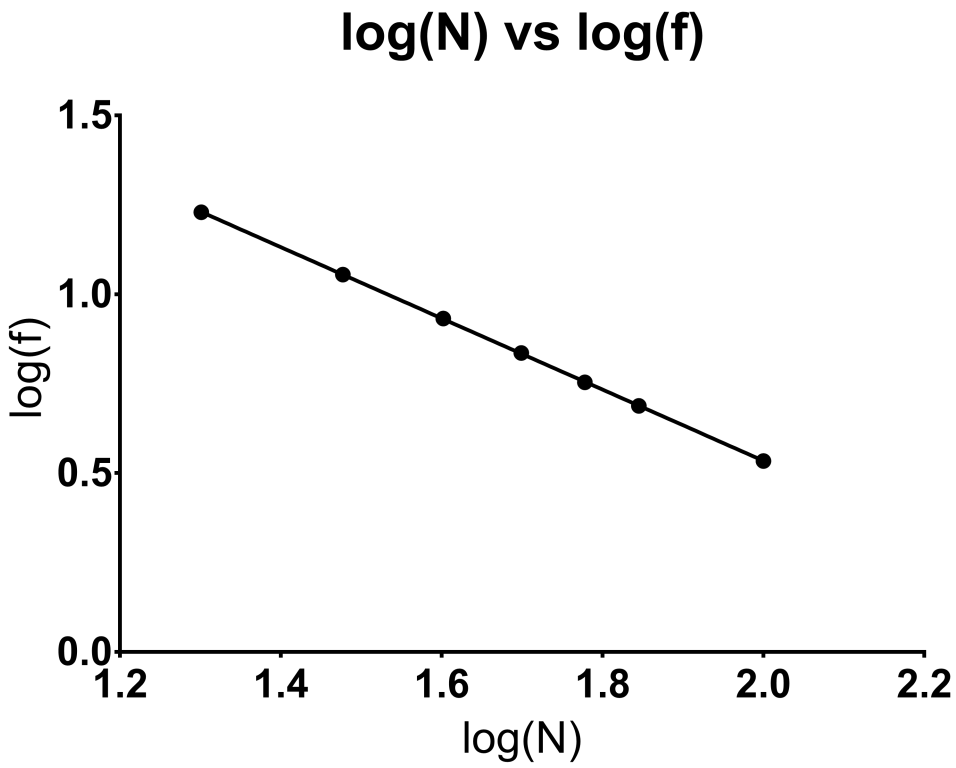


FIGURE 3.5: Window size vs sampling frequency

equal to log(k). Therefore k can be calculated to be $10^{2.525}$ which is 334.97.

Now the window size at f=4Hz can be calculated. To comply with the Nyquist-Shannon theorem, the maximum window size can be 84. There needs to be a large margin to keep the system responsive which is essential to suppress a fast oscillation. I chose a window size of 50 to make sure the system can keep up.

Chapter 4

Manufacturing and Assembly

This chapter will focus on the physical design of the wing that will be placed in the wind tunnel.

4.1 Main Wing

To calculate the lift produced by the wing, we will need local static pressure readings from several points along the top and the bottom of the wing. The easiest way to do this is by creating pressure tappings at specified points along the wing surface.

Firstly, an aerofoil needs to be chosen. I chose the NACA2412 aerofoil. This decision was not entirely arbitrary. The moderate thickness of 12% chord length and a camber of 2% at 40% chord length. This increases the L/D ratio and is good for low speed flight which is the regime most UAVs fly in.

The wing has a chord of 0.25m and a span of 0.35m. The trailing edge flap begins at x/c of 0.75. The pressure tappings are at x/c of 0.2, 0.4 and 0.6. There are probably better locations for these pressure tappings which will yield better lift estimates. However, finding the optimum locations is a large task and can be a separate project. For my project, the accuracy of the lift estimate is not highly important. The aim is to produce a controller that responds well to a change in airflow. Therefore, a lift estimator that can give values within 10% of the actual lift on average will be good enough to test the response of the control system.

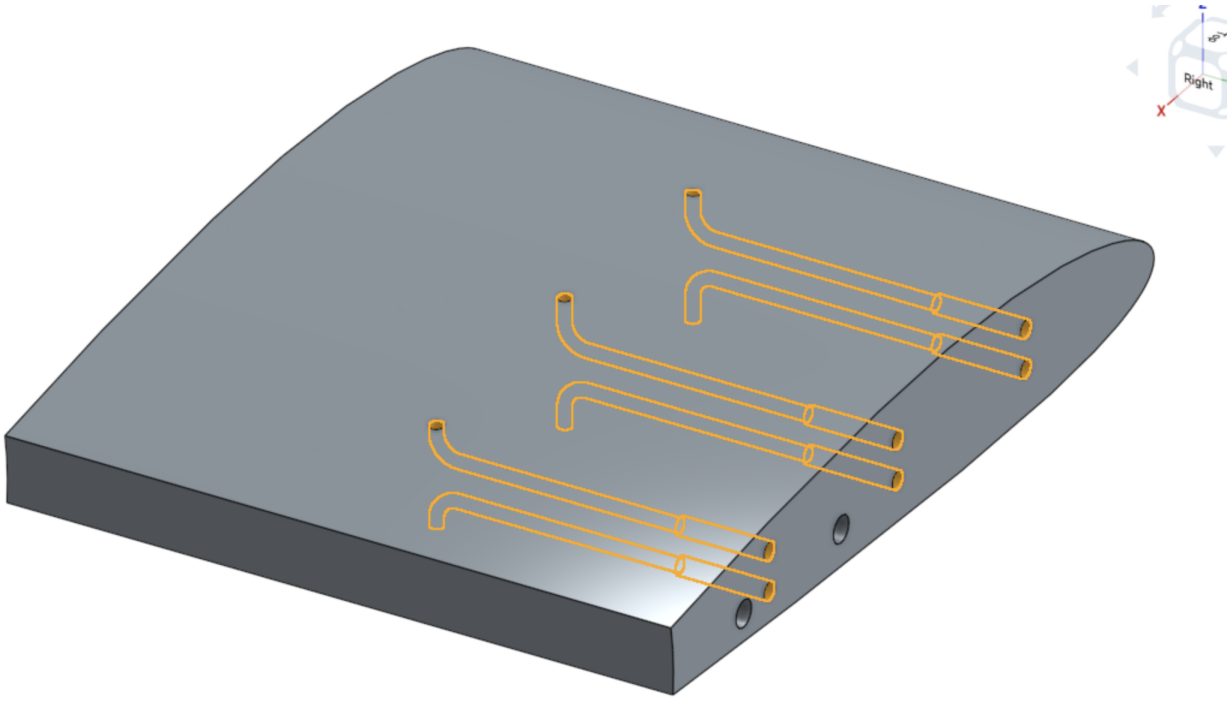


FIGURE 4.1: Midsection of the wing with the internal tubing highlighted

4.2 Gust Generator

To simulate an impulsive change in airspeed, a "gust generator" will be placed upstream from the wing. This consists of two NACA0015 airfoils with chord of 0.25 m and 0.5m span coupled to servos. The diagram below illustrates the configuration.

As the flaps move closer together, the airspeed of the flow between them will increase because of the decreasing cross-sectional area. This is simply due to conservation of mass.

$$\dot{m} = \rho V A \tag{4.1}$$

FIGURE 4.2: Gust generator in the wind tunnel

4.3 Assembly

The main wing was split into four parts: the two ends of the wings, the middle section and the flap. Different manufacturing processes had to be utilised to obtain these parts quickly and of sufficient quality.

The middle section is geometrically complex due to the presence of the pressure tappings. It is not very practical to hand craft the part and not feasible through 2D processes such as laser cutting or foam cutting due to the 3D geometric features. 3D printing is the only process that is suitable and available for this component. Unfortunately, the span of the wing is too great (30cm) for the 3D printer so to ends of the wing need to be extended. The 3D printed part was made 20cm wide.

To extend the wing by 5cm each side, the profile of the middle section was saved as a .dxf file. This profile was laser cut out of a 5mm thick plywood 20 times and glued together side-to-side.

The flap was made similarly. The profile was laser cut 60 times and glued together.

The main wing has been supported and mounted through two 6mm carbon fibre rods that go through the wing and attach to some fixtures on the walls of the tunnel. These fixtures were designed specifically for this project.

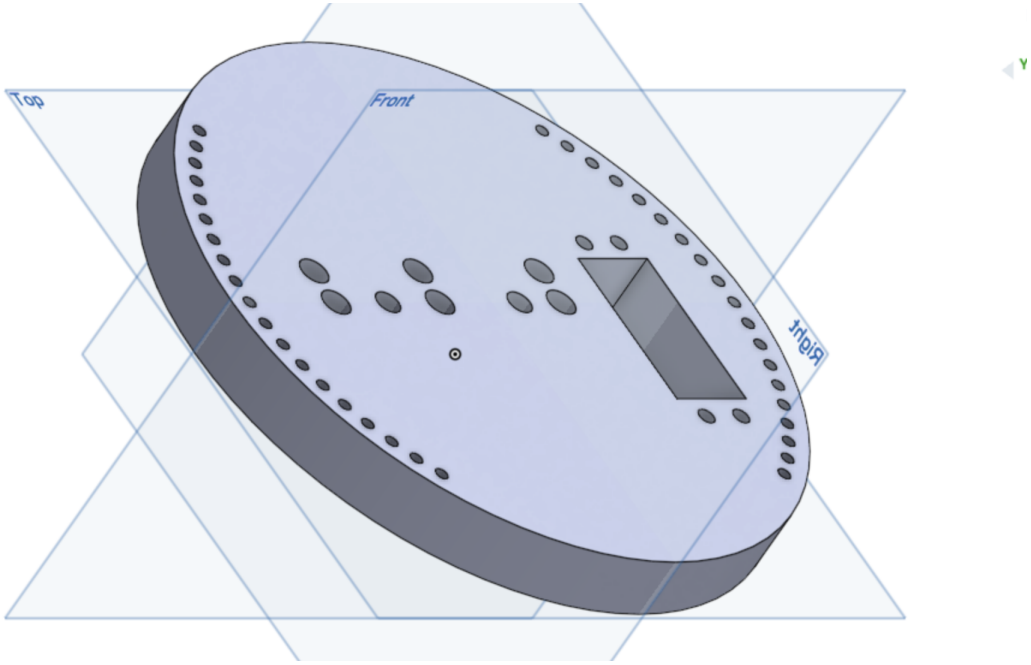


FIGURE 4.3: Fixture on the right hand side of the wind tunnel

Looking upstream, the fixture disc on the right hand side (Figure 4.3) has holes for the carbon fibre rods, a cut out for the flap servo to be mounted in, six holes for the PVC tubes connecting the pressure tappings to the sensors and several holes near the edges to allow changing the angle of attack of the wing to a desired setting. The angular separation between these holes is 5°.

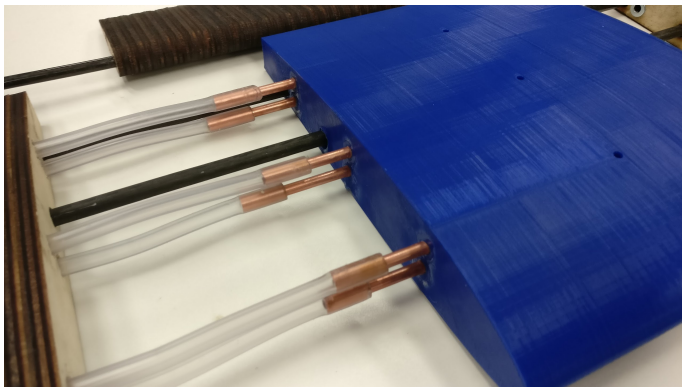


FIGURE 4.4: Picture of the PVC tubing and the midsection of the wing

The pressure tappings end at the right side of the middle section of the wing. Sections of a 4mm (outer diameter) copper pipe were placed in these holes and glued in place. These created "ports" which allowed easy and secure connection of the pressure tappings to the pressure sensors. Six equal sections of 3mm PVC tube(inner diameter) were cut to facilitate this connection. One end of each tube was secured to the pressure sensor ports, the other end had to be heated and expanded to fit onto the larger copper ports on the wing. This provided an excellent tight seal which is beneficial because less pressure will be lost.

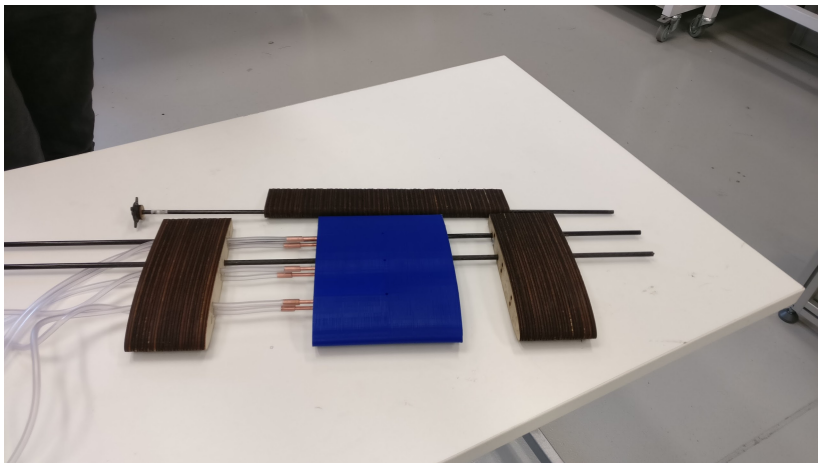


FIGURE 4.5: Picture showing how the different parts of the wing were integrated

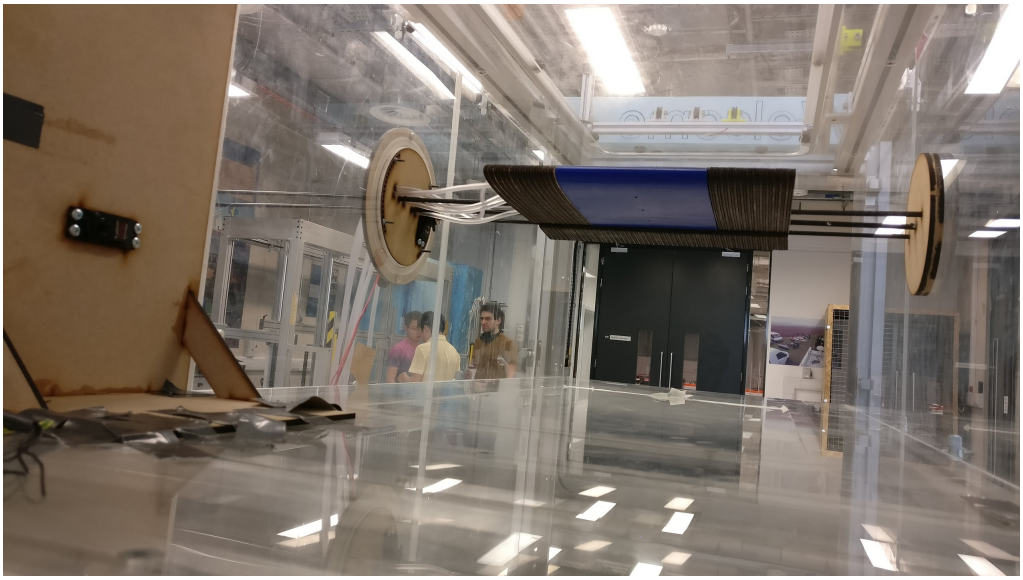


FIGURE 4.6: Picture showing the main wing assembly in the wind tunnel

4.4 Electronic setup

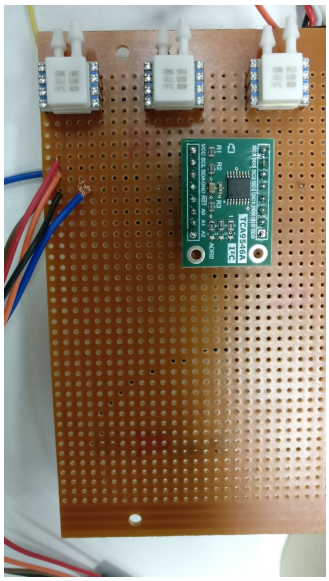


FIGURE 4.7: Photo of the breadboard with the multiplexer and the pressure sensors

As mentioned in section 3.1, the I2C pressure sensors all have the same address so are bound to create a conflict unless a switching mechanism is constructed. I used an I2C multiplexer called TCA9546a which is an I2C device itself.

It has four channels thus can handle four I2C devices. To select a channel, a number is sent to the I2C address 0x77 (where the multiplexer is located).

Channel 0	0x01
Channel 1	0x02
Channel 2	0x04
Channel 3	0x08

Once the channel is selected, I2C communication between the Raspberry Pi and the selected pressure sensor can begin. Figure 4.7 below illustrates the connection between the multiplexer and the pressure sensors.

Note that the servo is powered externally via a 6V DC wall adapter, but the ground of the external power source and the Raspberry Pi power source are tied together because the servo and the Raspberry Pi need a common ground.

