Word Counts

This section is *not* included in the word count.

11839 words

57564 characters (not including spaces)

```
File: main.tex
Encoding: utf8
Sum count: 11839
Words in text: 11052
Words in headers: 79
Words outside text (captions, etc.): 606
Number of headers: 35
Number of floats/tables/figures: 49
Number of math inlines: 89
Number of math displayed: 13
Subcounts:
  text+headers+captions (#headers/#floats/#inlines/#displayed)
  20+0+0 (0/0/0/0) _top_
  134+1+0 (1/0/24/0) Section: Nomenclature
  842+1+18 (1/3/0/0) Section: Introduction} % The \section*{
  0+1+0 (1/0/0/0) Section: Background} % The \section*{
  491+3+29 (1/1/0/0) Subsection: Water Locomotion methods
  594+2+78 (1/3/3/0) Subsection: Experimental Methods
  430+2+13 (1/1/0/0) Subsection: Model Validation
  377+3+24 (1/2/28/0) Subsection: PID control methods
  0+2+0 (1/0/0/0) Section: Model specification
  750+5+0 (1/0/1/2) Subsection: Design of the Robotic Mechanism
  574+12+54 (7/4/0/0) Subsection: Mechanical Alterations
  0+2+0 (1/0/0/0) Section: Validation Methods
  649+3+21 (1/3/1/4) Subsection: Motor Programming Methods
  821+2+71 (1/6/18/5) Subsection: PID Method
  244+3+12 (1/1/0/0) Subsection: Motion Capture Programming
  571+5+14 (1/2/2/2) Subsection: Validation of the Experimental Set-up
```

```
  45+1+0 (1/0/0/0) Section: Results
  1175+8+58 (4/6/1/0) Subsection: Encoder Verification
  2363+17+191 (6/15/8/0) Subsection: Motion capture Verification
  440+1+0 (1/0/1/0) Section: Conclusions
  532+4+23 (1/2/2/0) Section: Recommendations for Future Work
  0+1+0 (1/0/0/0) Section: Appendix

File: output.bbl
Encoding: utf8
Sum count: 0
Words in text: 0
Words in headers: 0
Words outside text (captions, etc.): 0
Number of headers: 0
Number of floats/tables/figures: 0
Number of math inlines: 0
Number of math displayed: 0
```

Development and Validation of Insect-inspired Flapping Wing Mechanism for Surface Locomotion

Draft 6

Zachary Phipps

20th April 2024

Supervisor - Dr Swathi Krishna

Word count : 10387

Hydrodynamics — Mechatronics — Verification

This report is submitted in partial fulfillment of the requirements for the MEng Aeronautical and Astronautical Engineering, Faculty of Engineering and Physical Sciences, University of Southampton

Declaration

I, Zachary Phipps, declare that this thesis and the work presented in it are my own and have been generated by me as the result of my 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 want to thank my supervisor, Dr. Swathi Krishna, for her continued guidance and support throughout this project. Without her, this project would not have been possible.

Abstract

This report details the methods used to verify and validate a mechanical model of a honeybee's sculling motion seen when landing on water. This scaled-up mechanism of a honeybee wing will be used to find an optimum water-treading motion needed to produce forward thrust on the water's surface for a flat plate wing. This translates to the future of micro unmanned air vehicles, allowing them to traverse the water's surface and sustain flight with one integrated mechanism instead of separate subsystems. Increasing understanding of this motion means it can further be applied to larger vehicles, providing a framework for manned-air-water vehicles.

This report details the mechanical challenges faced when building such a mechanism, the methods used to program the kinematics of the wing, and the impact this has on the output. Additionally, this report details the results of both encoder and motion capture validation methods and the drawbacks of the mechanism that become evident using this. To support further use of this mechanism, this report details improvements that could be made to improve accuracy, as well as a step-by-step guide on building and using the apparatus and code.

The key takeaways from this report are that the mechanism, working best at low speeds, functions to within 1.73% of accuracy in the pitching motion and $\pm 3^{\circ}$in the stroke motion, with an explanation on how this can be improved.

Contents

Nomenclature

1. MAV - Micro unmanned air vehicle
2. UAV - Unmanned air vehicle
3. SCL - Serial Communication Language
4. PID - Proportional, integrative, and deferential control
5. K_i - Integration constant used in PID control
6. K_d - Differentiation constant used in PID control
7. K_p - Proportional constant used in PID control
8. K_u - Ultimate gain constant
9. P_u - Period of oscillation with K_p set to K_u [s]
10. P_e - Expected position of the motor [steps]
11. α - Required Amplitude [°]
12. A - Required amplitude [steps]
13. M_s - Motor steps per revolution [steps]
14. G - Gear ratio
15. t - time [s]
16. f - frequency [Hz]
17. L - lead/lag constant [steps]
18. V_l - Lead velocity, calculated [steps/s]
19. P_a - Actual position given from the encoders [steps]
20. E - Error [steps]
21. P - Proportional control component [steps]
22. I - Integrative control component [steps]
23. D - Differential control component [steps]
24. V_f - feed velocity, sent to motors after PID [steps/s]
25. R - Stroke radius. From the centre of the movement to the edge of the tip of the wing [m]
26. S_f - Safety factor to avoid overloading the stroke motor.

1. Introduction

Many forms of wildlife traverse the water's surface; not many can also fly. In recent years, research has begun into these insects capable of both water and air travel and the methods they use to do so. One insect that has mainly been ignored in this research is the honeybee, with a 2019 study by Roh and Gharib being the main paper focusing specifically on this topic. This paper details the honeybees' hydrofoiling motion and provides qualitative and quantitative data on the force it produces [1].

Figure 1. Honeybee in water [2]

Honeybees use their wings as hydrofoils, benefiting from the surface tension to pull water up under their wings and then push it down and away, creating vortexes under the water and ripples on the water's surface. The imparting of momentum onto the water provides thrust in the forward direction, propelling them towards land. Understanding the honeybee's motion is vital to the future of water-air vehicles as it provides a blueprint for a singular system that can transition between air and water.

UAVs are being used in industry for a wide variety of tasks. Having UAVs that can traverse the water's surface is a cost-effective way of researching and characterising bodies of water [3]. This could include managing and sampling oil spills[4], water quality control[5], and pollution sampling[6]. Used in places where in-situ measurements would be difficult or economically impractical, they provide an opportunity to collect data where otherwise it could not. Using UAVs also allows for the implementation of swarm techniques to increase the sampling rate and coverage area[7]. Landing and collecting samples and being able to take off again makes for a more efficient system that can move to different water bodies without human intervention; this betters its watercraft alternatives that rely on propellers or other means of propulsion[8].

There is less research on the honeybee water locomotion method. This lack of knowledge may be due to honeybees not typically seen as water creatures, and the speed and efficiency at which they move is lower than that of their water-skating counterparts. However, they provide one significant benefit that other insects do not see. They use their wings to propel themselves instead of water-walking or water-skating, which requires hydrophobic legs. Understanding this technique and the conditions in which it becomes efficient is critical to applying this process to micro-unmanned air vehicles and other ornithopters for air-to-water transition. Without this technique, MAVs would need separate mechanisms for flying (wings) and water-skating (usually legs or skis). Reducing this to one motion reduces weight and complexity, improving MAVs' efficacy, efficiency, and range.

This report details the methods used to produce and validate sinusoidal movement on the pre-formed apparatus.