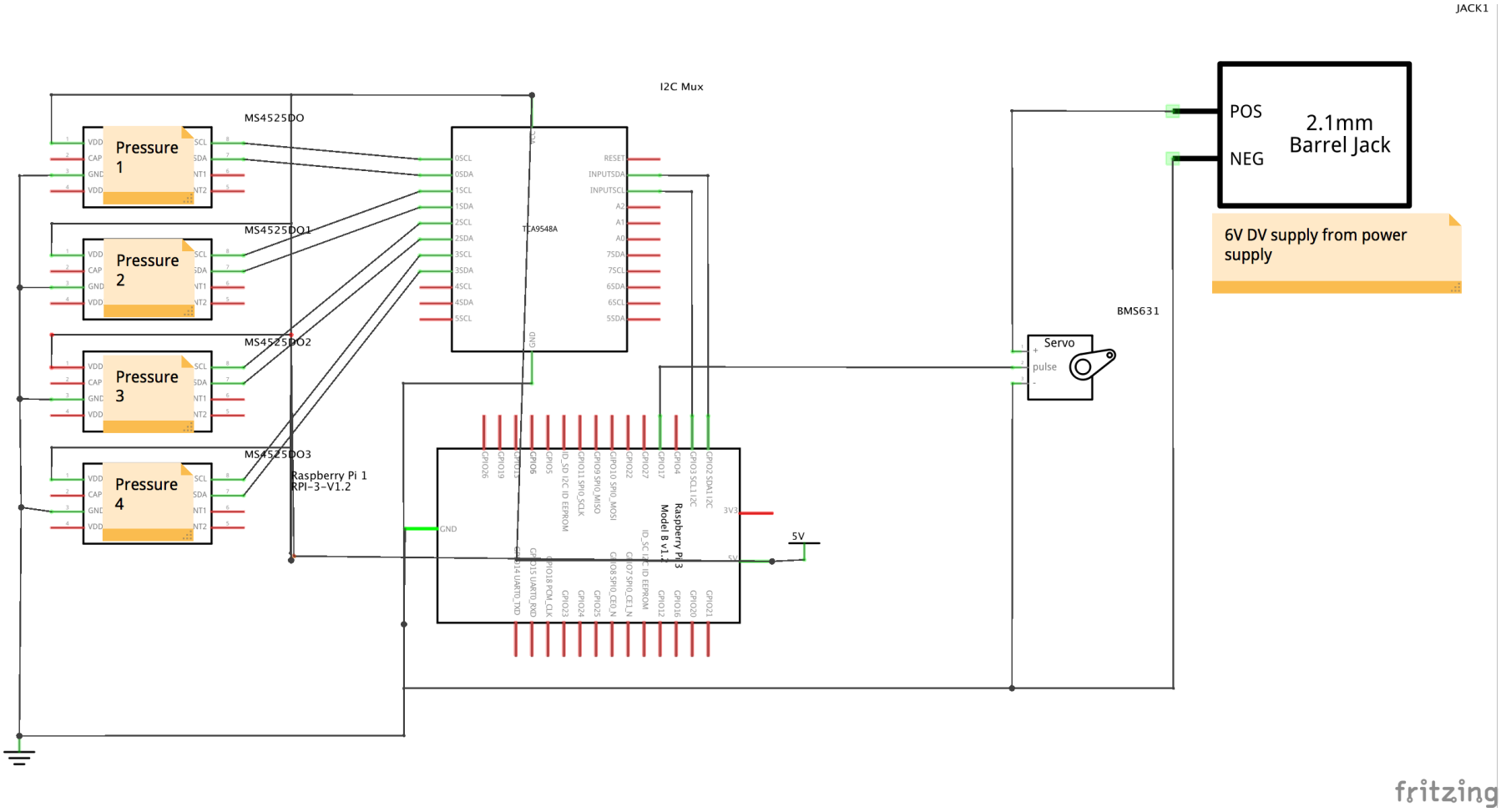


FIGURE 4.8: Fritzing diagram illustrating the electronic set up

Chapter 5

Plant dynamics and Simulink modelling

In this section, the dynamics of the plant will be estimated and analysed in MATLAB/Simulink to get initial estimates for the PID gains. There are two blocks in the plant for which we need a transfer function. We need to know how an input signal (PWM) translates to an angular displacement in the servo. Then, we need to understand how the angle changes the C_L and consequently the lift production.

5.1 Dynamics of the servo

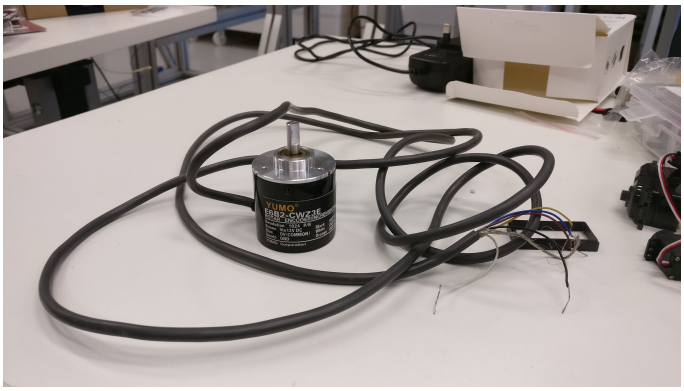


FIGURE 5.1: YUMO E6B2-CWZ3E

To measure the performance of the servo, I used a rotary encoder. This device converts the angular position into digital signals. The typical quadrature rotary encoders work using two signals: A and B. These are 90 degrees out-of-phase and operate on two logic levels. For clockwise rotation the A and B transition occurs in 4 phases:

TABLE 5.1: Phases of quadrature rotary encoders

	A	B
Phase 1	0	0
Phase 2	0	1
Phase 3	1	1
Phase 4	1	0

Signals produced during anticlockwise rotation occur in exactly the opposite sequence: signal A leads signal B.

Each phase change increments a counter. Each counter increment corresponds to a very small angular displacement. Thus, this device will be helpful to understand the dynamics of the servo.

The rotary encoder was coupled to the servo and the step response was monitored. The highest point was assumed to be 120 degrees because the manufacturer claims that 120 degrees is the throw of the servo. The following graph shows how the angle changes with time. The initial part of the graph is linear which implies that the servo has clearly reached the maximum speed it can achieve because the angular velocity is fairly constant. Right before the servo angle converges to the final value, the first order dynamics of the servo are visible. If we isolate that portion of the graph and analyse how it converges,

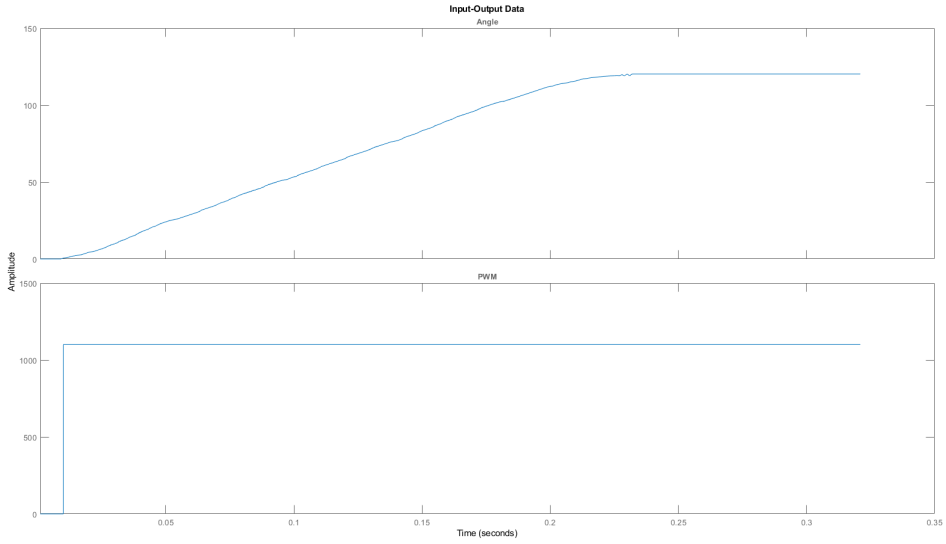


FIGURE 5.2: Step response of servo

we can find the time constant assuming that the servo follows first order dynamics (simple lag). The following expression is the transfer function for a first order system.

$$\frac{K}{Ts+1} \tag{5.1}$$

When subjected to a step input $1/s$, we get the following transfer function.

$$\theta(s) = \frac{1100K}{s(Ts+1)} \tag{5.2}$$

We can decompose this into partial fractions.

$$\theta(s) = \frac{1100K}{s} - \frac{1100KT}{Ts+1} \tag{5.3}$$

Then we can apply an inverse Laplace transform to find the transfer function in the time domain.

$$\mathcal{L}^{-1}\left(\frac{1100K}{s(Ts+1)}\right) = 1100K(1-e^{\frac{t}{T}}) \tag{5.4}$$

According to the data collected, the servo reaches to 94.799% within 0.195s. So the time constant can be found.

$$(1 - e^{-0.195/T}) = 0.94799 \tag{5.5}$$

$$T = 0.06596s$$

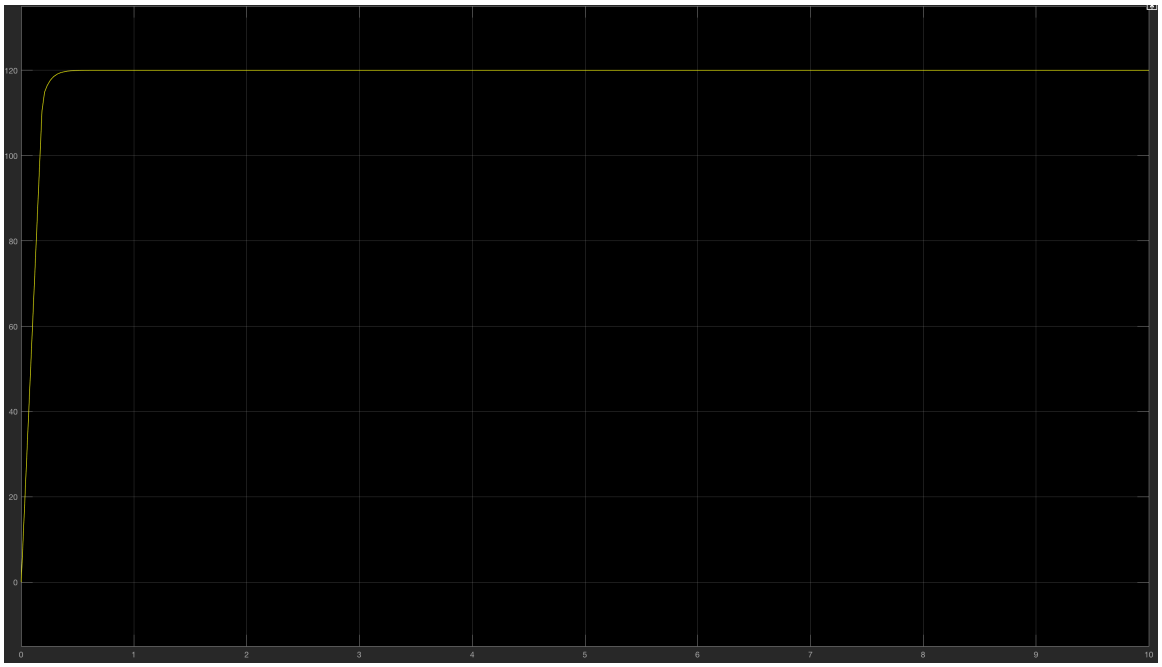


FIGURE 5.3: Servo approximation

The maximum speed can be found by taking the gradient at the straight line section of the graph.

$$\dot{\theta}_{MAX} = 0.599^{\circ} ms^{-1} \tag{5.6}$$

Finally, the gain is simply the ratio between the output and the input which is 120/1100.

$$K = 0.10909 \tag{5.7}$$

This model of the servo gives a very close approximation of the real servo. A Simulink model of the servo was made which incorporates the first order transfer function and a rate limiter set to 599. When a step input of 1100 is applied to the system, the output (figure 5.3) matches the graph of the measured data as shown in figure 5.2.

5.2 Dynamics of the flap

We now know how the angular displacement of the flap changes with time. We now need to know how that change in angle affects the coefficient of lift and consequently the lift.

To do this, I used Xfoil to see how the C_L changes with deflection of the flap.

The graphing software shows that a cubic function is a good fit for the data. However, I chose not to use a cubic function to approximate the flap dynamics because at the maximum/minimum point, it is not clear which direction the flap must deflect to increase/decrease the lift coefficient.

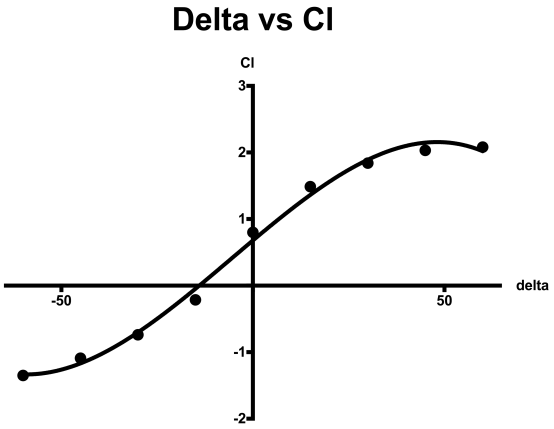


FIGURE 5.4: Flap deflection vs Coefficient of Lift

A better approximation is a linear function with saturation on both ends because it is monotonic. There is no ambiguity in which way the flap needs to deflect to increase/decrease C_L.

$$C_L(\delta) = 0.04257\delta + 0.6665 \tag{5.8}$$

$$-1.35 \leq C_L \leq 2.08 \tag{5.9}$$

5.3 Simulink first principles modelling

The last two sections identified the dynamics of the plant. It is now possible to import the models developed into Simulink and use the PID tuner app in MATLAB to tune controller for output disturbance rejection (impulse in output). The image below is a screenshot of

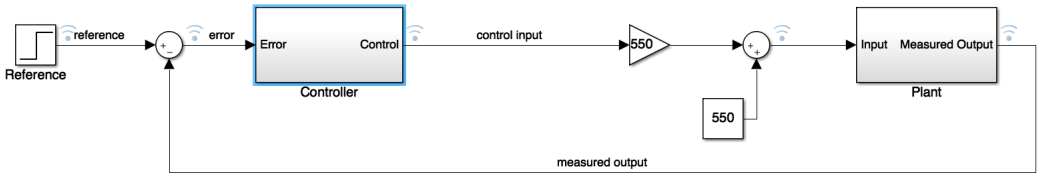


FIGURE 5.5: Simulink model

the Simulink model. The reference signal is a step signal that has a magnitude of 3.2 from 0s. The error signal is converted to a command signal or a 'control input' which ranges from -1 to 1. -1 corresponds to the flap being all the way up (-60 deg) which requires a PWM input of 0. On the other hand, for deflecting the flap all the way down (60 deg) a PWM value of 1100 is required.

Therefore, to convert the controller output to a PWM output, a gain of 550 is applied and a constant 550 is added.

The PWM signal is consequently fed into the plant. In figure 5.6, on the far left we have the controller generated PWM signal. This goes through a transfer function and a rate

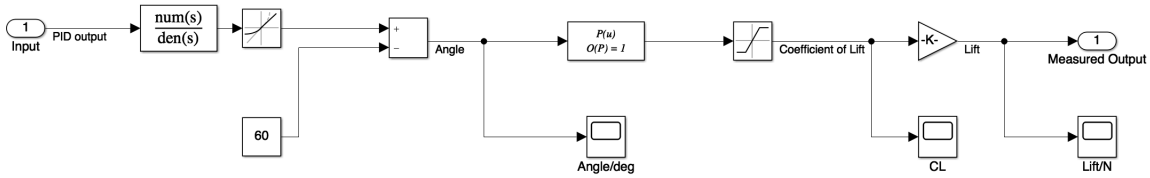


FIGURE 5.6: Simulink model of the plant

limiter as outlined in section 3.8.1. The output is the flap deflection angle. This signal then goes through a first order polynomial (refer to eq 5.8 and 5.9). The output is the coefficient of lift. This can be converted to lift by applying a gain of 6.615.

$$L = \frac{1}{2}\rho V^2 S C_L = \frac{1}{2} \cdot 1.225 \cdot 12^2 \cdot 0.30 \cdot 0.25 \cdot C_L = 6.615 C_L \tag{5.10}$$

This completes the model that I used for tuning my controller. To begin testing, I will need to find the optimal PID gains for my controller.

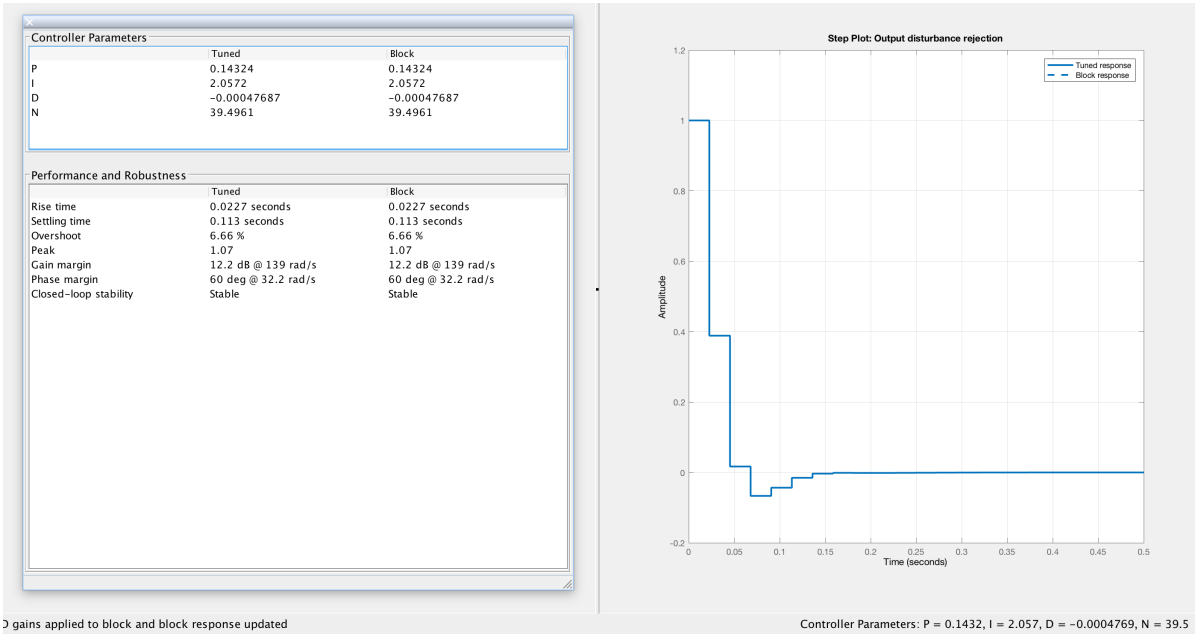


FIGURE 5.7: Tuner app with the performance parameters and the optimised gains in the bottom right

As mentioned earlier, I used the PID tuner app in MATLAB to determine these gains.

I started off by tracking the output disturbance rejection plot and then adjusting the response time until it was under 0.13s. The percentage overshoot was kept under 10%. The optimal gains found are shown in the table on the next page.

TABLE 5.2: Matlab predicted PID gains

K_p	0.14324
K_i	2.0572
K_d	-0.00047687

Chapter 6

Lift estimation and implementation

6.1 Lift Estimation

In section 3.3, I was able to accurately obtain pressure in pascals from the pressure sensors. This section will deal with using the pressure data from the sensors to calculate the lift produced by the wing. A good lift estimate is essential for feedback.

Lift and drag are both aerodynamic forces that are a result of an unequal pressure (thus force) distribution along the surface of the wing. If the force due to pressure normal to the direction of the free stream is resolved, the result would be the lift on the wing. Similarly, the component of pressure forces parallel to the free stream is resolved to find the drag.

$$\overrightarrow{F_N} = \oint p\hat{\mathbf{n}} dS \tag{6.1}$$

$$L = \overrightarrow{F_N} \cdot \hat{\mathbf{n}} \cos\alpha \tag{6.2}$$

$$D = \overrightarrow{F_N} \cdot \hat{\mathbf{n}} \sin\alpha \tag{6.3}$$

These equations cannot be directly applied to measured data because we have pressure readings over finite discrete points along the wing surface. But if the wing has a large number of pressure tappings, the lift can be estimated by just summing the product of the individual pressure readings and the distance between consecutive tappings.

$$L = \sum_{i=1}^{n} (p_{li} - p_{ui}) \delta A \tag{6.4}$$

Linear interpolation between consecutive points can further improve the estimation. The size of the wing in this project is unfortunately constrained due to the dimensions of the wind tunnel. Fitting a large number of pressure tappings within the wing was therefore not possible. A large number of pressure tappings also require a large number of pressure sensors which is not economically viable. So the lift estimation is expected to be a very rough approximation but the error should be under 10%. I will use an empirical approach to design the lift estimator.

We can assume that the lift is estimated by the following equation:

$$L = \sum_{i=1}^{3} w_i (p_{li} - p_{ui}) \tag{6.5}$$

w_1,w_2 andw_3 are weights given to the differential pressure readings from each location on the wing. These weights will be determined using Xfoil and Excel. I will use Xfoil to sweep through α between -5 and 10 and δ between -60° and 60°. The coefficient of pressure at C_p

at the top and bottom of the wing at x/c values of the pressure tappings will be recorded. These can be converted to pressure.

$$p = p_\infty + \frac{1}{2}\rho V_\infty^2 C_p \tag{6.6}$$

The pressure values should roughly be what the pressure sensors will actually measure. The C_L is also recorded for each test. This can be used to find the actual lift on the wing.

The results were all entered into an excel spreadsheet. A column for the estimated lift took the xfoil predicted pressure at each tapping, multiplied it by the corresponding weight and added them. The square of the error was calculated by taking the difference between the actual lift and the estimated lift and squaring the resulting value.

Excel has a solver function which is often used for optimisation. The sum of the squared error values for all tests (across all angles of attacks and flap deflections) was minimised by allowing the solver to adjust the weights. The following values were determined by excel solver. The average percentage error was about 7.3%.

w_1	0.021042611
w_2	-0.022570784
w_3	0.025133973

Considering a wide range of flap deflection angles and angle-of-attack when estimating the weightings gave me a lift estimator that works fairly well for different operating conditions the system is likely to see during testing. The full Excel spreadsheet can be found in the appendix.

6.2 The code

Before testing can begin, the code has to be written that will collect the data from the sensors and use it to calculate the lift. It should then control the flap angle in accordance to the controller that I designed in the preceding sections.

It will need to follow a certain sequence. Figure 6.1 shows the sequence of tasks that need to be carried out when the code is compiled and executed.

The 'main' void in my code ran an iterative loop that called different voids, each with a dedicated purpose. The main void was also used to set up the I2C devices and the servo. Both the I2C devices and the servo were implemented using the wiringPi and wiringPiI2C libraries both of which are made publicly available by Gordon[15] under GNU LGPLv3 license.

```
1  softServoSetup
   (0, 1, 2, 3, 4, 5, 6, 7);
2  int fd = wiringPiI2CSetup (0x28);
     // 0x28 I2C device address
3  int fd2 = wiringPiI2CSetup (0x77);
     // 0x77 I2C device address
```

The I2C devices at address 0x28 are the sensors and the device at 0x77 is the multiplexer (refer to section 3.5).

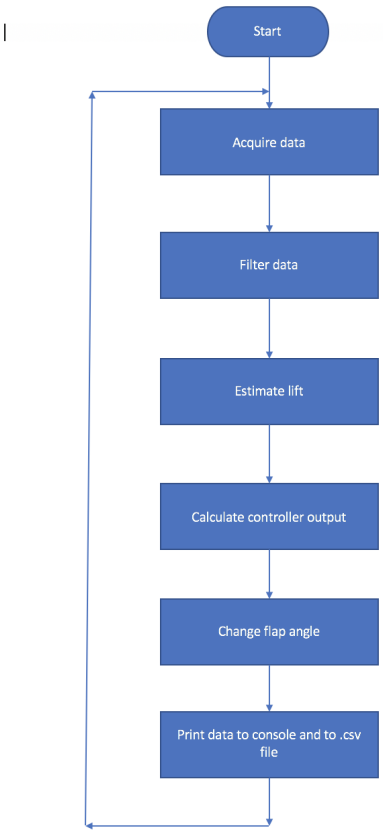


FIGURE 6.1: Flowchart showing the sequence of code blocks

The iterative loop mentioned above was done via an endless while loop as given below. The clockgettime function was called at the beginning and the end of the loop to calculate the loop/sample time which is useful for the PID calculation and also to understand whether the program was running fast enough to keep up with the oscillation.

```
1  while(1){
2    clock_gettime(CLOCK_REALTIME, &gettime_now);
3    start_time = gettime_now.tv_nsec;
4    counter+=1;
5    daq(fd, fd2);
6    estimateLift();
7    control();
8    printdata();
9    clock_gettime(CLOCK_REALTIME, &gettime_now);
10   if(gettime_now.tv_nsec - start_time>0){
```

```
      dt = (gettime_now.tv_nsec - start_time)/1000000000;
    }
  }
```

The 'daq' void being called above was used to acquire data from each sensor and to filter the data using an average filter with a window of 20 samples.

```
  while(i<20){
    wiringPiI2CWrite (fd2, ch_2) ; // select channel
    read(fd, &data2, 4); //request 4 byte data from device
    pres2 = (((int)(data2[0] & 0x3f)) << 8) | data2[1]; // acquiring pressure
    data
    p_out2 = 1.098*pres2-8914 ; //conversion and calibration to give pressure in
     Pa
    ap1+=p_out2;
    i+=1;
  }
  ap1=ap1/40;
  i=0;
```

The code above is an example which shows how data was collected and averaged from sensor 2 after converting to pascals. The 'ch2' variable is equal to 0x02. When this value is sent to the multiplexer, the second channel is selected.
Once data from all sensors is acquired and filtered, the lift is estimated in the 'estimateLift' void.

```
void estimateLift(){ //to estimate lift using acquired pressure data
  L=(w1*ap1+w2*ap2+w3*ap3);
}
```

This is fairly straightforward. The block of code above simply applies the equation 6.5. The weightings are global variables declared at the start of the file.
Once the lift is known, the error value and the controller output can be calculated.

```
void control(){ //run PID iteration and actuate
  e=L-3.2;
  P = Kp*e;
  if(output<-1){
  }
  else if(output>1){
  }
  else{
    int_e += e;
  }
  I = Ki*int_e*dt;
  D = Kd*(e-el)/dt;
  output = (P+I+D);
  if(output<-1){
    output=-1;
  }
  else if(output>1){
    output=1;
  }
  el=e;
  angle = -550*output+550;
  }
```

The void above is called after the lift is estimated is estimateLift void. It implements a PID controller with anti-windup protection. This prevents the integral error from accumulating a large value during initialisation or scenarios where the error is temporarily so high

that the output would saturate and take a long time to return to nominal values even after the cause of the error is gone.

Lastly, the data needs to be timestamped and printed to the console for live monitoring and to a .csv file to analyse the data after testing. All of this is done in the 'printdata' void. The following code is responsible for displaying and storing the data.

```
1  clock_gettime(CLOCK_REALTIME, &gettime_now);
2  printf("%f ; %f ; %d\n", gettime_now.tv_sec+((double)gettime_now.tv_nsec)
     /1000000000 - init_time, L, angle);
3  clock_gettime(CLOCK_REALTIME, &gettime_now);
4  fprintf(fp,"%f ;%d ; %f ; %f ; %f ; %f\n",gettime_now.tv_sec+((double)
     gettime_now.tv_nsec)/1000000000 - init_time, angle, ap1 , ap2 , ap3, L);
```

The wall time since program start, the lift and the angle of flap are printed to the console. All of the aforementioned data as well as the filtered pressure values are written to the .csv file.
The code can be found in its entirety in the appendix.

Chapter 7

Wind tunnel testing

All the tools, equipment and resources required for the project have been gathered and assembled as discussed in the preceding chapters. This chapter will outline how I carried out the testing in chronological order. As with most things, the first iteration of any design is likely to be flawed. But this chapter will show how I was able to improve and optimise the system in each successive iteration.

7.1 Measuring wind tunnel airspeed

The wind tunnel is operated using a control panel which displayed a frequency. This frequency is proportional to the airspeed in the wind tunnel. However, the exact relation between this frequency and the airspeed was unknown. Thus, the speed of the wind tunnel had to be measured at different frequencies to make sure that I was keeping the test conditions the same throughout testing.

Using a pitot probe and one of the differential pressure sensors, I was able to establish a clear linear relation between the frequency and the airspeed. The following figure shows this relation.

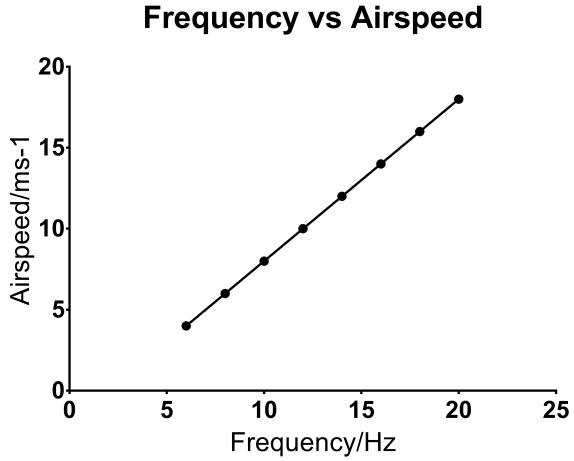


FIGURE 7.1: Simulink model

To keep the test conditions constant, I will run all tests (except the airspeed sweep test) at a constant speed of $12ms^{-1}$ which corresponds to 14Hz. I will also keep the angle of attack at 5° which according to the lift estimator should give me a lift of 4.6 N. I will keep the setpoint at 3.2N.

7.2 First test

I began by testing the control system in steady state conditions. The wind tunnel was set to $12ms^{-1}$. The controller should be able to hold the lift at roughly 3.2N.

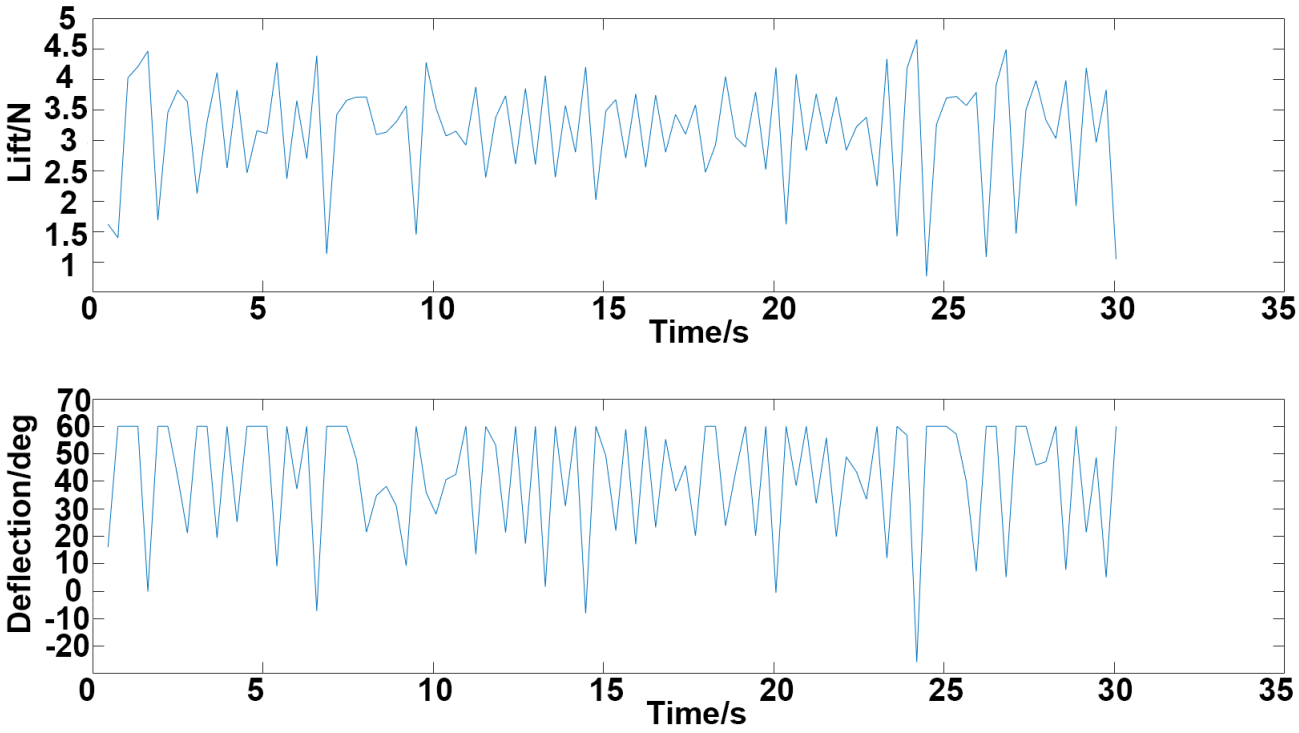


FIGURE 7.2: The controller was unable to converge the lift to the setpoint.

The above result was expected. The controller was unable to stabilise the system and oscillated violently. As you can see from figure 7.2, the flap angle repeatedly saturated at 60° (flap down position). This was expected because the controller was tuned very aggressively. There is also a considerable amount of noise in the data to which the controller was hypersensitive. One solution to this issue would be to increase the averaging window size to reduce noise. However, this would mean that the sample rate decreases below the Nyquist frequency. It may also mask the gust to the point where the impulse will go undetected.

In order to fix the controller, I took a different approach. By considering the system response to a specific input, we can approximate the actual plant and find appropriate gains accordingly. A heuristic method called Cohen and Coon[10] tuning helped me to improve the system.

7.3 Cohen and Coon tuning

This method involves using the plant in open loop. A step input is given to the system and the output is monitored. A step input from -60 degrees to 60 degrees was applied to the system and the figure below shows the response. We begin by identifying when the step

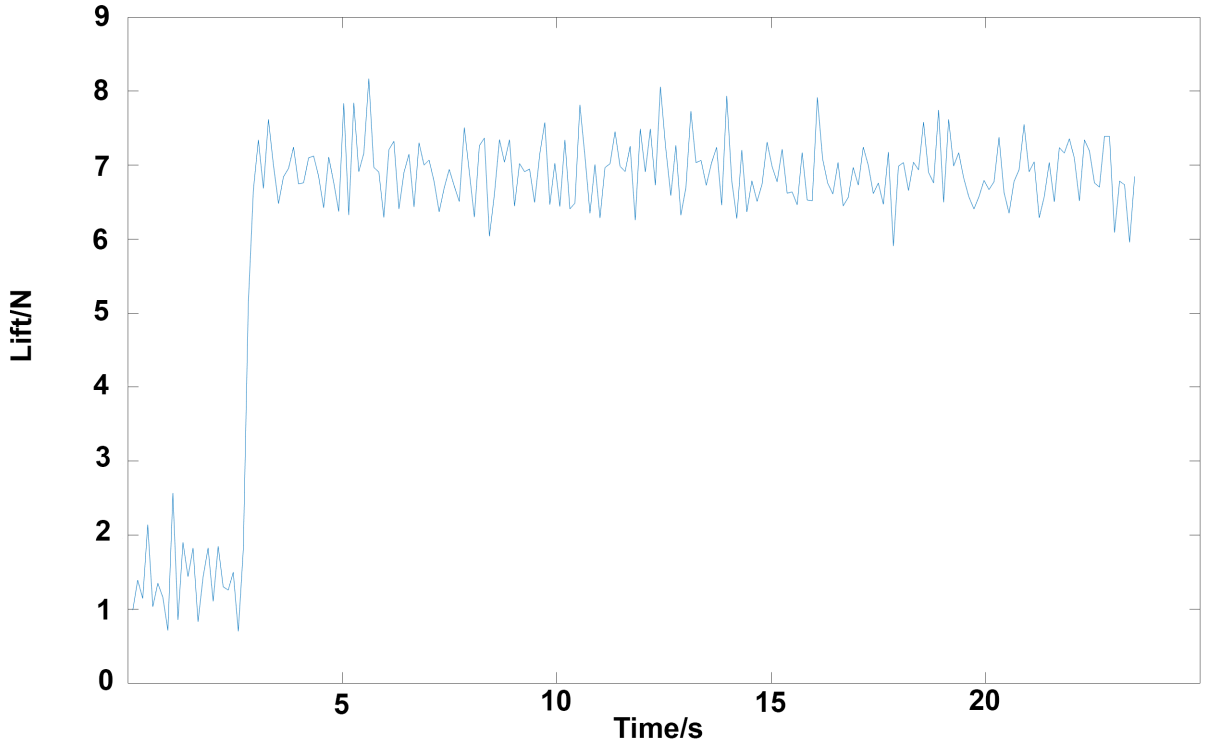


FIGURE 7.3: Step response

input was initiated(t_0). From the data, is found that $t_0 = 2.577s$.