This project aims to produce and verify the sinusoidal movement seen by honeybees in the water using the mechanical model with a flat wing to make pitch and stroke motions. The aim is that the error from this system is below 2% for the pitch and stroke motion, similar to the model produced by Krishna et al., 2018 [9].

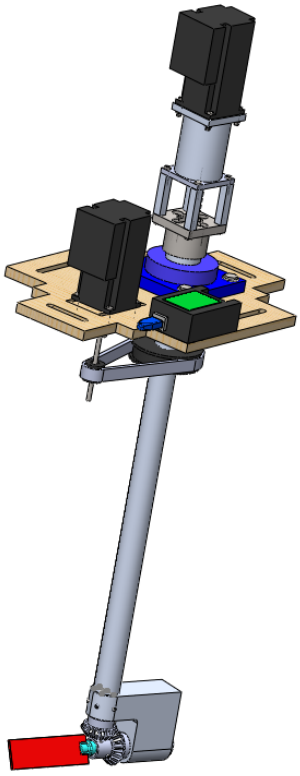


Figure 2. CAD Model of the mechanical model

This objective will be achieved by refining the mechanical systems used in this model to allow the mechanism to run as designed. Then, a MATLAB algorithm will be produced to reproduce the sinusoidal motion, using PID control methods to improve accuracy. This motion will then be run at various amplitudes and frequencies, and video recordings and readings from the onboard encoders will be made. The videos must be analysed using motion capture techniques to verify each oscillation's amplitude, frequency, and shape in stroke and pitch movements. This information will be used to conclude the model's accuracy and any further modifications that should be completed.

This report is structured into five main sections: Background, Model Specification, Validation Methods,

Results, and Conclusion and Future Work. The background portion of this report explains the motivation for this project using past literature to highlight the need for this research and the opportunities it brings to future MAVs. The model specification section details the components of the model and techniques used to improve the mechanism. Validation methods explains the motion programs tested for the mechanism and the programs used for motion capture and provide the setup for the experiments used to validate the model. The results section discusses the results of PID tuning and the final experiments run using this setup. It also explains where errors have been introduced and negated within the system. Finally, the conclusion gives an overall overview of the project, the main takeaways from the project, and future recommendations for further testing and analysis that can be done using this mechanism.

are these not the same?

For the motion to be sinusoidal, it must be correct in three areas: pitch, frequency, and shape, as seen in figure 3. These three areas define the required waveform, confirming all three means 'sinusoidal' can be correctly used to explain the mechanism's motion. The shape refers to the oscillatory pattern of the motion; for this report, a sinusoidal shape is required. As there are infinite numbers of oscillatory waveforms with the required amplitude and frequency, the shape must also be validated alongside frequency and amplitude.

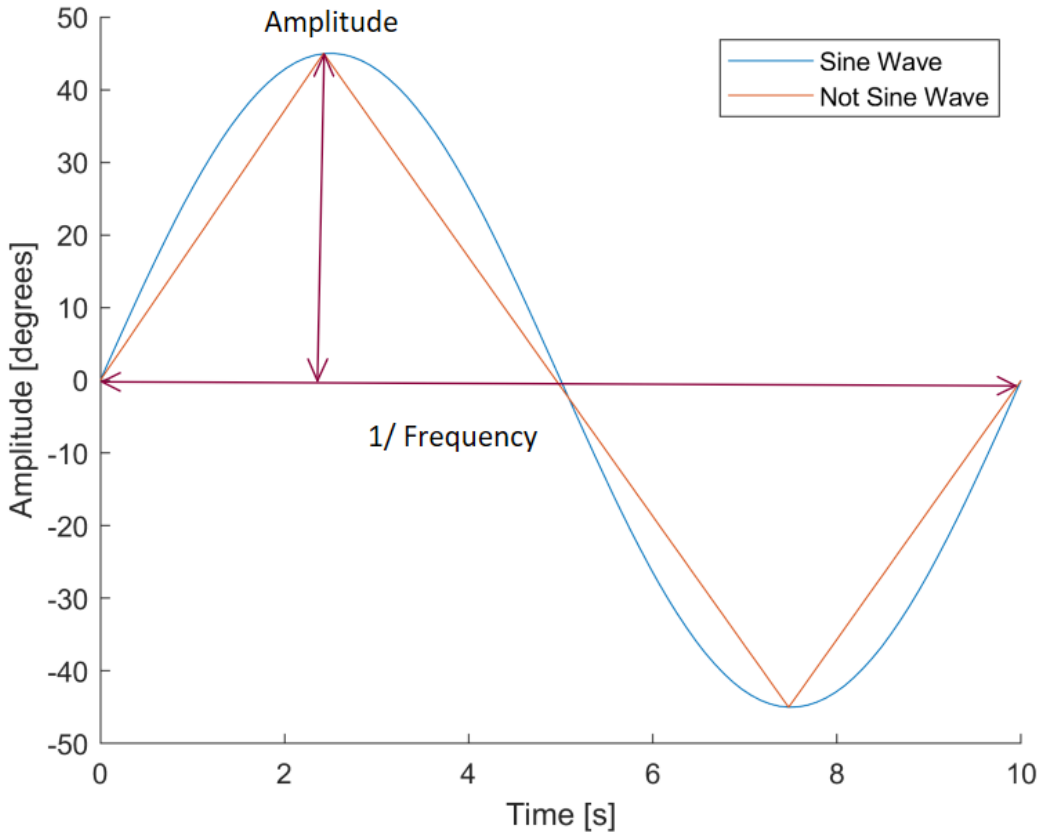


Figure 3. Figure showing the qualities this report is validating for

2. Background

2.1 Water Locomotion methods

Water surface locomotion has been observed and studied in multiple species [10]. Reptiles, such as the Basilisk Lizard, are known for traversing the water surface using water slapping [11]. Insects such as water spiders, water lily beetles, and water striders can all walk on water as their legs are covered in thousands of hydrophobic hairs [12, 13, 14, 10]. Stoneflies and Mayflies row across the surface, using their legs to maintain contact with the water and beating their wings to produce forward momentum [15]. Each of these techniques is faster and more efficient than that seen by the honeybee. However, they all require legs, so copying these mechanisms adds another subsystem to an already complicated MAV. Using the honeybee's method of locomotion streamlines the process, providing a more elegant solution to air-water travel.

This report focuses on the honeybee's method of water locomotion. Once a honeybee lands on water, its wings are captured by the adhesive effect of the water's surface. As a result, they cannot regain flight as this surface force is too great to overcome [1]. Therefore, to preserve life, they use their wings as hydrofoils to propel themselves towards land. This differs from a typical hovering motion and is called the water-treading mode [16].

direction of motion?

is the water in and out of the page?

perchance another diagram to clarify, idk if you can put a gif in a pdf

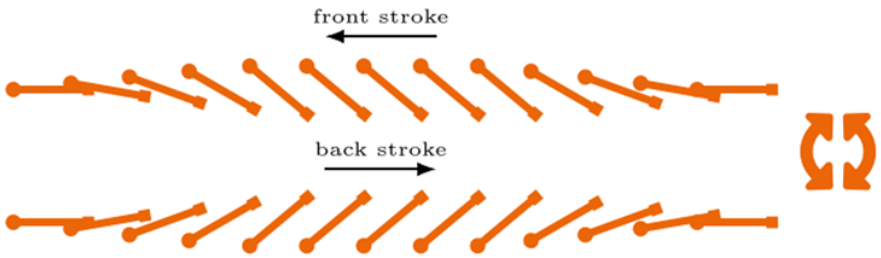


Figure 4. Front and backstroke in the water treading mode where the orange lines represent the wing's chord through one oscillation. Circles and squares represent the leading and trailing edges, respectively. [17].