The half-way point t_2(when the lift is halfway between the initial value and the final value) and the time when 63.2% of the value between the initial and final lift occurs (t_3) is also found. $t_2 = 2.771s$ and $t_3 = 2.799s$. The end of deadtime can be calculated (t_1). This is the time when the process variable (lift) starts rising.

$$t_1 = \frac{t_2 - ln(2)t_3}{1 - ln(2)} = 2.708s \tag{7.1}$$

τ is the time between the end of deadtime and the 63.2% point (t_3).

$$\tau = t_3 - t_1 - 0.091s \tag{7.2}$$

τ_{del} is the duration between the step input initiation and the end of deadtime.

$$\tau_{del} = t_1 - t_0 = 0.13127s \tag{7.3}$$

K is the ratio between the final and the initial value which is roughly 6/2=3. The last value needed to calculate the gains is r:

$$r = \frac{\tau_{del}}{\tau} \tag{7.4}$$

When all of the above parameters have been calculated, the gains can be calculated.

$$K_p = \frac{1}{Kr}\left(\frac{4}{3} + \frac{r}{4}\right) = 0.391 \tag{7.5}$$

$$t_i = \tau_{del}\frac{32 + 6r}{13 + 8r} \tag{7.6}$$

$$K_i = \frac{K_p}{t_i} = 1.7990 \tag{7.7}$$

$$t_d = \tau_{del}\frac{4}{11 + 2r} \tag{7.8}$$

$$K_d = K_p t_d = 0.0148 \tag{7.9}$$

Using these values yielded a much improved result albeit not perfect.

7.4 Airspeed Sweep test

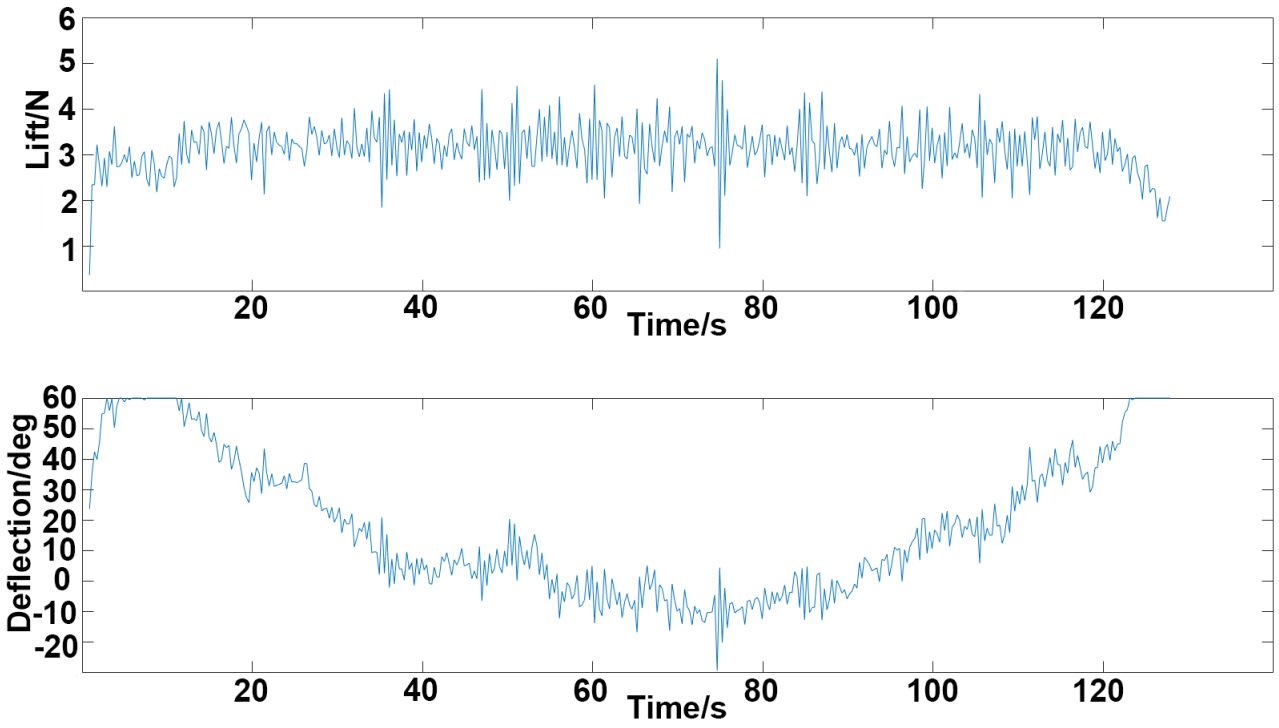


FIGURE 7.4: Self-induced oscillations were noticed with the Cohen and Coon tuned controller

This test was was able to verify that the control system attempts to correct the lift when there is a change in airflow and is able to converge to a value when the conditions are held steady. By changing the airspeed in the wind tunnel, we should be able to observe that the controller changes the flap setting in order to maintain the lift. Eventually, a steady state flap angle should be observable. During high speeds, the flap should be at a relatively high angle to avoid excessive lift. Similarly at low speeds, the lift will be kept at the setpoint by

a relatively low flap deflection angle. This test is different to gust tests because the airspeed changes more gradually than in a gust (which is characterised by an impulse in airspeed).

Figure 7.4 shows the result from the airspeed sweep test. It shows that the controller is certainly responding to changing airspeeds and on averages holds the lift at 3.2N. Discrete sections are observable where the flap angle is roughly constant such as between 40s and 50s. During these times, the speed in the wind tunnel has stabilised at the setpoint. However, it is also inducing a few oscillations by itself. A major oscillation is visible at around 75s. This was again expected because the Cohen and Coon tuning method is used for fast response. This means that the controller is still a little too aggressive.

I had to reduce the proportional gain to 0.02 to remove all self-induced oscillations. I did not change the integral and derivative gains.

The airspeed sweep test was repeated with the more robust controller and the results can be seen in figure 7.5 below. We can see that the controller is far less jittery than it was before in figure 7.4. It also clearly shows the controller's ability to converge to a certain value. The lift is held steady at 14 meters per second airspeed when the flap is at roughly -20 degrees (seen between 50s and 70s). As the wind tunnel airspeed ramps up, we can see the controller manages to keep the lift fairly constant. The only exception is when the system is saturated. This is seen when the wind tunnel is off or ramping up to $14ms^{-1}$ (between 0s and 40s). The flap is fully down while the speed is not high enough to produce the setpoint lift. The opposite situation is also observed. At $18ms^{-1}$ (between 100s and 120s), the speed is too high. Even when the flap is saturated at $-60°$, the lift cannot be reduced to the setpoint.

The test shows that the control system works very well in steady conditions as long as the actuator is not saturated.

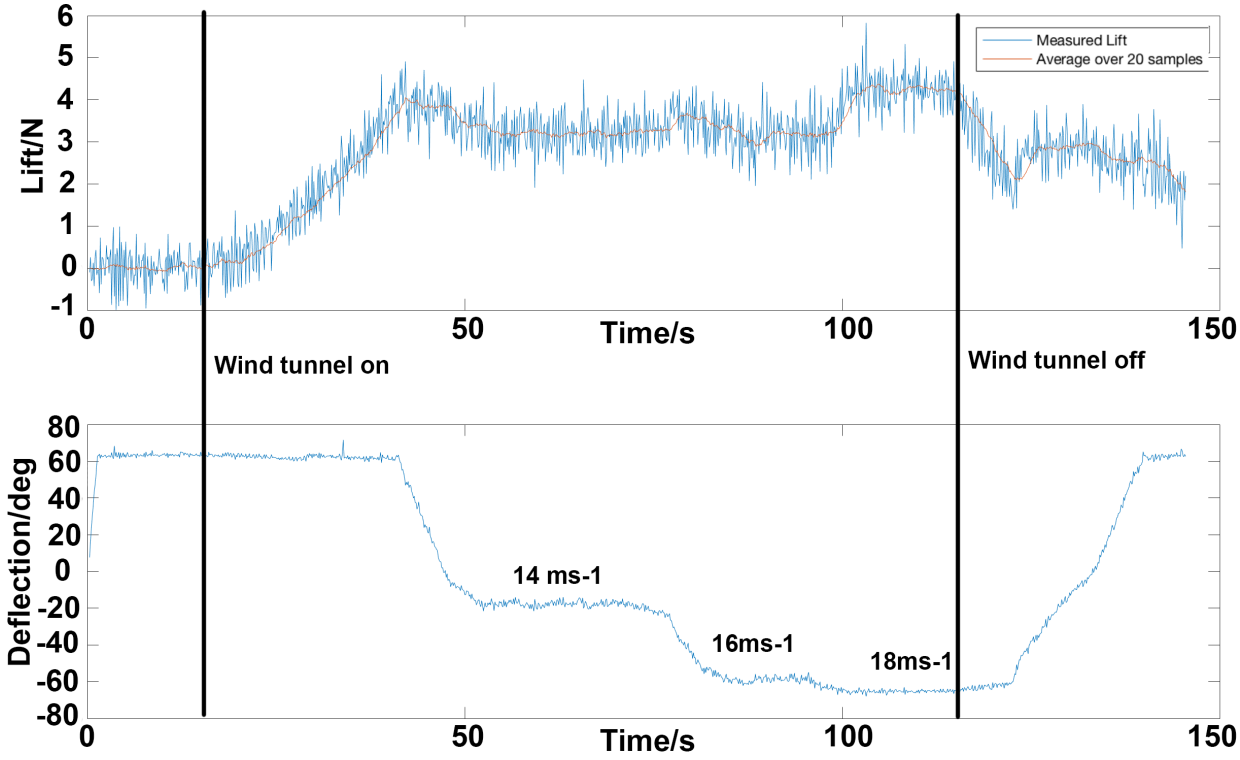


FIGURE 7.5: Airspeed sweep test. Note the steady states achieved by controller at different airspeeds.

7.5 Gust Response

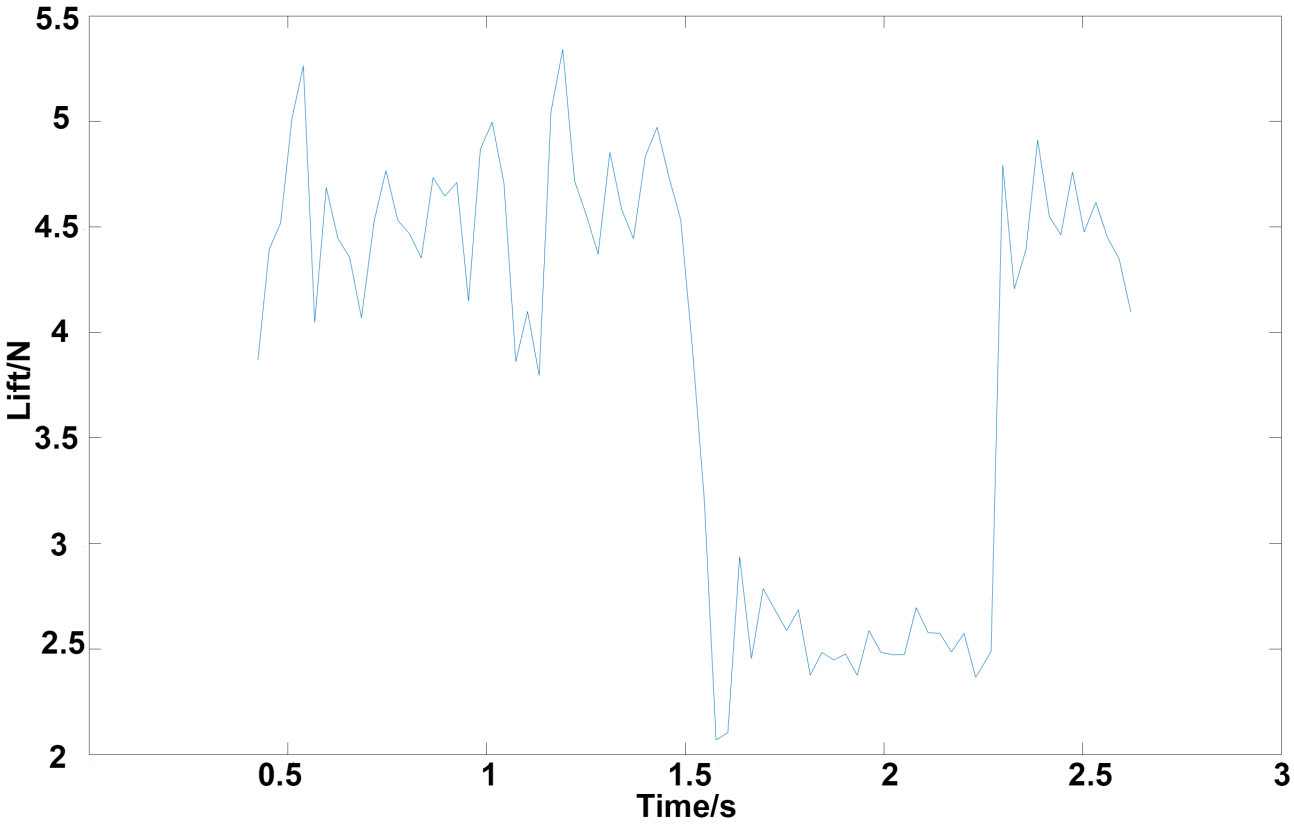


FIGURE 7.6: Impulse in lift as a result of the gust

The airspeed sweep test confirmed that the control system works in normal conditions. In this test, the controller will be tested through an impulse. The gust generator will be used to produce a large and a short lived drop in airspeed and the lift. To achieve this, the gust flaps were held steady at 10 degrees angle of attack towards each other. Then the flaps were suddenly deflected away from each other at 30 degrees. This caused a drop in airspeed and the lift.

To compare how well the system performs, I measured the lift changes with no controller present. The controller was disabled and the lift drop through the gust was observed.

Figure 7.6 above shows the lift on the wing changes through a gust when the controller is disabled and the trailing edge flap is set to 0 degrees. As a side note, this test also shows that the lift estimator corroborates with Xfoil predictions (see Appendix C).

The controller was enabled and the same test was repeated. Figure 7.7 on the next page shows the graph of the test.

The gust was initiated at roughly 27.5s and is clearly visible as a great drop in lift. This is counteracted by the controller's attempt to increase lift by increasing the flap deflection angle by over 30 degrees. The lift production returns very quickly to the original value

which confirms that the control system works well. The lift crosses 3.2N roughly 0.2s after the gust but it is difficult to identify the settling time because of the noisy data. However, the relatively quick recovery of the loss of lift makes this project a resounding success.

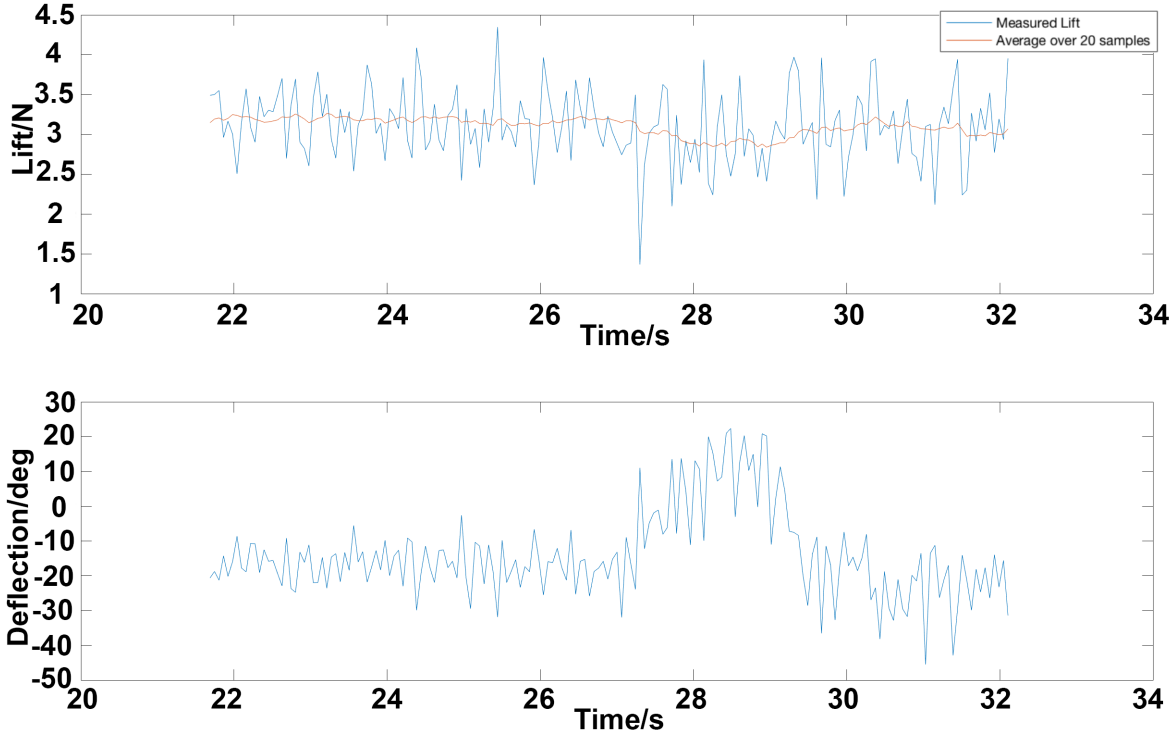


FIGURE 7.7: Lift and flap angle vs time. Note the big drop in lift and the consequent increase in flap angle in an attempt to recover the lost lift.

7.6 Discussion of results

As shown by the airspeed sweep test, the system is able to maintain a constant 3.2N lift (as long as the airspeed is between $8ms^{-1}$ and $16ms^{-1}$ to avoid saturation). The controller after manual tuning is very steady at constant airspeeds and causes only small overshoots when the airspeed gradually changes.

The controller also performs very well in gusts. Compared to the 'no control' test, the size of the overshoot is reduced slightly, but more importantly the time to return the lift to the setpoint is advanced by almost 0.8s. This is better than what studies thus far have achieved. However, the system is not as fast as I had aimed for (settling time of 0.13s). In the next chapter, I shall discuss the reasons why this was not possible.

Chapter 8

Summary

This project was overall a success. In this chapter, I will be addressing how I met most of the aims and objectives set out in 1.1.3.
First and foremost, the aim of the project was to design a control system than can suppress the rapid change in lift production seen by the wing through a gust. Comparing the how the lift changed when the flap wasn't controlled to when it was controlled confirms that this objective was fulfilled. The amplitude of the oscillation as well as the time within which the lift was returned to the nominal values was reduced.

Secondly, I set a qualitative target. The settling time should be about 0.13s. Although the exact settling time cannot be identified from the noisy data, the system crosses 3.2N for the first time 0.2s after the gust. Therefore, the settling time is definitely more than 0.13s. The control system is still more robust than what has been achieved by similar systems thus far.

I had hoped that the amplitude of the oscillation on the wing would be lower with the controller present than without. With no control, the lift dropped from 4.5N to 2.5N which gives a ΔL of 2N.

With control, the lift dropped from 3.2N to 1.5N. This gives a ΔL of 1.7N. In this regard, the controller did slightly better.

It can also be assumed that in a positive gust, the controller will behave similarly because lift change is roughly linear with flap deflection (see section 3.8.2).

Lastly, I aimed to design the control system from first principles and then validate it during testing. It is quite clear that my model in chapter 3 is flawed because the estimated PID gains and manually tuned PID gains are vastly different. But the reasons why the Simulink model did not work are also quite clear.

8.1 Further recommendations

After completing this project, it is clear to me that several things can be done in the future to obtain a faster and more robust system.

Firstly, it's worth understanding why the self-induced oscillations occurred after Cohen and Coon tuning. The system is certainly stable because the oscillations do not increase and the controller always manages to remove them. Ideally, after a steady state is achieved there would be no reason for the controller action to oscillate. This phenomenon is most likely excited by the noise in the data. Reducing this noise can be effectively done by simply increasing the average window size. But this was not desirable because increasing the window size increases the loop time. According to Nyquist's theorem, the frequency of the loop should be at least twice than the oscillation's frequency. But by experimentation it was seen that in practice the loop time has to many times faster than this so that the system actually responds to disturbance. I had to reduce the window size to 20 samples to make the system responsive enough.

A severe bottleneck was caused by the relatively slow I2C bus. In the future, it will certainly be beneficial to invest in faster sensors such as the ones based on the SPI bus. This will allow averaging over more samples reducing the noise.

Alternatively, advanced filters such as a Kalman filter should be strongly considered. A problem with the Kalman filter is that it is difficult to determine the tuning parameters. [13] Some fairly advanced systems have been developed to calculate these parameters such as "reinforcement learning"[14] which involved some trial and error and dynamic programming in order to optimise the filer.

The Simulink model that I made had a few flaws that may have given me an inaccurate representation of the system.

Adding white-noise to the Simulink model would ensure that the sensor noise is accounted for when calculating the PID gains.

It was noticed that the servo throw was visibly reduced by 20-30 degrees when the flap was subjected to an aerodynamic torque in the wind tunnel. Clearly, the servo dynamics are different when a resistive torque is applied to it. This was not considered when approximating the servo dynamics in section 5.1.

Lastly, there were some minor concerns with the manufacturing quality of the laser cut parts. The surface of these parts were quite rough and could potentially cause turbulent flow. However these parts were either downstream of the pressure tappings or far away from them so the influence the surface roughness would have on the pressure readings is unlikely to amount to much.

References

[1] Catterall, R. (2003). State-Space Modeling of the Rigid-Body Dynamics of a Navion Airplane From Flight Data, Using Frequency-Domain Identification Techniques. [online] Trace.tennessee.edu. Available at: http://trace.tennessee.edu/cgi/viewcontent.cgi?article=3269&context=utk_gradthes [Accessed 2 Feb. 2018].

[2] B. Etkin and L. Reid, Dynamics of flight. Chichester: Wiley, 1996, pp. 15, 174,175.

[3] H. Chun and C. Chang, "Longitudinal stability and dynamic motions of a small passenger WIG craft", Ocean Engineering, vol. 29, no. 10, p. 1161, 2002.

[4] C. Huang, Q. Shao, P. Jin, Z. Zhu and B. Zhang, "Pitch Attitude Controller Design and Simulation for a Small Unmanned Aerial Vehicle," 2009 International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, Zhejiang, 2009, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5336043&isnumber=5335881

[5] "14 CFR 23.181 - Dynamic stability.", LII / Legal Information Institute, 2011. [Online]. Available: https://www.law.cornell.edu/cfr/text/14/23.181. [Accessed: 03- Feb- 2018].

[6] B. Aliyu, C. Osheku, P. Okeke, F. Opara and B. Okere, "Oscillation Analysis for Longitudinal Dynamics of a Fixed-Wing UAV Using PID Control Design", Advances in Research, vol. 5, no. 3, p. 6, 2015.

[7] N. Li and M. Balas, "Aeroelastic Control of Wind Turbine Blade Using Trailing-Edge Flap", Wind Engineering, vol. 38, no. 5, p. 1, 2014.

[8] J. Lee, J. Han, H. Shin and H. Bang, "Active load control of wind turbine blade section with trailing edge flap: Wind tunnel testing", Journal of Intelligent Material Systems and Structures, vol. 25, no. 18, pp. 2253-2255, 2014.

[9] Python 3 vs C gcc (64-bit Ubuntu quad core) | Computer Language Benchmarks Game. [online] Available at: https://benchmarksgame.alioth.debian.org/u64q/compare.php?lang=python3&lang2=gcc [Accessed 13 Apr. 2018].

[10] "Cohen Coon Tuning Method", Pages.mtu.edu, 2018. [Online]. Available: http://pages.mtu.edu/~tbco/cm416/cctune.html. [Accessed: 06- Apr- 2018].

[11] Anderson, Fundamentals of aerodynamics. New York: McGraw-Hill, 2011, p. 55.

[12] C. Shannon, "Communication in the presence of noise", Proceedings of the IEEE, vol. 72, no. 9, p. 11, 1984.

[13] Saha, M., Goswami, B. and Ghosh, R. (2013). Two novel metrics for determining the tuning parameters of the Kalman Filter. p.1.

[14] Goodall, C. and El-Sheimy, N. (n.d.). INTELLIGENT TUNING OF A KALMAN FILTER USING LOW-COST MEMS INERTIAL SENSORS. p.1.

[15] "Raspberry Pi | Wiring | Gordons Projects", Projects.drogon.net, 2018. [Online]. Available: https://projects.drogon.net/raspberry-pi/wiringpi/. [Accessed: 01- Feb- 2018].

Appendix A

Risk Assessment

UNIVERSITY OF Southampton

RECORD OF RISK ASSESSMENT

Title of the risk assessment	Smart wing project
Date risk assessment carried out	16/04/2018
Describe the work being assessed	Control system designed to suppress short period oscillations
Describe the location at which the work is being carried out	Building 13, Highfield Campus / Design studio, Boldrewood Campus
Where appropriate list the individuals doing the work and the dates/times when the work will be carried out	Bhagyesh Govilkar, Semester 2 2018
List any other generic or specific risk assessments or other documents that relate to this assessment-use hyperlinks if possible	
Name and post of risk assessor	Bhagyesh Govilkar, 3rd year Aeronautics and Astronautics student
List the names and posts of those assisting in compiling this risk assessment	T Glyn Thomas, Supervisor
Name, post and where required, signature of the responsible manager/supervisor approving the risk assessment	T Glyn Thomas, Supervisor
Reference number and version number of risk assessment	1

Page 1 of 9 HS/UOS/FR/038/04

FIGURE A.1: Section A

Assessment	
Title of risk assessment	Smart wing project

Risk Acceptability	
1-3	Risk acceptable
4-6	Risk to be reduced if readily possible
7-14	Risk to be reduced if reasonably practicable
15-25	Risk unacceptable

Risk Matrix			Severity				
			very low	low	medium	high	very high
			1	2	3	4	5
Likelihood	certainty	5	5	10	15	20	25
	likely	4	4	8	12	16	20
	possible	3	3	6	9	12	15
	less likely	2	2	4	6	8	10
	improbable	1	1	2	3	4	5

ref	Task/Aspect of work	Hazard	Harm and how it could arise	Who could be affected?	Existing measures to control risk	Risk Factors		Residual Risk score	Any changes or extra controls?
						Overall Likelihood	Overall Severity		
	Use of laser cutter	Fume inhalation could be hazardous	Shortly after the wood is cut, the laser cutter is filled with fumes. This could be harmful if inhaled	People in the workshop	Make sure the extractor fans are on. Do not lift the lid too early. Wait for the fumes to clear before taking out the laser cut parts	3	2	6	no
	Operation of Boldrewood wind tunnel	Risk of injury from parts breaking in the tunnel	The aerodynamic forces on the parts could cause the rig to fail and the wing could fly out of the tunnel potentially causing bodily harm	People in the vicinity of wind tunnel	Request induction and supervision from the lab technician before operating the wind tunnel. Fasten the parts in the wind tunnel securely. Use nuts, bolts, adhesives to make sure the structure will not fail when using the wind tunnel.	2	2	4	no
	Motors, servos, electromechanical devices.	Risk of injury from sudden or unexpected movement.	Risk of injury from sudden or unexpected movement of device or attached parts.	Person working with the device.	Do not apply power unless you've checked that it is safe to do so, remove powder when not in use.	2	1	2	no

FIGURE A.2: Section B

	Construction and use of light tools.	Cuts and scrapes.	Cuts and scrapes from sharp edges and blades.	Person using the tools.	Follow correct operating procedures and guidelines for good practice. Don't use power tools unless you've been shown and/or trained, for example attended a workshop course.	2	3	6	no
	Battery use and storage (particularly LiPos)	Stored energy	Heat from uncontrolled release of stored energy. Burns, fire, hot splashes.	Anyone close to the battery.	Follow manufacturer guidelines on charging, storage, and discharging. For LiPos: always charge using balancer, do not charge unattended, take special care to avoid shorts between terminals, avoid overheating, do not over-discharge.	1	4	4	no
	Lifting and manual handling	Lifting and manual handling	Injury to arms, legs, and joints from lifting, lowering, pushing, pulling and carrying.	Person lifting	Follow general guidelines on good handling techniques for lifting, see, for example, hse.gov.uk. Light lifting: adopt a stable posture, get good hold, don't flex the back, keep load close to body, avoid twisting, keep within safe/comfortable limits. Heavier Lifting: avoid, or use appropriate and suitable lifting aids. Seek advice.	1	3	3	no
	Soldering light electronics	Heat, smoke, solder and flux splashes.	Burns, smoke inhalation, splashes to eyes.	Person soldering	Use work holding clips and clamps as necessary, use eye protection, and ensure adequate ventilation.	2	2	4	no

FIGURE A.3: Section B

Post Risk Assessment Actions

Title of risk assessment
Smart wing project

Have any of the specialist control measures listed below been identified as required during this risk assessment? - indicate yes or no - if yes then include details on the post assessment action list below.	**yes/no**
is any exposure monitoring required?	no
Is any occupational health monitoring required?	no
Are there any hazards or other factors that could affect pregnant or nursing mothers?	no

Is any specific training required before people can carry out this work?	no

Are any additional procedures or risk assessments required as a result of this assessment?	no

Are any specialist disposal arrangements required?	no

Are any special emergency arrangements required?	no

	Post Assessment actions		
ref	**action**	**by whom**	**by when**

FIGURE A.4: Section C

Appendix B

Method Statement

This method statement is in reference to the project "Active Control of Wing Lift: Suppressing SPOs". The scope of the work to be done is as follows: analysis of SPOs, design of controller, design of wing and the rig, building the wing, the rig and the gust generator and then finally testing the controller in a wind tunnel.

Most of the risk arises from manufacturing the parts and testing the controller in the wind tunnel. The wing will be split into four parts: the central section, the two ends of the wing and the movable trailing edge flap. The central section of the wing has pressure tappings which complicate the geometry of the part. 3D printing will therefore be necessary to produce this part. The ends of the wing are the same profile as the 3D printed parts. Due to the simple geometry of the part, the same profile can be laser cut several times and the laser cut parts can be stuck together side to side in order to extend the span of the wing. The flap will also be made similarly since all features of the flap are two dimensional. A RC servo will actuate the flap. The wing will be mounted using some fixtures on the left and right walls of the wind tunnel. These are circular discs with holes in them for the carbon fibre rods and some more holes for the PVC tubes connected to the pressure tappings of the wing. These fixtures will be made exclusively by laser cutting.

The gust generator consists of two NACA0015 flaps which are 250mm in chord and 500mm in span. Due to the simple geometry, these can be foam cut. Laser cutting will be very inconvenient for this part due to the size of the flaps. The gust generator will operate by using two RC servos to deflect the flaps. Some carbon fibre rods will be used to provide structural support to the flaps. The rods will also serve as a shaft to connect the flap to the RC servos. The rods will be supported by a stand on either side of the flap as outlined in the diagram. These stands will be laser cut out of MDF.

Once the wing and gust generator is manufactured and mounted, the electronics need to be built. I will have to solder some differential pressure sensors onto a breadboard. These will be connected to a Raspberry Pi 3 (a microcontroller). The Raspberry Pi 3 will be connected to my laptop via WiFi through the Secure Shell protocol. This will allow me to monitor the data for my test. The Raspberry Pi will be running a code to implement my controller. It will use the pressure data to estimate the lift and remove the error by adjusting the flap angle through the RC servo mentioned above. The entire setup is outlined in the diagram on the next page.

The project will finish after testing the wing in the wind tunnel. I will use the Boldrewood wing tunnel (600mm by 400mm). I will firstly request induction and supervision from the lab supervisor. When testing, the wind tunnel will be turned on and the gust generator

will be programmed to produce an impulsive gust by deflecting the flaps very quickly. The system should attempt to correct the lift change caused by the gust by moving the flap.