The water-treading motion differs from previous rowing, sculling, or slapping methods, as the water is pulled up and under the wing instead of pushing down on the surface. When the front leading edge of the wing moves up and back, water moves up and under the wing above the water's surface. The leading edge pitches down, and the stroke reverses direction, causing water under the wing to be pushed down and away. This imparts momentum onto the water [1], propelling the honeybee forward.

This motion causes an overall increase in mean lift compared to conventional hovering techniques [17]. The reasoning behind the improvement in the lift has been studied by Issac, Rowles, and Colozza, first in 2006 [18] and then later in 2008 [19]. As the leading edge pushes down, the leading edge and trailing edge switch roll.

On the trailing edge, a vortex has formed; when the wing switches leading and trailing edges, this vortex moves from the trailing to the leading edge. This causes an increase in lift at the beginning of the oscillation, as a new vortex doesn't need to be formed; the lift is higher for longer, so the mean lift increases.

This pattern of pivoting and pitching a fully submerged wing has been studied in detail. Flow visualisationn studies by Freymuth [20, 16] have been used to vet data for computational models of this water-treading mode [21]. Building on this model, force data was produced for the water-treading mode when pivoting around the midpoint, providing force data for different points on the sinusoidal motion[22]. Previous research on this water-treading mode has been done at a reasonable depth to avoid surface effects impacting results. This differs from the goal of this mechanism as the focus is on how the surface interacts with this form of motion.

2.2 Experimental Methods

Building a robot that can recreate the surface movements of insects has already been achieved on many occasions for a range of surface traversing methods, e.g. jumping, rowing, and slapping [23, 5, 24, 25, 26, 27]. These robots are designed using previous mechanical and CFD models as testing on live insects is, understandably, quite tricky and evokes ethical questions [28]. A robotic model of the honeybee locomotion method specifically for water-surface use has not yet been created, and this report is a stepping stone towards this goal.

capitalisation

Mechanical models of insect wings have produced data using force-torque sensors [17, 18, 19], particle image velocimetry, and computational fluid dynamics. Force-torque sensors, like those to be used in the 2018 paper by Gehrke, Guyon-Crozier, and Mulleners [29] and can be seen in Figure 5, have been used to track force data over motion profiles. Using the highly accurate nano-17 force torque sensor, they can produce force data with a resolution of 0.318 gram-force [30]. This sensor is the same sensor that this report's design has incorporated. With a maximum force reading of 70N they can easily be overloaded with unexpected movements [30]. This means the motion of this model must be smooth, accurate, and reliable.

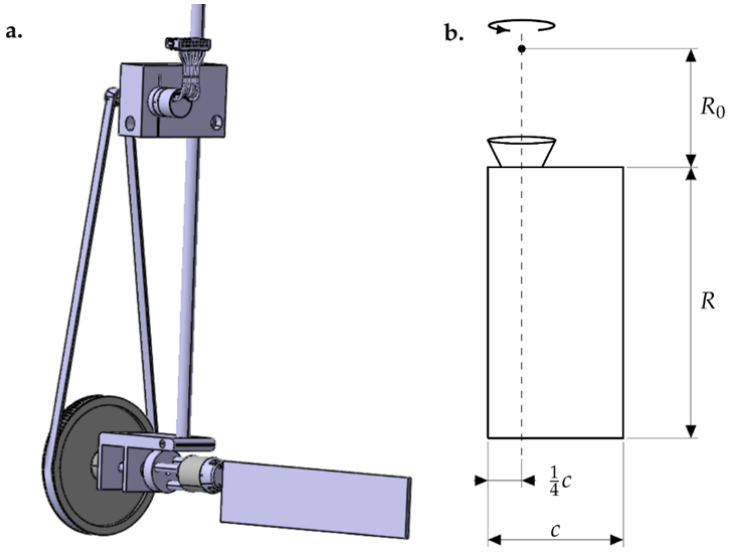


Figure 5. Model used by Gehrke, Guyon-Crozier, and Mulleners a) Model to produce pitch and stroke motion using a belt for pitch and gear for stroke
b) A plan view of the model wing [29]

The Gehrke, Guyon-Crozier, and Mulleners setup was used to find a typical hovering motion's maximum lift and efficiency and was tested within an octagonal tank. The rotational axis on their flat plate in this report is at the 1/4 chord point. This differs from other models, such as those by Krishna et al., 2022. whose pivot point is at the 1/2 chord point. This is likely due to the differences in hover mechanisms the models try to reproduce. Additionally, in the paper by Gehrke, Guyon-Crozier, and Mulleners, no detailed effort was made to reduce the impact the tank walls would have on reflected waves interfering with the force sensor, and no detailed adjustments were made to account for inertia within the calculations. Alternatively, Isaac, Rolwes, and Colozza [19] discuss the errors that could have occurred using their vertical wing set-up.

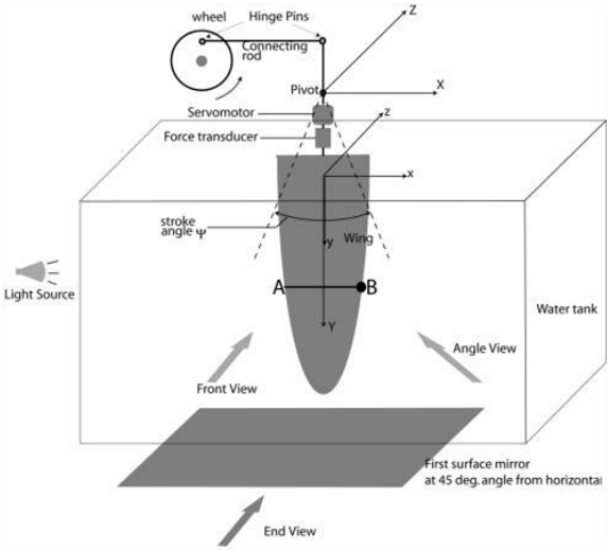


Figure 6. Diagram of the model used by Isaac, Rolwes, and Colozza[18]

The weight, inertia, and imperfections in apparatus fabrication were considered when formulating force data, and the error this caused on force data was estimated at $\pm 10\%$. Additionally, the experimentation results were qualitatively compared to those of previous research done by Sunada et al. [22] . This was done to ensure the data collected matched those from different experimental setups. However, this setup is also vertical, increasing the impact weight had on the movement as performing a stroke motion meant that the motor also had to lift this wing in the water. This differs from the design used in this report, as it places the wing horizontally, so this should have less of an effect.

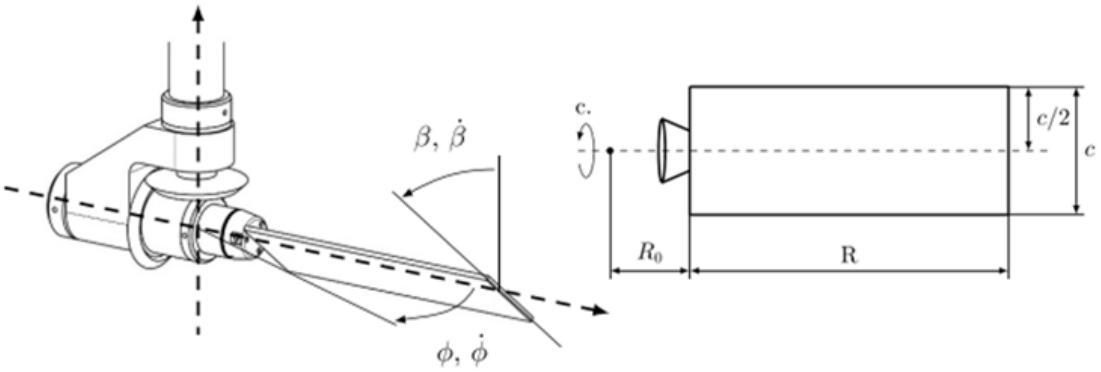


Figure 7. Diagram of the model used in Krishna et al. 2022 for calculating force data of different hovering types. Where β is the pitch angle, ϕ is the stroke angle, C is the chord, and R is the span

A similar model to that used in this report is used in both Krishna et al. 2018 [9]. However, this has two axles and uses worm gears to create stroke and rotational motion. This setup could also flap along the x-axis,

define the coordinate system used

but this wasn't explored during this study. The error of the mechanical apparatus was estimated to be less than 2% in this experiment. Therefore, for this report, it can be expected that the error for both the pitch motion and stroke motion combined should be around 4%

In the Krishna et al. 2022 paper, a different mechanical model was used [17], as seen in Figure 7. This more closely matches the mechanical model used in this report. This provided two axes of rotation, pitch, and stroke, and the authors deemed this design highly accurate. Therefore, it can be expected that a similar model, as used in this report, should have a comparable level of accuracy.

2.3 Model Validation

idk what a contact angle is

The mechanical model's movement is expected to follow a sinusoidal motion in both pitch and stroke [1, 17, 31]. This is to mimic the honeybee locomotion. The model is expected to move through an angle that matches the angles seen in honey bees. Roh and Ghrarib stated that the contact angle of the wing was between 85° and 103°. This gives 20° of movement in the pitch angle; this paper also shows that the stroke angle in the water is less than 10°. Therefore, the model will need to be able to produce these angles and more to provide an accurate determination of the motion. This report will also focus on higher angles of attack to allow for other motion patterns to be tested with the mechanism when optimising for a mechanical model. Wing beating frequency while in the water is lower than in flight at frequencies ranging between 40 Hz and 120 Hz [1]; this will be reduced significantly when scaled for the mechanical model.

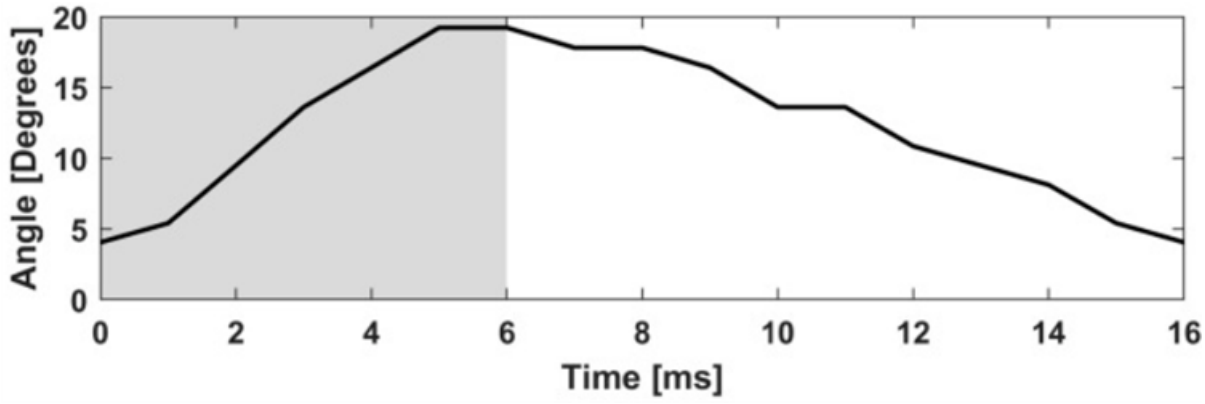


Figure 8. Pitch angle in the mechanical wing through one oscillation by Roh and Gharib [1]

The mechanical wing is being validated to produce force data on the water's surface. As the Nano-17 force-torque sensor is highly accurate, reflecting waves off the tank's walls will interfere with the results. Ensuring that the mechanism is centred in the tank and waves moving parallel with the wing's chord head down the tank length-ways is one way to mitigate this; the further the wing is from the walls, the less impact they have [32]. Another way this has previously been mitigated is by only running the experiment for a few oscillations

so that the reflected waves don't have time to interfere with the force sensor [17]. This would reduce the data sets and make it difficult to compare oscillations. Another way this can be mitigated is by damping the waves. Placing Styrofoam on top of the water and the tank's walls should reduce the waves' amplitude [33]. However, this method primarily mitigates free surface waves when measuring just below the surface. Therefore, it might not be a helpful method in this report; another method might have to be sought for future force-data experimentation.

Motion capture has been used to prove the movements of wings [34]. This report was used to verify the mechanism's pitch, stroke, and frequency. Seshadri, Benedict, and Chopra have previously done this in a 2012 study, as well as many others. This paper studied the 3-dimensional movement of a flapping wing. As a result, they measured three angles, pitch, stroke, and flapping angles, and used this to verify the figure of eight motions of their wing. Whilst the wing's motion differs, the method remains similar to this report[35].

2.4 PID control methods

PID control methods use a closed-loop control cycle to reduce errors within a system. Proportional, integrative, and derivative control constants can be used depending on the system's needs to force the system to behave in a desired way. Control is done in a closed loop, meaning the output of the loop will impact the input of the next iteration. The error between the iterations is the difference between where the object is and where one expects the object to be within one time-step. [36].

i think the bit about what a PID controller is can be cut down to like a sentece because its pretty well known