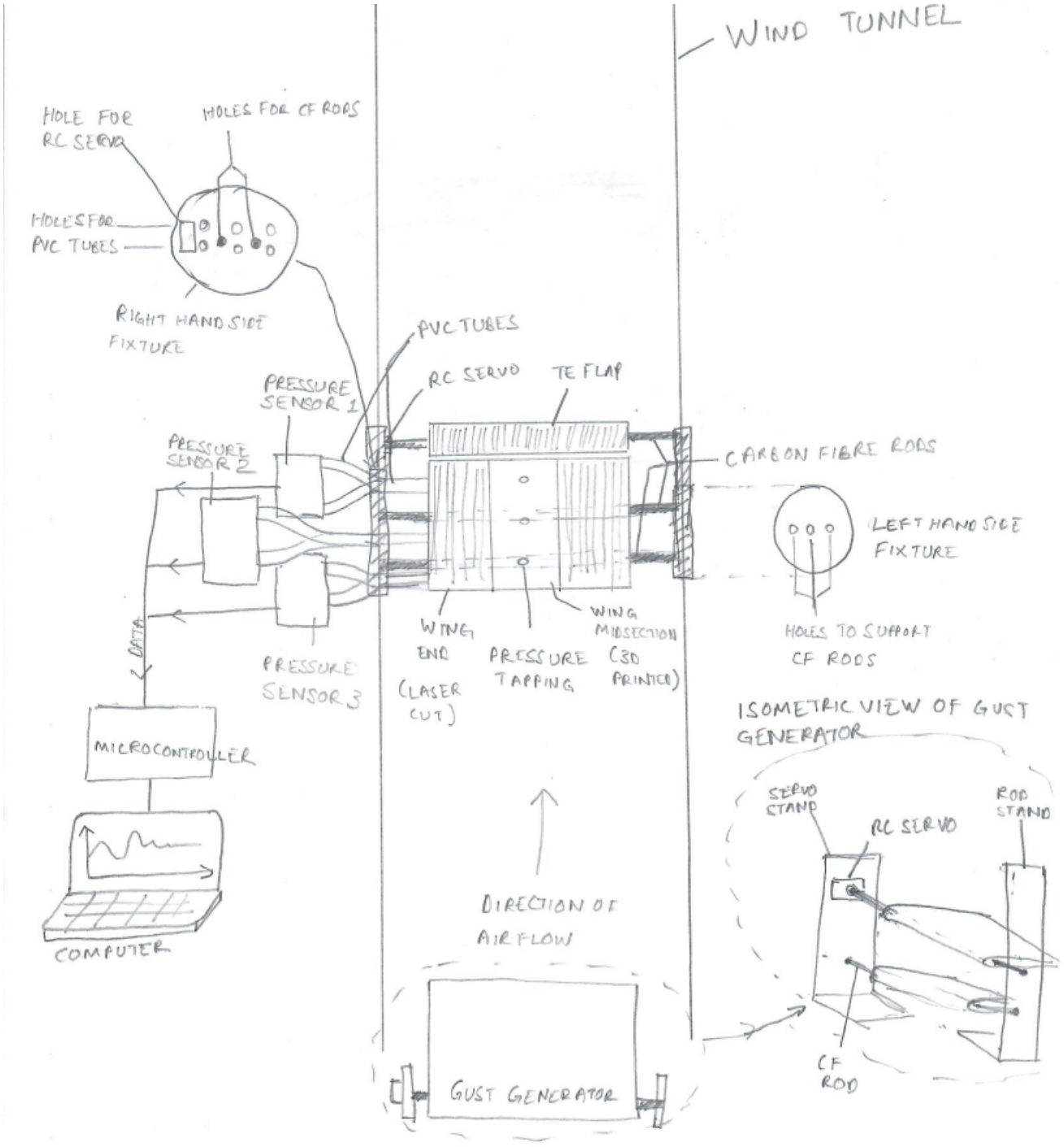


FIGURE B.1: Diagram of the setup

Appendix C

Lift Estimator

TABLE C.1: Lift estimator values and xfoil predicted values

delta	alpha	Cp1	Cp2	Cp3	p1	p2	p3	Cl	Actual Lift	Estimated Lift	Error^2	%Error
0	-5	-0.5629	-0.1619	-0.0523	-137.911	-39.6655	-12.8135	-0.3757	-2.485256	-2.328769747	0.024488	6.296565995
0	-4.5	-0.4513	-0.101	-0.0122	-110.569	-24.745	-2.989	-0.3034	-2.006991	-1.84326134	0.026807	8.157966835
0	-4	-0.3476	-0.0319	0.0351	-85.162	-7.8155	8.5995	-0.223	-1.475145	-1.399489283	0.005724	5.128696979
0	-3.5	-0.3279	0.0311	0.0736	-80.3355	7.6195	18.032	-0.1551	-1.025987	-1.409230972	0.146876	37.35375389
0	-3	-0.2003	0.0998	0.1116	-49.0735	24.451	27.342	-0.0901	-0.596012	-0.897299727	0.090775	50.55074051
0	-2.5	-0.099	0.0947	0.1499	-24.255	23.2015	36.7255	-0.0306	-0.202419	-0.111006851	0.008356	45.15986598
0	-2	-0.01	0.1603	0.1742	-2.45	39.2735	42.679	0.020367	0.1347255	0.134704751	4.31E-10	0.01540107
0	-1.5	0.0744	0.2118	0.1779	18.228	51.891	43.5855	0.071333	0.47187	0.307820941	0.026912	34.76573182
0	-1	0.1557	0.2578	0.2125	38.1465	63.161	52.0625	0.1223	0.8090145	0.685646144	0.01522	15.24921447
0	-0.5	0.2402	0.3048	0.2465	58.849	74.676	60.3925	0.1734	1.147041	1.070744217	0.005821	6.651617747
0	0	0.3389	0.3626	0.2914	83.0305	88.837	71.393	0.2391	1.5816465	1.536447515	0.002043	2.857717242
0	0.5	0.4536	0.4322	0.3497	111.132	105.889	85.6765	0.3213	2.1253995	2.101900546	0.000552	1.105625287
0	1	0.5769	0.5086	0.417	141.3405	124.607	102.165	0.413	2.731995	2.729507842	6.19E-06	0.091038158
0	1.5	0.6637	0.5559	0.458	162.6065	136.1955	112.21	0.4685	3.0991275	3.167909238	0.004731	2.219390401
0	2	0.7423	0.5977	0.4959	181.8635	146.4365	121.4955	0.516	3.41334	3.575360908	0.026251	4.746697009
0	2.5	0.8211	0.641	0.5364	201.1695	157.045	131.418	0.5634	3.726891	3.991559243	0.070049	7.101582601
0	3	0.9001	0.6867	0.5217	220.5245	168.2415	127.8165	0.6109	4.0411035	4.055605192	0.00021	0.358854756
0	3.5	0.9791	0.7356	0.4583	239.8795	180.222	112.2835	0.6584	4.355316	3.802069646	0.306082	12.70278331
0	4	1.0583	0.7882	0.4727	259.2835	193.109	115.8115	0.7058	4.668867	4.008183434	0.436503	14.15083286
0	4.5	1.1376	0.8446	0.5056	278.712	206.927	123.872	0.753	4.981095	4.307719099	0.453435	13.51863196
0	5	1.2175	0.9047	0.5365	298.2875	221.6515	131.4425	0.8004	5.294646	4.577571966	0.514195	13.54338013
0	5.5	1.2981	0.9683	0.5642	318.0345	237.2335	138.229	0.8474	5.605551	4.811974158	0.629764	14.15698193
0	6	1.3801	0.9185	0.5891	338.1245	225.0325	144.3295	0.894	5.91381	5.663436156	0.062687	4.233714715
0	6.5	1.4644	0.9275	0.6113	358.778	227.2375	149.7685	0.9388	6.210162	6.184974825	0.000634	0.405579992
0	7	1.5562	0.9788	0.6301	381.269	239.806	154.3745	0.9809	6.4886535	6.490330372	2.81E-06	0.025843141
0	7.5	1.6219	1.0202	0.6452	397.3655	249.949	158.074	1.02	6.7473	6.693090432	0.002939	0.803426084
0	8	1.6171	1.0535	0.6567	396.1895	258.1075	160.8915	1.0566	6.989409	6.555015548	0.188698	6.215024075
0	8.5	1.6936	1.0822	0.6655	414.932	265.139	163.0475	1.0912	7.218288	6.844889064	0.139427	5.172957019
0	8	1.7583	1.1083	0.6721	430.7835	271.5335	164.6645	1.1245	7.4385675	7.07475877	0.132357	4.890843973
0	9.5	1.8145	1.1292	0.6737	444.5525	276.654	165.0565	1.1531	7.6277565	7.2587733	0.136149	4.83737519
0	10	2.1254	1.7566	1.7207	520.723	430.367	421.5715	1.1817	7.8169455	11.83941768	16.18028	51.45836292
15	-5	0.4177	0.6491	0.7953	102.3365	159.0295	194.8485	0.728	4.81572	3.461323619	1.83439	28.12448358
15	-4.5	0.5579	0.7427	0.8796	136.6855	181.9615	215.502	0.7856	5.196744	4.185627561	1.022356	19.45672981
15	-4	0.7666	0.895	1.0023	187.817	219.275	245.5635	0.8432	5.577768	5.174937811	0.162272	7.222067849
15	-3.5	0.8569	0.9405	1.0341	209.9405	230.4225	253.3545	0.9008	5.958792	5.584684986	0.139956	6.278235821
15	-3	0.9472	0.9832	1.0652	232.064	240.884	260.974	0.9584	6.339816	6.005605243	0.111697	5.271616034
15	-2.5	1.0045	1.0116	1.0731	246.1025	247.842	262.9095	0.9841	6.5098215	6.192611228	0.100622	4.872795232
15	-2	1.0338	1.015	1.0337	253.281	248.675	253.2565	0.9804	6.485346	6.082245906	0.16249	6.215552625
15	-1.5	1.0824	1.0342	1.0464	265.188	253.379	256.368	0.9971	6.5958165	6.304831666	0.084672	4.411657513
15	-1	1.1377	1.0599	1.056	278.7365	259.6755	258.72	1.021	6.753915	6.506925645	0.061004	3.656980506
15	-0.5	1.2023	1.0984	1.0691	294.5635	269.108	261.9295	1.054	6.97221	6.707735617	0.069947	3.793264739
15	0	1.2673	1.0869	1.0566	310.4885	266.2905	258.867	1.0864	7.186536	7.02945959	0.024673	2.185704068
15	0.5	1.3339	1.1346	1.0581	326.8055	277.977	259.2345	1.1218	7.420707	7.118275142	0.091465	4.075512722
15	1	1.4024	1.1728	1.0755	343.588	287.336	263.4975	1.1583	7.6621545	7.367328922	0.086922	3.847815625
15	1.5	1.4771	1.2083	1.0918	361.8895	296.0335	267.491	1.1952	7.906248	7.656503396	0.062372	3.158825831
15	2	1.5129	1.2449	1.1099	370.6605	305.0005	271.9255	1.2352	8.170848	7.75013252	0.177002	5.148981838
15	2.5	1.6046	1.2801	1.1276	393.127	313.6245	276.262	1.2757	8.4387555	8.137229375	0.090918	3.573111281
15	3	1.6818	1.3176	1.1455	412.041	322.812	280.6475	1.3177	8.7165855	8.438085282	0.077562	3.195060935
15	3.5	1.7543	1.3546	1.1633	429.8035	331.877	285.0085	1.3601	8.9970615	8.71685976	0.078513	3.114369503
15	4	1.8288	1.3906	1.1802	448.056	340.697	289.149	1.4026	9.278199	9.00593292	0.074129	2.934471232
15	4.5	1.9024	1.4263	1.1967	466.088	349.4435	293.1915	1.4451	9.5593365	9.289562006	0.072778	2.822104797
15	5	1.9747	1.462	1.2116	483.8015	358.19	296.842	1.4869	9.8358435	9.556636504	0.077957	2.838668554
15	5.5	2.0451	1.4971	1.2259	501.0495	366.7895	300.3455	1.5279	10.107059	9.813538877	0.086154	2.904105316
15	6	2.1167	1.5305	1.2389	518.5915	374.9725	303.5305	1.5684	10.374966	10.07802334	0.088175	2.862107318
15	6.5	2.1844	1.5625	1.2505	535.178	382.8125	306.3725	1.6073	10.63229	10.32152241	0.096576	2.92286142
15	7	2.2507	1.5934	1.2605	551.4215	390.383	308.8225	1.6454	10.884321	10.55403418	0.109089	3.034519299
15	7.5	2.3164	1.6236	1.2693	567.518	397.782	310.9785	1.6822	11.127753	10.77993418	0.120978	3.125687794
15	8	2.3806	1.6513	1.2764	583.247	404.5685	312.718	1.7176	11.361924	11.00145733	0.129936	3.172584742
15	8.5	2.4403	1.6774	1.2811	597.8735	410.963	313.8695	1.7509	11.582204	11.19384998	0.150818	3.353019353
15	9	2.4986	1.7021	1.2838	612.157	417.0145	314.531	1.7829	11.793884	11.37445113	0.175924	3.556355003
15	9.5	2.5559	1.7243	1.2848	626.1955	422.4535	314.776	1.8135	11.996303	11.55325316	0.196293	3.693215816
15	10	2.6103	1.7441	1.2826	639.5235	427.3045	314.237	1.8412	12.179538	11.71067099	0.219836	3.849628825
30	-5	1.6576	1.6272	1.8181	406.112	398.664	445.4345	0.9964	6.591186	10.74303655	17.23786	62.9909481
30	-4.5	1.5026	1.4945	1.6624	368.137	366.1525	407.288	1.0399	6.8789385	9.718980337	8.065838	41.2860478
30	-4	1.4287	1.4285	1.5911	350.0315	349.9825	389.8195	1.0834	7.166691	9.263910112	4.398328	29.26342313