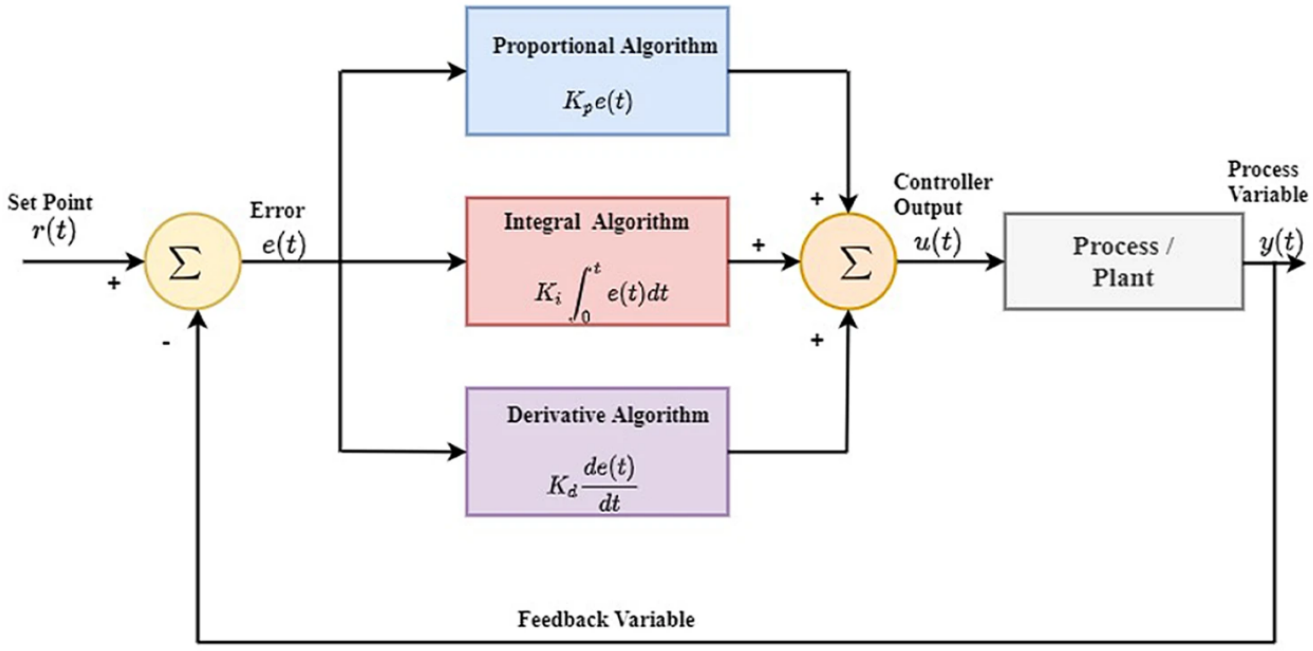
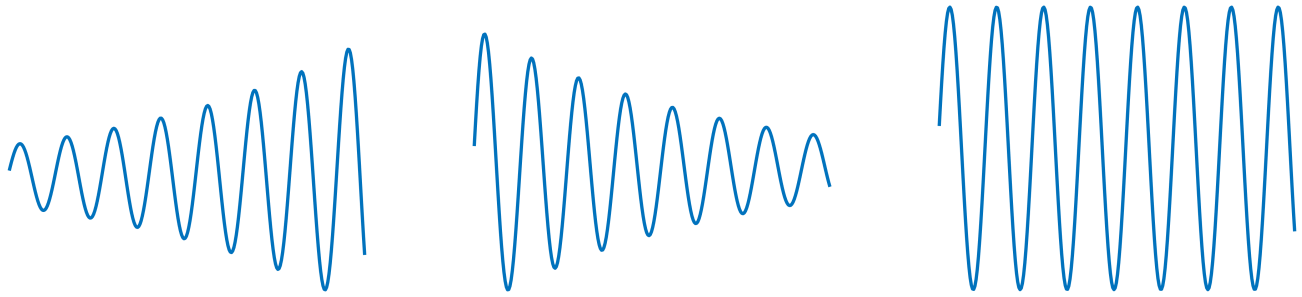


Figure 9. Closed loop PID controller [36]

The control loop is tuned using the terms K_p, K_i, and K_d; these must be correct for the system it controls or else the error can be magnified. Many methods are used to tune systems. The trial and error method can give an overall idea of where the expected PID values are but are hard to get entirely right. The trial and error method sets K_i and K_d to low values and increases K_p until the system responds desirably, using K_i to fine-tune steady-state error and K_d to help reduce overshoot and unwanted oscillation within the system [36].



(a) K_u is too low **(b)** K_u is too high **(c)** K_u is correct

Figure 10. Ziegler-Nichols Method control loop reacting to a perturbation

Another method used to tune closed-loop control cycles is the Ziegler-Nichols Method. This uses an ultimate gain value K_u that the proportional gain, K_p, is set to. K_u is then raised until the system steadily oscillates around the set point it tries to achieve. Plotting the error from the set point should give a curve much like Figure 10c. The period of the oscillation seen in 10c is recorded as P_u. From this, the terms K_p, K_i and K_d can be calculated for P, PI and PID control methods using table 1 [37].

This method is used for systems where characteristics cannot be easily mathematically derived and instead uses experimentation to calculate the expected values. However, the tester would have to be careful not to set K_u too high as this would create an unsteady system, and the oscillations created from error would increase exponentially, ultimately damaging the system being tested[37].

Table 1. The Ziegler-Nichols Method Settings [38]

	K_p	K_i	K_d
P controller	$0.5K_u$	0	0
PI controller	$0.45K_u$	$0.833/P_u$	0
PID controller	$0.65K_u$	$0.5P_u$	$0.125K_u$

The motor's onboard encoders must be used to complete PID on this setup. These provide feedback from the motor, including power consumption, current, voltage, velocity, position, and temperature [39]. Therefore, they can be used to create a PID system and ensure the motor remains safe and isn't overloaded during tuning.

i think encoders only provide position and it would be different sensors that give the other results

3. Model specification

3.1 Design of the Robotic Mechanism

The mechanical model is expected to produce smooth sinusoidal movement profiles in both pitch and stroke while running in a recirculating water tank at 3m/s. It is expected to be accurate from the first oscillation to study how the vortices form around the wing. In addition, it must remain consistent and accurate across 20 oscillations, with an error rate of less than 5

unfinished

This base design was completed in a previous independent project. However, in the initial assessment phase of this project, it was clear some aspects of the model needed to be revised as the wing was incapable of consistent movement. This section details the model, its subsystems, and components, then explains the elements of the model that were altered.

Number	Componant
1,2	STM23S-3RE Motor running at 24v
3	1-6 Planetary gearbox
4	12mm Plywood Baseboard
5	Blue F-206 Flange Bearing
6	USB to RS485/422 Converter
7	48 Tooth Steel Timing Pulley
8	15mm wide Timing Pulley
9	Secondary Axle
10	12 Tooth Steel Timing Pulley
11	Main Axle
12	Mechanism block
13	Bevel gear set
14	Force torque senor
15	3D Printed Wing adapter
16	3mm acrylic flat plate wing

Table 2. A List of main componants

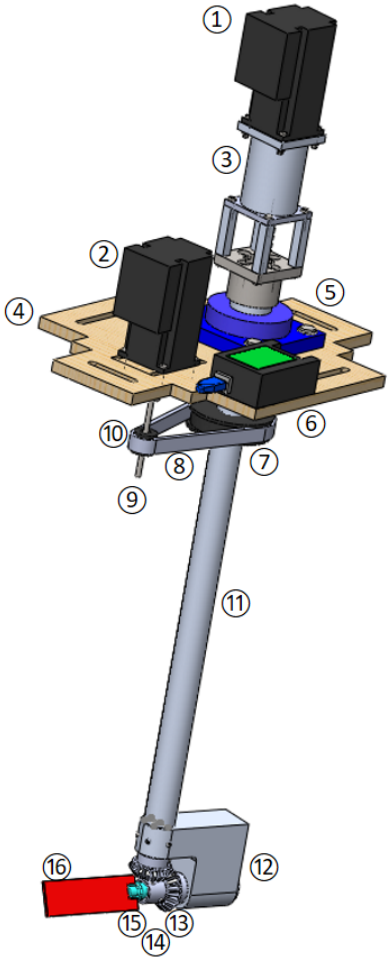


Figure 11. A labelled version of the mechanical models's CAD model

confusing

This setup ran two STM23S-3RE motors, one for controlling pitch and the other for controlling stroke. The

motor has a resolution of 1.8°per step; however, a resolution of 0.018°can be achieved using micro-stepping features on the motor.

A USB to RS485/422 Industrial Grade Isolated Converter was used to communicate with the motors. The USB was connected to the controlling computer running the appropriate MATLAB software. A variable power supply was used and set to 24v and 8A, as guided by the motor specifications [40].

The stroke motion was controlled using a motor (Motor 2) attached to a wooden baseboard. A rigid coupler attached the motor's shaft to the secondary axle, which had a 12-tooth pulley mounted to it. This pulley provided torque to a 15mm wide rubber belt connecting to the central shaft with a 48-tooth pulley.

The stroke motion must execute accurate movements in flowing water, so the torque required will be higher than in still water. This can be calculated using equation 1 to find the equivalent flow speed when the wing is also in motion. Using a recirculating water tank at 0.3m/s (V_{inf}), a maximum frequency of 0.4Hz, a safety factor of 2.5, and a stroke radius of 0.14m the force experienced on the wing using equation 2

use \infty for infinity symbol
caplitalisation of Equation in the text

$$U_{inf} = V_{inf} + 2pifRS_f \tag{1}$$

inline this

$$F = \frac{1}{2} C_d \rho A_r U_{inf}^2 \tag{2}$$

Translating this force into torque using T =F*R gives a necessary torque of 4.228Nm. The STM23S-3RE motors are capable of producing a torque of 1.41Nm. A gear ratio of 1:4, from the pully system allows the motor to produce more than the needed torque for this project. This means the motor can be accurate to 0.0045°; however, the pulley system is the least accurate component in the stroke motion, with initial experimentation showing a reduced accuracy of 0.5°when completing point-to-point turning.

The pitch motion was controlled by motor one, connected to a 1-6 planetary gearbox. This increased the number of steps per rotation by a multitude of six, improving accuracy and torque. This motor sat atop the moving shaft, coupling the movement of the motor to the main axle. The main axle contains an interior axle to ensure that the stroke motion cannot influence the pitching motion. Held in place using internal bearings, it allows the internal axle to rotate freely without the influence of the main axle. The interior axle was connected to bevel gears, translating the rotation from the z-axis to the x-axis. This bevel gear set had a ratio of 1:1.5, meaning that the overall ratio of the pitching motion was 1:9. Whilst this gear ratio means that motor 1 could be accurate to 0.002°, the bevel gears become the limiting factor with an accuracy of 0.2°.

The horizontal bevel gear was connected to a shaft that ran through the mechanism block to the force-torque sensor. From here, the 3D-printed wing adapter was screwed into a dummy force-torque sensor, and to this, the wing was attached.

The wing chosen in the model matches the wing from [17] Krishna et al. l 20022 paper, with a wing chord

airfoil profile?

length of 34mm, a thickness of 3mm and a span of 107mm.

The procedures for setting up and using this mechanism, as well as a more in-depth parts list, are included in the appendix.

3.2 Mechanical Alterations

3.2.1 Bevel Gears

Due to misalignment in bevel gears and related axles, the gears would lock up at different points in the cycle. This was caused by manufacturing tolerances being set too low for previously machined components in the mechanism block. To overcome this, waterproof tape was placed in all the affected joints to increase the diameter of the inner part. As shown in Figure 6 right, this allows the parts to sit concentrically and move as initially designed.

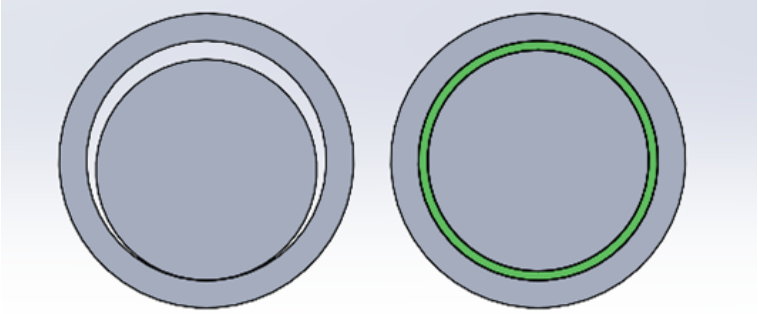


Figure 12. Left: Before improvements were made Right: Tape placed

This method has been used on the connection from the main axle to the mechanism block, resulting in an increase of 0.6mm in diameter on the main axle. Additionally, it is seen in the wing axle, where an increase of 0.75mm around the wing axle connector, 0.6 mm underneath the bevel gear, and 0.5mm for the rest of the axle was achieved. This resulted in steady pitch motion without the gears locking up.

3.2.2 Stroke mechanism

The stroke mechanism uses 2 timing pulleys and a timing belt to convey torque from the secondary axle to the main axle. This section was originally fitted with a flexible coupling to reduce the impact of vibrations on the system. However, for the timing belt to be accurate, it must be pulled taut. This is not possible if one end is supported by a flexible coupling. To remedy this, a new 12-tooth timing pulley was ordered, as the old one was super-glued to the previous axle, and a rigid coupler was sourced from the EDMC. This meant that the belt could be pulled taut, allowing for the designed accuracy of 0.5°.

3.2.3 Main Axle

One significant issue was the main axle's unrestricted tilting movement. It was supported by one point of contact, an F-206 bearing, allowing it to tilt during motion. This movement would have led to significant errors

in verification testing and force data. To address this, an identical F-206 bearing was sourced from Simply Bearing Ltd and attached to the underside of the plate, providing a secondary point of contact and effectively restraining movement.

3.2.4 Wiring

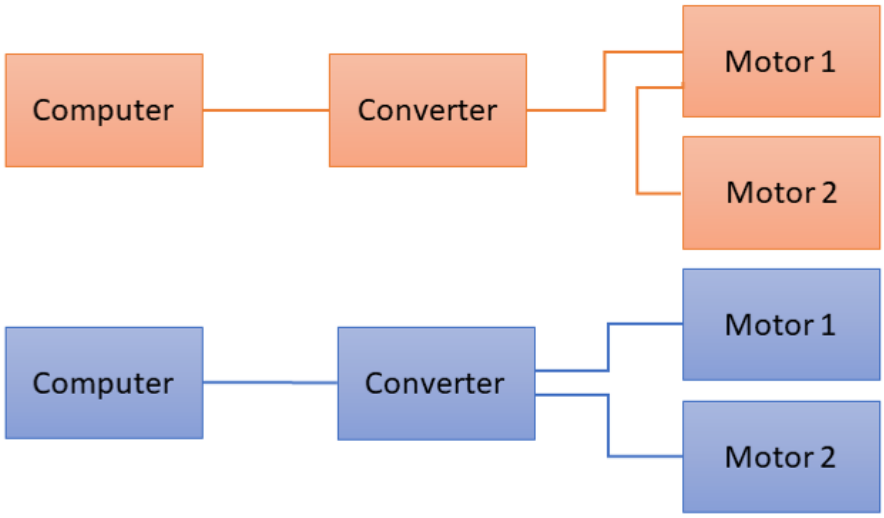


Figure 13. Red: previous wiring methods. Blue: current wiring

The previous wiring for this project used daisy-chained single-core wires for communication between the motors and the USB to the RS422 converter. This meant the controls for motor 1 passed through the connector for the controls for motor 2. This increased noise in the system, and the wires habitually disconnected during operation. To amend this, motors were reassigned, and the wires were replaced with non-daisy-chained, longer multi-core alternatives, reducing the connection issues seen during operation.

capitalisation inconsisten with rest of report

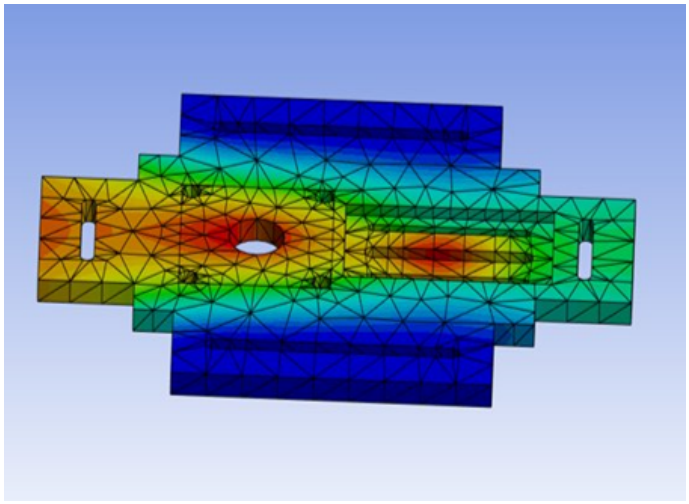
3.2.5 Base-plate

hmmm?