30	-3.5	1.1681	1.1739	1.2518	286.1845	287.6055	306.691	1.1269	7.4544435	7.238950837	0.046437	2.890794777
30	-3	1.2413	1.2164	1.2848	304.1185	298.018	314.776	1.1704	7.742196	7.584518907	0.024862	2.036593916
30	-2.5	1.3197	1.2644	1.3101	323.3265	309.778	320.9745	1.2139	8.0299485	7.879065893	0.022766	1.878998444
30	-2	1.4016	1.3162	1.3359	343.392	322.469	327.2955	1.2634	8.357391	8.173722429	0.033734	2.197678335
30	-1.5	1.4808	1.3268	1.358	362.796	325.066	332.71	1.3091	8.6596965	8.659504826	3.67E-08	0.002213403
30	-1	1.558	1.3779	1.3793	381.71	337.5855	337.9285	1.353	8.950095	8.906091479	0.001936	0.491654229
30	-0.5	1.6451	1.4207	1.4018	403.0495	348.0715	343.441	1.398	9.24777	9.257004064	8.53E-05	0.099851789
30	0	1.6922	1.4636	1.424	414.589	358.582	348.88	1.4433	9.5474295	9.399298728	0.021943	1.551525174
30	0.5	1.7804	1.5014	1.4443	436.198	367.843	353.8535	1.4861	9.8305515	9.769984295	0.003668	0.616111975
30	1	1.856	1.54	1.4631	454.72	377.3	358.4595	1.5282	10.109043	10.06205071	0.002208	0.464853965
30	1.5	1.9291	1.5774	1.4778	472.6295	386.463	362.061	1.5701	10.386212	10.32261727	0.004044	0.612294813
30	2	2.0044	1.6137	1.4835	491.078	395.3565	363.4575	1.6116	10.660734	10.5451882	0.013351	1.08384468
30	2.5	2.0787	1.6495	1.467	509.2815	404.1275	359.415	1.6526	10.931949	10.62866494	0.091981	2.774290851
30	3	2.1499	1.6842	1.4779	526.7255	412.629	362.0855	1.6925	11.195888	10.870967	0.105573	2.902141511
30	3.5	2.2209	1.7179	1.493	544.1205	420.8855	365.785	1.7315	11.453873	11.14363068	0.09625	2.708619499
30	4	2.2893	1.7496	1.5061	560.8785	428.652	368.9945	1.7692	11.703258	11.40163424	0.090977	2.577263144
30	4.5	2.3546	1.7811	1.5179	576.877	436.3695	371.8855	1.8057	11.944706	11.63675675	0.094832	2.578119246
30	5	2.421	1.81	1.5278	593.145	443.45	374.311	1.8409	12.177554	11.88022796	0.088402	2.441586801
30	5.5	2.4823	1.8377	1.5357	608.1635	450.2365	376.2465	1.8746	12.400479	12.09172659	0.095328	2.489842574
30	6	2.5465	1.863	1.5406	623.8925	456.435	377.447	1.906	12.60819	12.31297415	0.087152	2.341460955
30	6.5	2.6051	1.8871	1.5447	638.2495	462.3395	378.4515	1.9371	12.813917	12.5070608	0.09416	2.394706545
30	7	2.662	1.9092	1.5449	652.19	467.754	378.5005	1.9654	13.001121	12.67942738	0.103487	2.474352964
30	7.5	2.7149	1.9276	1.5407	665.1505	472.262	377.4715	1.9903	13.165835	12.82453819	0.116483	2.592287748
30	8	2.7652	1.9429	1.5336	677.474	476.0105	375.732	2.0132	13.317318	12.95552967	0.130891	2.716675588
30	8.5	2.8135	1.9586	1.5265	689.3075	479.857	373.9925	2.0367	13.472771	13.07399834	0.159019	2.959837814
30	9	2.8571	1.969	1.5121	699.9895	482.405	370.4645	2.055	13.593825	13.1525925	0.194686	3.245830361
30	9.5	2.8951	1.975	1.4924	709.2995	483.875	365.638	2.0698	13.691727	13.19401104	0.247721	3.635158396
30	10	2.935	1.982	1.474	719.075	485.59	361.13	2.0867	13.803521	13.24770024	0.308936	4.02665584
45	-5	2.0434	1.952	2.1479	500.633	478.24	526.2355	1.3171	8.7126165	12.96676264	18.09776	48.82742333
45	-4.5	1.9784	1.8978	2.0908	484.708	464.961	512.246	1.3595	8.9930925	12.57976479	12.86422	39.88252412
45	-4	1.9245	1.8549	2.0427	471.5025	454.4505	500.4615	1.4019	9.2735685	12.2429255	8.817081	32.01957266
45	-3.5	1.888	1.8055	1.9825	462.56	442.3475	485.7125	1.4443	9.5540445	11.95722519	5.775277	25.15354293
45	-3	1.834	1.7343	1.8848	449.33	424.9035	461.776	1.4867	9.8345205	11.47093685	2.677858	16.63951334
45	-2.5	1.7773	1.6115	1.6686	435.4385	394.8175	408.807	1.5291	10.114997	10.52636661	0.169225	4.066932771
45	-2	1.8208	1.6509	1.6843	446.096	404.4705	412.6535	1.5715	10.395473	10.62943028	0.054736	2.250573832
45	-1.5	1.9057	1.6898	1.6993	466.8965	414.001	416.3285	1.6134	10.672641	10.94438361	0.073844	2.546160874
45	-1	1.9805	1.7271	1.7121	485.2225	423.1395	419.4645	1.6533	10.93658	11.20256753	0.07075	2.432095241
45	-0.5	2.053	1.7623	1.7236	502.985	431.7635	422.282	1.6921	11.193242	11.45250144	0.067216	2.316218554
45	0	2.1234	1.7959	1.7339	520.233	439.9955	424.8055	1.7298	11.442627	11.69306728	0.06272	2.188660691
45	0.5	2.1931	1.8287	1.7432	537.3095	448.0315	427.084	1.7667	11.686721	11.92829036	0.058356	2.067045792
45	1	2.2619	1.8599	1.7518	554.1655	455.6755	429.191	1.8024	11.922876	12.16341082	0.057857	2.01742285
45	1.5	2.3277	1.8892	1.7583	570.2865	462.854	430.7835	1.8368	12.150432	12.38064024	0.052996	1.894650634
45	2	2.3917	1.9186	1.7633	585.9665	470.057	432.0085	1.8696	12.367404	12.57880014	0.044688	1.709300821
45	2.5	2.4531	1.9445	1.7658	601.0095	476.4025	432.621	1.9004	12.571146	12.76751578	0.038561	1.562067492
45	3	2.5151	1.9691	1.767	616.1995	482.4295	432.915	1.9301	12.767612	12.95850832	0.036442	1.495164698
45	3.5	2.5735	1.9917	1.7668	630.5075	487.9665	432.866	1.9583	12.954155	13.13338	0.032122	1.383536867
45	4	2.63	2.0135	1.764	644.35	493.3075	432.18	1.9851	13.131437	13.28686988	0.02416	1.183673882
45	4.5	2.6848	2.0334	1.7597	657.776	498.183	431.1265	2.0104	13.298796	13.43286548	0.017975	1.008132482
45	5	2.7338	2.0494	1.7494	669.781	502.103	428.603	2.0317	13.439696	13.53357897	0.008814	0.698553577
45	5.5	2.7862	2.0646	1.7386	682.619	505.827	425.957	2.0538	13.585887	13.65316592	0.004526	0.495211849
45	6	2.8305	2.0742	1.7189	693.4725	508.179	421.1305	2.0706	13.697019	13.7071563	0.000103	0.074010968
45	6.5	2.8754	2.0863	1.6969	704.473	511.1435	415.7405	2.0896	13.822704	13.73625234	0.007474	0.625432367
45	7	2.9151	2.0924	1.6674	714.1995	512.638	408.513	2.104	13.91796	13.72553546	0.037027	1.382562786
45	7.5	2.9506	2.0948	1.6262	722.897	513.226	398.419	2.1152	13.992048	13.64157963	0.122828	2.504768216
45	8	2.9859	2.0968	1.5853	731.5455	513.716	388.3985	2.1275	14.073413	13.56065199	0.262923	3.643469634
45	8.5	3.0167	2.0946	1.5436	739.0915	513.177	378.182	2.1367	14.134271	13.47482395	0.43487	4.665586039
45	9	3.038	2.0853	1.4983	744.31	510.8985	367.0835	2.1408	14.161392	13.35711295	0.646865	5.679378512
45	9.5	3.0624	2.0738	1.4491	750.288	508.081	355.0295	2.1453	14.19116	13.24353395	0.897994	6.677576647
45	10	3.0852	2.0613	1.3962	755.874	505.0185	342.069	2.1502	14.223573	13.10445214	1.252431	7.86807126
60	-5	2.8981	2.6236	2.7989	710.0345	642.782	685.7305	1.5257	10.092506	17.66801805	57.38839	75.06077212
60	-4.5	2.0694	1.9289	2.0895	507.003	472.5805	511.9275	1.5613	10.328	12.86892654	6.45631	24.60231568
60	-4	2.0209	1.8818	2.031	495.1205	461.041	497.595	1.5969	10.563494	12.51911061	3.824438	18.51297686
60	-3.5	1.9676	1.8034	1.9045	482.062	441.833	466.6025	1.6325	10.798988	11.89890063	1.209809	10.1853357
60	-3	2.0345	1.8354	1.9131	498.4525	449.673	468.7095	1.6681	11.034482	12.11980188	1.17792	9.83571707
60	-2.5	2.104	1.87	1.9244	515.48	458.15	471.478	1.7037	11.269976	12.35635581	1.180222	9.639597786
60	-2	2.1718	1.904	1.9321	532.091	466.48	473.3645	1.7384	11.499516	12.56529523	1.135885	9.268035546
60	-1.5	2.2379	1.936	1.9377	548.2855	474.32	474.7365	1.7721	11.722442	12.76359866	1.084008	8.881743276
60	-1	2.3036	1.9658	1.9405	564.382	481.621	475.4225	1.8029	11.926184	12.95476366	1.057977	8.624554202
60	-0.5	2.3633	1.9935	1.9401	579.0085	488.4075	475.3245	1.8328	12.123972	13.10690366	0.966155	8.107340199
60	0	2.4253	2.0194	1.9376	594.1985	494.753	474.712	1.8645	12.333668	13.26792345	0.872834	7.574843
60	0.5	2.485	2.0432	1.9348	608.825	500.584	474.026	1.8926	12.519549	13.42685105	0.823197	7.247082566
60	1	2.5439	2.0664	1.9303	623.2555	506.268	472.9235	1.9198	12.699477	13.57450391	0.765672	6.89025942
60	1.5	2.5991	2.0871	1.9226	636.7795	511.3395	471.037	1.9463	12.874775	13.69720121	0.676386	6.38789216
60	2	2.6515	2.1055	1.9118	649.6175	515.8475	468.391	1.9693	13.02692	13.79909266	0.596251	5.927519278
60	2.5	2.7039	2.1223	1.9	662.4555	519.9635	465.5	1.9915	13.173773	13.90367404	0.532756	5.540565866
60	3	2.7536	2.138	1.8838	674.632	523.81	461.531	2.0136	13.319964	13.97332414	0.426879	4.905119384
60	3.5	2.8005	2.1498	1.8686	686.1225	526.701	457.807	2.0317	13.439696	14.05626321	0.380156	4.587661277
60	4	2.8472	2.1607	1.8466	697.564	529.3715	452.417	2.0504	13.563396	14.10127485	0.289314	3.965664994
60	4.5	2.8864	2.1679	1.8244	707.168	531.1355	446.978	2.0652	13.661298	14.12684954	0.216738	3.407813381
60	5	2.9289	2.175	1.796	717.5805	532.875	440.02	2.081	13.765815	14.13181167	0.133954	2.658735901
60	5.5	2.9654	2.1775	1.7661	726.523	533.4875	432.6945	2.0929	13.844534	14.12204169	0.077011	2.004460388
60	6	2.9975	2.176	1.7282	734.3875	533.12	423.409	2.1026	13.908699	14.06244456	0.023638	1.105391387
60	6.5	3.0307	2.1759	1.6877	742.5215	533.0955	413.4865	2.1138	13.982787	13.9847663	3.92E-06	0.014155228
60	7	3.0584	2.1706	1.6434	749.308	531.797	402.633	2.122	14.03703	13.88408856	0.023391	1.089556968
60	7.5	3.0848	2.1614	1.59	755.776	529.543	389.55	2.1282	14.078043	13.74223895	0.112764	2.385303502
60	8	3.102	2.1449	1.5254	759.99	525.5005	373.723	2.1306	14.093919	13.52435951	0.324398	4.041171851
60	8.5	3.1239	2.1291	1.4662	765.3555	521.6295	359.219	2.1348	14.121702	13.360092	0.58005	5.393188429
60	9	3.1387	2.1082	1.404	768.9815	516.509	343.98	2.1372	14.137578	13.16894959	0.938241	6.851445182

60	9.5	3.1515	2.0817	1.344	772.1175	510.0165	329.28	2.1387	14.147501	13.01201063	1.289337	8.026081125
60	10	3.1578	2.047	1.3008	773.661	501.515	318.696	2.1367	14.134271	12.97035745	1.354694	8.234687791
-15	-5	-1.5325	-0.9498	-0.8876	-375.463	-232.701	-217.462	-1.2972	-8.580978	-8.114151402	0.217927	5.440249328
-15	-4.5	-1.4534	-0.9063	-0.8626	-356.083	-222.044	-211.337	-1.245	-8.235675	-7.792958666	0.195998	5.375592573
-15	-4	-1.3735	-0.8624	-0.8371	-336.508	-211.288	-205.09	-1.1924	-7.887726	-7.466774604	0.1772	5.336790298
-15	-3.5	-1.2929	-0.8183	-0.8121	-316.761	-200.484	-198.965	-1.1396	-7.538454	-7.141166614	0.157837	5.270144062
-15	-3	-1.212	-0.7733	-0.7867	-296.94	-189.459	-192.742	-1.0865	-7.187198	-6.816525721	0.137398	5.157389638
-15	-2.5	-1.131	-0.7287	-0.761	-277.095	-178.532	-186.445	-1.0333	-6.83528	-6.48731	0.121083	5.090786709
-15	-2	-1.0478	-0.6837	-0.7353	-256.711	-167.507	-180.149	-0.9798	-6.481377	-6.148964248	0.110498	5.128736564
-15	-1.5	-0.9662	-0.6381	-0.7095	-236.719	-156.335	-173.828	-0.9263	-6.127475	-5.821569323	0.093578	4.99235333
-15	-1	-0.8834	-0.5924	-0.6832	-216.433	-145.138	-167.384	-0.8724	-5.770926	-5.485461942	0.08149	4.94659016
-15	-0.5	-0.8007	-0.5459	-0.6569	-196.172	-133.746	-160.941	-0.8186	-5.415039	-5.15429398	0.067988	4.815201153
-15	0	-0.7158	-0.4991	-0.6304	-175.371	-122.28	-154.448	-0.7645	-5.057168	-4.812211437	0.060003	4.843740345
-15	0.5	-0.633	-0.4481	-0.6039	-155.085	-109.785	-147.956	-0.7105	-4.699958	-4.504180656	0.038329	4.165502441
-15	1	-0.5488	-0.4199	-0.5768	-134.456	-102.876	-141.316	-0.6558	-4.338117	-4.059157164	0.077819	6.430435974
-15	1.5	-0.463	-0.3702	-0.5498	-113.435	-90.699	-134.701	-0.601	-3.975615	-3.725392356	0.062611	6.293935496
-15	2	-0.379	-0.3189	-0.5225	-92.855	-78.1305	-128.013	-0.5461	-3.612452	-3.407907741	0.041838	5.662187002
-15	2.5	-0.2943	-0.2702	-0.4952	-72.1035	-66.199	-121.324	-0.4914	-3.250611	-3.072436728	0.031746	5.481254823
-15	3	-0.2083	-0.2228	-0.4677	-51.0335	-54.586	-114.587	-0.4362	-2.885463	-2.721843284	0.026771	5.670483948
-15	3.5	-0.1226	-0.1757	-0.4404	-30.037	-43.0465	-107.898	-0.3812	-2.521638	-2.372369084	0.022281	5.919522005
-15	4	-0.0371	-0.1276	-0.4126	-9.0895	-31.262	-101.087	-0.3258	-2.155167	-2.026376902	0.016587	5.975875558
-15	4.5	0.0487	-0.08	-0.3853	11.9315	-19.6	-94.3985	-0.2707	-1.790681	-1.679152079	0.012439	6.228270255
-15	5	0.1348	-0.0325	-0.3572	33.026	-7.9625	-87.514	-0.2149	-1.421564	-1.324901381	0.009344	6.799704605
-15	5.5	0.2346	0.0157	-0.3291	57.477	3.8465	-80.6295	-0.1588	-1.050462	-0.903892049	0.021483	13.95290371
-15	6	0.2959	0.0635	-0.3007	72.4955	15.5575	-73.6715	-0.1024	-0.677376	-0.677307862	4.64E-09	0.010059167
-15	6.5	0.3864	0.1113	-0.2729	94.668	27.2685	-66.8605	-0.0461	-0.304952	-0.303879529	1.15E-06	0.351521911
-15	7	0.4761	0.1578	-0.2443	116.6445	38.661	-59.8535	0.0102	0.067473	0.077539506	0.000101	14.91930995
-15	7.5	0.5618	0.2072	-0.2165	137.641	50.764	-53.0425	0.066	0.43659	0.417373981	0.000369	4.401387826
-15	8	0.6458	0.2619	-0.188	158.221	64.1655	-46.06	0.12255	0.8106683	0.723446521	0.007608	10.75923827
-15	8.5	0.7319	0.3011	-0.1599	179.3155	73.7695	-39.1755	0.1791	1.1847465	1.123594909	0.00374	5.161576015
-15	9	0.816	0.3466	-0.1312	199.92	84.917	-32.144	0.23565	1.5588248	1.482289105	0.005858	4.909829955
-15	9.5	0.9018	0.394	-0.1008	220.941	96.53	-24.696	0.2922	1.932903	1.849709149	0.006921	4.304088231
-15	10	0.9354	0.4119	-0.0862	229.173	100.9155	-21.119	0.34875	2.3069813	2.013851972	0.085925	12.7061838
-30	-5	-2.0184	-1.3441	-1.3023	-494.508	-329.305	-319.064	-1.7167	-11.35597	-10.99241219	0.132175	3.201472836
-30	-4.5	-1.9454	-1.3052	-1.2828	-476.623	-319.774	-314.286	-1.6698	-11.04573	-10.71109839	0.111976	3.029484696
-30	-4	-1.872	-1.2659	-1.263	-458.64	-310.146	-309.435	-1.6229	-10.73548	-10.42808701	0.094493	2.863368877
-30	-3.5	-1.7979	-1.2275	-1.2431	-440.486	-300.738	-304.56	-1.5757	-10.42326	-10.13587417	0.082588	2.757116773
-30	-3	-1.7215	-1.1858	-1.2216	-421.768	-290.521	-299.292	-1.5269	-10.10044	-9.840199792	0.067727	2.576557236
-30	-2.5	-1.6461	-1.145	-1.1999	-403.295	-280.525	-293.976	-1.4782	-9.778293	-9.543472427	0.055141	2.401447502
-30	-2	-1.5689	-1.1042	-1.1786	-384.381	-270.529	-288.757	-1.4299	-9.458789	-9.2399284	0.0479	2.313828036
-30	-1.5	-1.4924	-1.0634	-1.1573	-365.638	-260.533	-283.539	-1.3818	-9.140607	-8.93999318	0.040246	2.194753801
-30	-1	-1.415	-1.0218	-1.1361	-346.675	-250.341	-278.345	-1.3335	-8.821103	-8.640457721	0.032633	2.047870759
-30	-0.5	-1.339	-0.9805	-1.1145	-328.055	-240.223	-273.053	-1.285	-8.500275	-8.344017796	0.024416	1.838260577
-30	0	-1.2584	-0.9372	-1.0908	-308.308	-229.614	-267.246	-1.2347	-8.167541	-8.021991103	0.021185	1.78204684
-30	0.5	-1.1805	-0.8949	-1.0677	-289.223	-219.251	-261.587	-1.1852	-7.840098	-7.712048949	0.016397	1.633258301
-30	1	-1.1016	-0.8513	-1.0457	-269.892	-208.569	-256.197	-1.1363	-7.516625	-7.410913755	0.011175	1.406359259
-30	1.5	-1.0242	-0.8036	-1.0235	-250.929	-196.882	-250.758	-1.0874	-7.193151	-7.13895251	0.002937	0.753473551
-30	2	-0.9441	-0.7787	-1.0012	-231.305	-190.782	-245.294	-1.0386	-6.870339	-6.726375394	0.020726	2.095436713
-30	2.5	-0.8656	-0.7334	-0.9789	-212.072	-179.683	-239.831	-0.9895	-6.545543	-6.434855762	0.012252	1.69102467
-30	3	-0.7838	-0.6876	-0.9546	-192.031	-168.462	-233.877	-0.9387	-6.209501	-6.116772452	0.008598	1.493325399
-30	3.5	-0.7044	-0.6417	-0.9301	-172.578	-157.217	-227.875	-0.8877	-5.872136	-5.810383617	0.003813	1.051608616
-30	4	-0.6242	-0.5986	-0.9071	-152.929	-146.657	-222.24	-0.838	-5.54337	-5.493623608	0.002475	0.897403424
-30	4.5	-0.5446	-0.5546	-0.8847	-133.427	-135.877	-216.752	-0.7887	-5.217251	-5.188628414	0.000819	0.548604782
-30	5	-0.4662	-0.5122	-0.8618	-114.219	-125.489	-211.141	-0.7393	-4.89047	-4.877893089	0.000158	0.257161622
-30	5.5	-0.3855	-0.4692	-0.8392	-94.4475	-114.954	-205.604	-0.6899	-4.563689	-4.560465505	1.04E-05	0.070622584
-30	6	-0.3047	-0.4257	-0.8158	-74.6515	-104.297	-199.871	-0.6369	-4.213094	-4.24036104	0.000744	0.647209458
-30	6.5	-0.2185	-0.3788	-0.7882	-53.5325	-92.806	-193.109	-0.5839	-3.862499	-3.885355804	0.000522	0.591775097
-30	7	-0.1375	-0.3351	-0.7649	-33.6875	-82.0995	-187.401	-0.5336	-3.529764	-3.565942001	0.001309	1.024941083
-30	7.5	-0.0659	-0.2908	-0.7422	-16.1455	-71.246	-181.839	-0.4829	-3.194384	-3.302001931	0.011582	3.368989078
-30	8	0.0194	-0.2501	-0.7204	4.753	-61.2745	-176.498	-0.4334	-2.866941	-2.953066946	0.007418	3.004105991
-30	8.5	0.0973	-0.2066	-0.6972	23.8385	-50.617	-170.814	-0.3839	-2.539499	-2.649144821	0.012022	4.317636751
-30	9	0.1794	-0.1617	-0.6737	43.953	-39.6165	-165.057	-0.32767	-2.167515	-2.32946428	0.026228	7.471656705
-30	9.5	0.2929	-0.0923	-0.6216	71.7605	-22.6135	-152.292	-0.27143	-1.795532	-1.807270314	0.000138	0.653779332
-30	10	0.3548	-0.0631	-0.6156	86.926	-15.4595	-150.822	-0.2152	-1.423548	-1.612673044	0.035768	13.28547012
-45	-5	-2.4015	-1.6613	-1.6289	-588.368	-407.019	-399.081	-2.0095	-13.29284	-13.22454036	0.004665	0.513826469
-45	-4.5	-2.3375	-1.6291	-1.618	-572.688	-399.13	-396.41	-1.9671	-13.01237	-13.00553286	4.67E-05	0.052516541
-45	-4	-2.2679	-1.5945	-1.6044	-555.636	-390.653	-393.078	-1.9247	-12.73189	-12.75430039	0.000502	0.176013838
-45	-3.5	-2.2001	-1.5597	-1.59	-539.025	-382.127	-389.55	-1.8815	-12.44612	-12.50852742	0.003894	0.501400531
-45	-3	-2.1317	-1.5237	-1.5751	-522.267	-373.307	-385.9	-1.8378	-12.15705	-12.26321809	0.011272	0.873329636
-45	-2.5	-2.0583	-1.4868	-1.5593	-504.284	-364.266	-382.029	-1.7932	-11.86202	-11.99156638	0.016783	1.092127685
-45	-2	-1.9846	-1.4473	-1.5409	-486.227	-354.589	-377.521	-1.7461	-11.55045	-11.71673529	0.02765	1.439630196
-45	-1.5	-1.9082	-1.4072	-1.5207	-467.509	-344.764	-372.572	-1.6973	-11.22764	-11.42021833	0.037087	1.715220982
-45	-1	-1.8349	-1.3676	-1.5023	-449.551	-335.062	-368.064	-1.6506	-10.91872	-11.14800239	0.052571	2.099911112
-45	-0.5	-1.7592	-1.3287	-1.4847	-431.004	-325.532	-363.752	-1.6046	-10.61443	-10.86446877	0.06252	2.355659193
-45	0	-1.6861	-1.2897	-1.4673	-413.095	-315.977	-359.489	-1.5588	-10.31146	-10.59612384	0.081032	2.760635143
-45	0.5	-1.6119	-1.2505	-1.4496	-394.916	-306.373	-355.152	-1.513	-10.0085	-10.32136655	0.097889	3.126059938
-45	1	-1.5365	-1.2104	-1.4319	-376.443	-296.548	-350.816	-1.467	-9.704205	-10.04539959	0.116414	3.515945837
-45	1.5	-1.4594	-1.1689	-1.4141	-357.553	-286.381	-346.455	-1.4208	-9.398592	-9.76779438	0.13631	3.928273298
-45	2	-1.3856	-1.1267	-1.3962	-339.472	-276.042	-342.069	-1.3744	-9.091656	-9.510457226	0.175394	4.606435025
-45	2.5	-1.3074	-1.1013	-1.378	-320.313	-269.819	-337.61	-1.3279	-8.784059	-9.135687443	0.123643	4.003035082
-45	3	-1.2317	-1.059	-1.36	-301.767	-259.455	-333.2	-1.2809	-8.473154	-8.868492156	0.156293	4.665779467
-45	3.5	-1.1537	-1.0165	-1.3416	-282.657	-249.043	-328.692	-1.2338	-8.161587	-8.588082197	0.181898	5.225640513
-45	4	-1.0774	-0.9744	-1.3227	-263.963	-238.728	-324.062	-1.1865	-7.848698	-8.311145636	0.213858	5.892036688
-45	4.5	-0.9988	-0.9324	-1.3045	-244.706	-228.438	-319.603	-1.1387	-7.532501	-8.026109057	0.243649	6.553050432
-45	5	-0.9209	-0.8907	-1.2861	-225.621	-218.222	-315.095	-1.0906	-7.214319	-7.741790767	0.278226	7.311456111
-45	5.5	-0.8416	-0.8482	-1.2669	-206.192	-207.809	-310.391	-1.042	-6.89283	-7.449752478	0.310163	8.079736153
-45	6	-0.7622	-0.8064	-1.2477	-186.739	-197.568	-305.687	-0.9929	-6.568034	-7.153327754	0.342569	8.911255617
-45	6.5	-0.6819	-0.7612	-1.2268	-167.066	-186.494	-300.566	-0.9414	-6.227361	-6.860596299	0.400987	10.16859788