The original base-plate could not support the set-up's weight. It boasted a 2-thickness design to allow Motor 2 to reach its coupling. However, having a 6mm plate glued into a 12mm plate with no additional support created a weak point in the system. The plate deformed under the weight and was beginning to snap; in replacing the plate, a new design was made with a singular thickness, and the supporting areas were modified to reduce stress.

im so sorry

This was then analysed using the ANSYS workbench to confirm a decrease in stress. The new plate was laser-cut out of 12mm birch Plywood in the EDMC, and parts were transposed onto the new board with no further issues.



maybe just show the bar on the side?

Figure 14. Prevous base plate. Red shows high levels of deformation, and blue shows low levels.

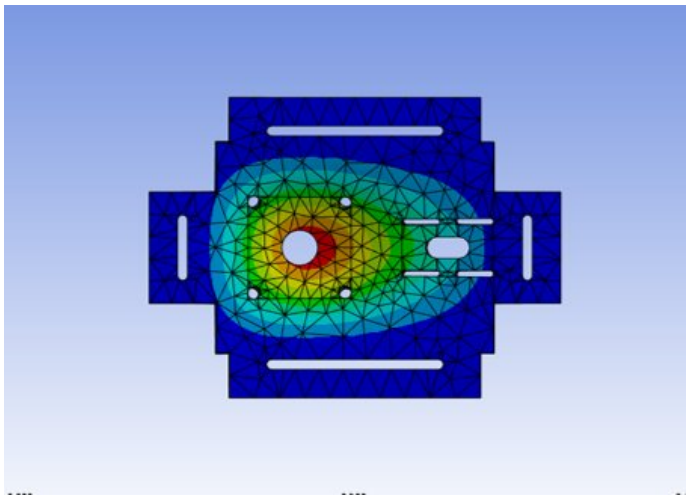


Figure 15. New base plate. Red shows high levels of deformation, and blue shows low levels.

3.2.6 Wing Adapter

A new 3D-printed wing-to-force-torque sensor adapter was also designed and implemented. The original could not support the force felt by the wing moving through still water; it would not have been compatible with recirculating water testing. The new design increased strength while remaining relatively streamlined to reduce the impact on future experimentation.

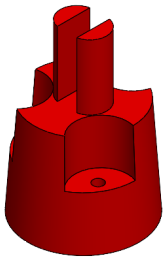


Figure 16. Previous wing adapter design.

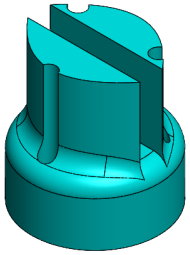


Figure 17. New wing adapter design with increased support.

4. Validation Methods

4.1 Motor Programming Methods

The STM23S-3RE motors can be controlled directly using the ST configurator. This allows the user to change settings, confirm communications, and run the motors. However, the motors must be placed in SCL (Serial Command Language) communication mode to allow for control using MATLAB commands.

Using SCL commands [39], there are multiple ways to produce a sinusoidal motion: time-position control, position-velocity control, and time-velocity control. Each method was tested with the mechanism moving in the air, as the initial runs were done to test the viability of the various methods.

$$A = \frac{\alpha M_s G}{360} \tag{3}$$

big spacing

weirdass cos

$$P_e = Acos(2\pi t_1 f - L) \tag{4}$$

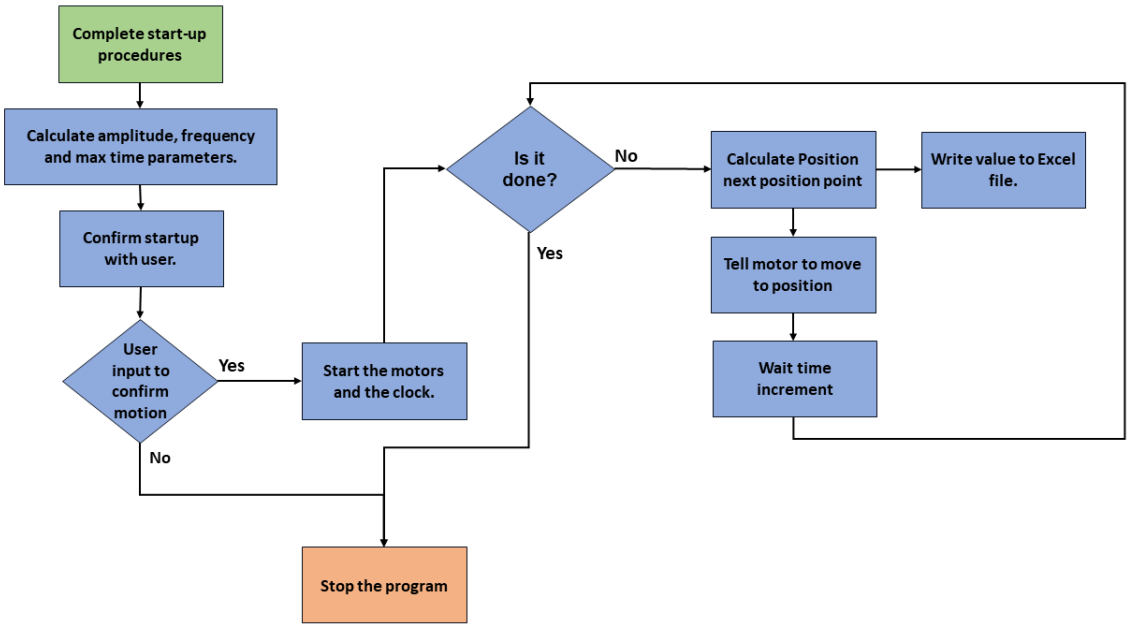


Figure 18. Flowchart showing the position time control method

The first method attempted was position-time control. As shown in figure 19, this produced a position points a set increment in time apart and then commanded the motor to go through this position. The position P_e with respect to time could be calculated using equation **??** where the wave's amplitude was calculated using 3. This used the feed to position command 'FP' and the control computer's internal clock to output the commands at the right point in the cycle.

This produced highly accurate positioning, with an average of +- 10 steps or 0.02 degrees error at the motor level. However, when increasing the frequency, the motion lost it's sinusoidal-like nature. This was because the motors have internal memory, and the commands were stored and done in order. So, whilst the timing could be set on the computer, if the commands were sent before the previous command had finished executing, they would sit in memory and be executed regardless of the time. The motors would also stop between each point, leading to a jittery motion that couldn't maintain the necessary frequency. Therefore, whilst point-to-point positioning was effective for exact movements and checking the oscillatory amplitudes of the apparatus, it could not be used to complete a smooth sinusoidal motion.