-45	7	-0.5961	-0.7155	-1.2034	-146.045	-175.298	-294.833	-0.8877	-5.872136	-6.526880287	0.428691	11.15002859
-45	7.5	-0.5162	-0.6721	-1.1849	-126.469	-164.665	-290.301	-0.8374	-5.539401	-6.241036067	0.492292	12.66626243
-45	8	-0.436	-0.6292	-1.1672	-106.82	-154.154	-285.964	-0.7873	-5.20799	-5.955806554	0.55923	14.35903536
-45	8.5	-0.3552	-0.5867	-1.1506	-87.024	-143.742	-281.897	-0.73765	-4.879555	-5.672045445	0.628042	16.24104526
-45	9	-0.2762	-0.5447	-1.133	-67.669	-133.452	-277.585	-0.688	-4.55112	-5.388641384	0.701442	18.40253352
-45	9.5	-0.1912	-0.5016	-1.1151	-46.844	-122.892	-273.2	-0.6367	-4.211771	-5.078540163	0.75129	20.57969833
-45	10	-0.1036	-0.4565	-1.0963	-25.382	-111.843	-268.594	-0.5854	-3.872421	-4.760552442	0.788777	22.93478531
-60	-5	-3.9453	-2.9049	-2.9584	-966.599	-711.701	-724.808	-2.1784	-14.41012	-22.49342278	65.33985	56.09466837
-60	-4.5	-2.6361	-1.8808	-1.8588	-645.845	-460.796	-455.406	-2.1433	-14.17793	-14.63588978	0.209728	3.230092768
-60	-4	-2.5791	-1.8553	-1.8561	-631.88	-454.549	-454.745	-2.1082	-13.94574	-14.46641456	0.271099	3.733551965
-60	-3.5	-2.5185	-1.8281	-1.8521	-617.033	-447.885	-453.765	-2.0731	-13.71356	-14.27977533	0.320604	4.128898484
-60	-3	-2.458	-1.799	-1.8474	-602.21	-440.755	-452.613	-2.0365	-13.47145	-14.09984786	0.394887	4.664683276
-60	-2.5	-2.3942	-1.7698	-1.8409	-586.579	-433.601	-451.021	-1.999	-13.22339	-13.89237634	0.447549	5.059153475
-60	-2	-2.3285	-1.7381	-1.8345	-570.483	-425.835	-449.453	-1.9602	-12.96672	-13.68954988	0.522479	5.574476119
-60	-1.5	-2.2627	-1.7066	-1.8271	-554.362	-418.117	-447.64	-1.9206	-12.70477	-13.47894408	0.599347	6.093578533
-60	-1	-2.1957	-1.6729	-1.8176	-537.947	-409.861	-445.312	-1.8805	-12.43951	-13.26138597	0.675484	6.607001698
-60	-0.5	-2.1266	-1.6392	-1.8076	-521.017	-401.604	-442.862	-1.8393	-12.16697	-13.02992253	0.744688	7.092588109
-60	0	-2.0578	-1.604	-1.7966	-504.161	-392.98	-440.167	-1.7971	-11.88782	-12.80214266	0.835992	7.691287662
-60	0.5	-1.9851	-1.566	-1.7837	-486.35	-383.67	-437.007	-1.7523	-11.59146	-12.55804027	0.934269	8.33868554
-60	1	-1.9125	-1.5259	-1.7703	-468.563	-373.846	-433.724	-1.7075	-11.29511	-12.32298719	1.056526	9.100172178
-60	1.5	-1.8391	-1.4954	-1.7561	-450.58	-366.373	-430.245	-1.6631	-11.00141	-12.025797	1.049376	9.311450326
-60	2	-1.7665	-1.4648	-1.7417	-432.793	-358.876	-426.717	-1.6192	-10.71101	-11.73205259	1.042532	9.532665729
-60	2.5	-1.6932	-1.4271	-1.7269	-414.834	-349.64	-423.091	-1.5752	-10.41995	-11.47149812	1.105758	10.09170217
-60	3	-1.6207	-1.3887	-1.7117	-397.072	-340.232	-419.367	-1.5306	-10.12492	-11.21647576	1.191496	10.78089375
-60	3.5	-1.5457	-1.3494	-1.6935	-378.697	-330.603	-414.908	-1.4854	-9.825921	-10.93506819	1.230207	11.28797177
-60	4	-1.4706	-1.3112	-1.676	-360.297	-321.244	-410.62	-1.4411	-9.532877	-10.65137272	1.251034	11.73304012
-60	4.5	-1.3958	-1.273	-1.654	-341.971	-311.885	-405.23	-1.3959	-9.233879	-10.34151369	1.226856	11.99534071
-60	5	-1.3199	-1.2337	-1.6287	-323.376	-302.257	-399.032	-1.35	-8.93025	-10.01174567	1.169633	12.11047479
-60	5.5	-1.245	-1.1956	-1.6079	-305.025	-292.922	-393.936	-1.3045	-8.629268	-9.708207497	1.164112	12.50326284
-60	6	-1.1673	-1.1554	-1.5812	-285.989	-283.073	-387.394	-1.2575	-8.318363	-9.365515598	1.09653	12.58845233
-60	6.5	-1.0902	-1.1163	-1.5578	-267.099	-273.494	-381.661	-1.211	-8.010765	-9.040154954	1.059644	12.85008303
-60	7	-1.0137	-1.0767	-1.5327	-248.357	-263.792	-375.512	-1.164	-7.69986	-8.710184195	1.020755	13.12133201
-60	7.5	-0.9339	-1.0361	-1.5041	-228.806	-253.845	-368.505	-1.1152	-7.377048	-8.347177945	0.941152	13.15065247
-60	8	-0.8534	-0.9955	-1.4756	-209.083	-243.898	-361.522	-1.0663	-7.053575	-7.98117867	0.860449	13.1508382
-60	8.5	-0.7758	-0.9549	-1.4468	-190.071	-233.951	-354.466	-1.0172	-6.728778	-7.628282823	0.809109	13.36802645
-60	9	-0.6888	-0.9106	-1.411	-168.756	-223.097	-345.695	-0.9623	-6.365615	-7.204281494	0.703362	13.17495733
-60	9.5	-0.6033	-0.8667	-1.3678	-147.809	-212.342	-335.111	-0.9053	-5.98856	-6.740233495	0.565014	12.55183312
-60	10	-0.523	-0.8271	-1.3392	-128.135	-202.64	-328.104	-0.8561	-5.663102	-6.369119683	0.498462	12.46698797
											296.8883	7.317886888

Appendix D

Code

```
 /* UAV Smart Wing Control Program - Testing
 * -Bhagyesh Govilkar
 *
 * Function daq is data acquisition only. PID control occurs in the "control"
    function. Use estimateLift function to estimate lift and print data to file
    and to console in printdata.
 * All functions should be called from within the while loop in the main
    function to avoid recursion.
 *
 * TEST SEQUENCE:
 * Gust response with FP model guesses....
 * Gust response with second guesses....
 * Gust response with no control....
 * */


#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <stdio.h>
#include <stdint.h>
#include <errno.h>
#include <string.h>
#include <math.h>
#include <time.h>

uint8_t data[4]; // 4 byte word
uint8_t data2[4]; // 4 byte word
uint8_t data3[4]; // 4 byte word
uint8_t data4[4]; // 4 byte word

unsigned int pres; // raw pressure
double p_out; // pressure in kPa
unsigned int pres2; // raw pressure
double p_out2; // pressure in kPa
unsigned int pres3; // raw pressure
double p_out3; // pressure in kPa
double p_o3l;
unsigned int pres4; // raw pressure
double p_out4; // pressure in kPa
double rho = 1.225; //density in kgm^-3
double airspeed;
double aas;
double ap1;
double ap2;
double ap3;
int i=0;
double L;
```

```
double Kp=0.14324; // set proportional gain
double Ki=2.0572; // set integral gain
double Kd=-0.00047687; // set derivative gain
double e;
double el;
double int_e;
double P;
double I;
double D;
double output;
double dt = 0.01;
double dt1;
double dt2;
double w1 = 0.021042611;
double w2 = -0.022570784;
double w3 = 0.025133973;
int fd;
int fd2;
int angle;
int pos = 0;
int counter=0;
double start_time;
struct timespec gettime_now;
double init_time;
uint8_t ch_1 = 0x01; // address for pitot probe reading
uint8_t ch_2 = 0x02; // address for 1st pressure sensor
uint8_t ch_3 = 0x04; // address for 2nd pressure sensor
uint8_t ch_4 = 0x08; // address for 3rd pressure sensor
FILE *fp = NULL;

void main(){ //acquire pressure reading from pitot probe and calculate airspeed
  clock_gettime(CLOCK_REALTIME, &gettime_now);
  init_time = gettime_now.tv_sec+(gettime_now.tv_nsec)/1000000000;
  fp = fopen("data.csv" ,"a");
  fprintf(fp,"\n----------GUST RESPONSE ACTUAL 2----------\n");
  fflush(fp);
  if (wiringPiSetup () == -1) {                    // setup to use Wiring pin
    numbers
    fprintf (stdout, "oops: %s\n", strerror (errno)) ;
    return 1 ;
  }
  softServoSetup (0, 1, 2, 3, 4, 5, 6, 7) ;
  int fd = wiringPiI2CSetup (0x28) ; // 0x28 I2C device address
  int fd2 = wiringPiI2CSetup (0x77) ; // 0x77 I2C device address
  while(1){
    clock_gettime(CLOCK_REALTIME, &gettime_now);
    start_time = gettime_now.tv_nsec;
    counter+=1;
    daq(fd, fd2);
    estimateLift();
    control();
    printdata();
    clock_gettime(CLOCK_REALTIME, &gettime_now);
    if(gettime_now.tv_nsec - start_time>0){
      dt = (gettime_now.tv_nsec - start_time)/1000000000;
    }

  }
}

```

```
104 void daq(int fd, int fd2){ //acquire data and average 50 consecutive readings
      from each sensor
105   while(i<40){
106     wiringPiI2CWrite (fd2, ch_1) ; // select channel
107     read(fd, &data, 4); //request 4 byte data from device
108     pres = (((int)(data[0] & 0x3f)) << 8) | data[1]; // acquiring pressure data
      in number of "counts"
109     p_out = 1.058*pres-8620 ; //conversion and calibration to give pressure in
      kPa
110     airspeed = sqrt(fabs(2*p_out/rho));
111     aas+=airspeed-2.5;
112     i+=1;
113   }
114   aas=aas/40;
115   i=0;
116
117   while(i<40){
118     wiringPiI2CWrite (fd2, ch_2) ; // select channel
119     read(fd, &data2, 4); //request 4 byte data from device
120     pres2 = (((int)(data2[0] & 0x3f)) << 8) | data2[1]; // acquiring pressure
      data
121     p_out2 = 1.098*pres2-8914 ; //conversion and calibration to give pressure in
       Pa
122     ap1+=p_out2;
123     i+=1;
124   }
125   ap1=ap1/40;
126   i=0;
127
128   while(i<40){
129     wiringPiI2CWrite (fd2, ch_3) ; // select channel
130     read(fd, &data3, 4); //request 4 byte data from device
131     pres3 = (((int)(data3[0] & 0x3f)) << 8) | data3[1]; // acquiring pressure
      data
132     p_out3 = 0.9036*pres3-7371 ; //conversion and calibration to give pressure
      in kPa
133     ap2+=p_out3;
134     i+=1;
135   }
136   ap2=ap2/40;
137   i=0;
138
139   while(i<40){
140     wiringPiI2CWrite (fd2, ch_4) ; // select channel
141     read(fd, &data4, 4); //request 4 byte data from device
142     pres4 = (((int)(data4[0] & 0x3f)) << 8) | data4[1]; // acquiring pressure
      data
143     p_out4 = 1.146*pres4-9357 ; //conversion and calibration to give pressure in
       kPa
144     ap3+=p_out4;
145     i+=1;
146   }
147   ap3=ap3/40;
148   i=0;
149 }
150
151 void estimateLift(){ //to estimate lift using acquired pressure data
152   L=(w1*ap1+w2*ap2+w3*ap3);
153 }
154
```

```
155 void control(){ //run PID iteration and actuate
156   e=L-3.2;
157   P = Kp*e;
158   if(output<-1){
159     output=-1;
160   }
161   else if(output>1){
162     output=1;
163   }
164   else{
165     int_e += e;
166   }
167   I = Ki*int_e*dt;
168   D = Kd*(e-el)/dt;
169   output = (P+I+D);
170   el=e;
171   angle = -550*output+550;
172   softServoWrite(0,angle);
173   /*if(counter>=200){ //uncomment for step response
174     softServoWrite(0,0);
175   }
176   else{
177     softServoWrite(0,1100);
178   }*/
179 }
180
181 void printdata(){ //print data to console and file, set all average values to 0
182   //printf("%f ;%d ; %f ; %f ; %f ; %f\n",(double)clock()/CLOCKS_PER_SEC, angle,
        ap1 , ap2 , ap3, L);
183   clock_gettime(CLOCK_REALTIME, &gettime_now);
184   printf("%f ; %f ; %d\n", gettime_now.tv_sec+((double)gettime_now.tv_nsec)
        /1000000000 - init_time, L, angle);
185   clock_gettime(CLOCK_REALTIME, &gettime_now);
186   fprintf(fp,"%f ;%d ; %f ; %f ; %f ; %f\n",gettime_now.tv_sec+((double)
        gettime_now.tv_nsec)/1000000000 - init_time, angle, ap1 , ap2 , ap3, L);
187   fflush(fp);
188   aas=0;
189   ap1=0;
190   ap2=0;
191   ap3=0;
192   i=0;
193 }
```