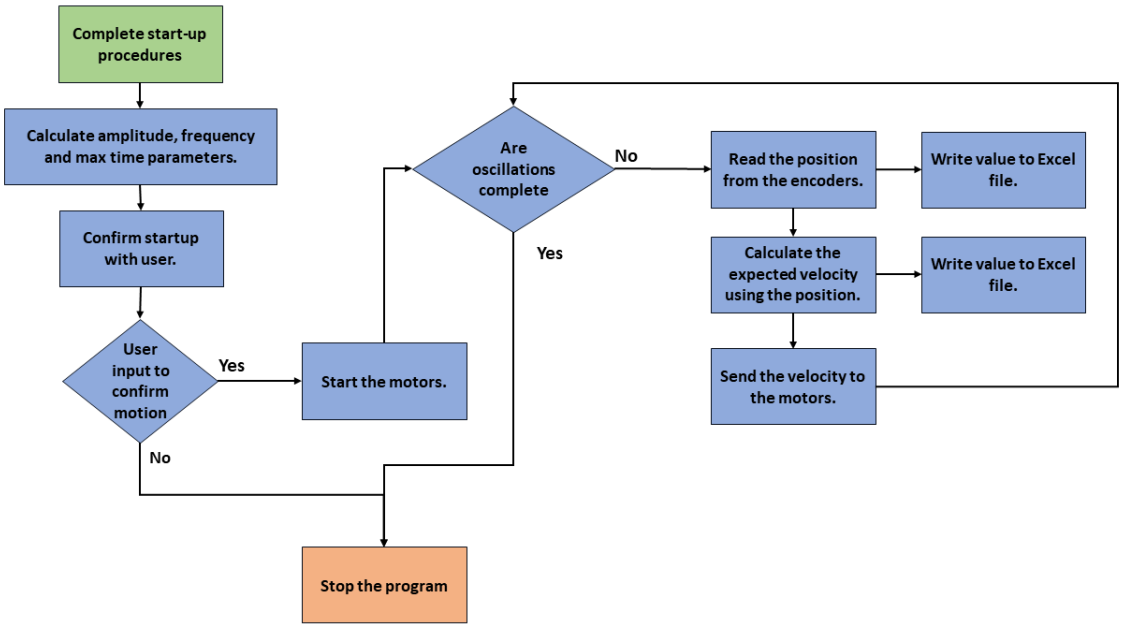


Figure 19. Flowchart showing the position velocity control method

The next attempted method used the jogging command set within the SCL command reference. This sets the motors to run at a constant speed of 0 steps/s, which can be instantaneously changed with the change speed command 'CS'.

$$V_l = \frac{3\pi f cos(sin^{-1}(P_a))}{2A} \tag{5}$$

Instead of using an internal clock, this relied on finding the motor's instantaneous position using the command 'IP'. This was then used to calculate the speed the motor should be going at using equation 5.

This meant that commands bypassed the onboard memory on the motors and would be immediately executed, improving the frequency issues. This method was, however, flawed; the position was a sin wave, and the velocity wave was a cosine wave; translating between the two gave two possible speeds for the same amplitude. This meant that a switchback in the code had to be created to switch between the fore and backward motion. As the motor was still moving as the commands were being read, this inconsistency led to irregular and unexpected movements at the motion peaks, so was disregarded as a motor control method.

The method that produced executable results was using velocity-time control.

$$V_l = 4A\pi f \frac{sin(2\pi t_1 f - L)}{M_s} \tag{6}$$

This while loop read time from the computer's internal clock, calculated the lead velocity using equation 6, and immediately executed it in the motor using the jogging command set.

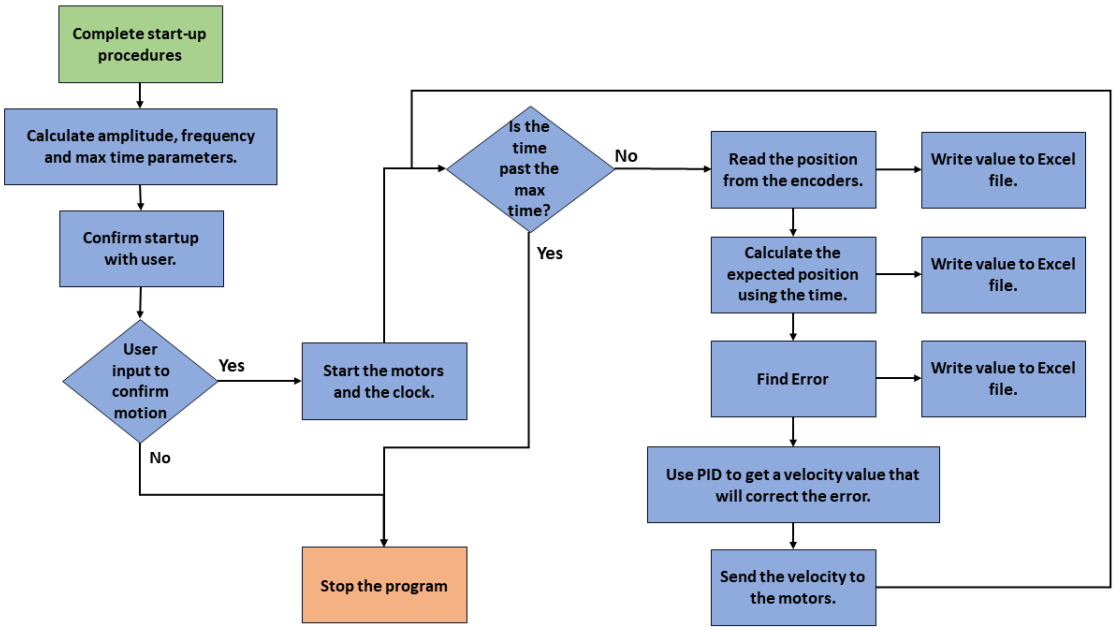


Figure 20. Flowchart showing the velocity time control method

On first attempts, this did create a sinusoidal motion. However, at higher frequencies or amplitudes, the sinusoidal wave began to slip and would cause each oscillation to move slightly to the left. After 20 oscillations, the pitch or stroke would be entirely out of the phase where it began. Fixing this was best done by adding PID control to this programming method.

Using time as the value that the velocity was based on meant that integrating lead and lag into the program was relatively trivial, as it meant adding a constant into the initial section that shifts the expected velocity back or forward to the lead or lag position, and then adjusting the start position to ensure that the wing is in the correct orientation for that lead or lag. This became an option for users to adjust during set-up without altering the code.

4.2 PID Method

When communicating with the motors in SCL commands, there is a delay between the message being sent, received, and executed. This delay, while around 3ms per command, impacts timed commands.

Adding PID control can help reduce the effect of the delay in the wings' waveform. Including this in the system increases the delay as it needs information from the motors to calculate the error. However, as PID is intended to reduce the effects of the initial delay, introducing a secondary delay and correcting it is trivial.

The expected position was calculated using equation 4. Then, the error could be calculated using the immediate position given by the SCL Command 'IP'.

$$E = P_e - P_a \tag{7}$$

The data from the motors arrived in discrete packages, meaning a discrete method of calculating PID was chosen over the continuous versions of the equations seen in 9. Equations 8, 9, and 10 were used to calculate the different components of the PID algorithm, and the feed velocity being sent to the motors was calculated using equation 11

$$P = K_p E \tag{8}$$

$$I = k_i \frac{(t_1 - t_2)(E_1 + E_2)}{2} \tag{9}$$

$$D = k_d \frac{E_1 - E_2}{t_1 - t_2} \tag{10}$$

$$V_f = V_l + \frac{P + I + D}{M_s(t_1 - t_2)} \tag{11}$$

This calculated the error and then used PID control methods to increase or decrease the velocity to match the motor's position in the cycle to its expected position.

Initially, it would seem using velocity for the error cycle would remove the extra calculation seen in equation 11 to translate the PID components from position to velocity. However, the STM23S-RE motors have an onboard motor driver that controls velocity [40]. This means that even when commanding the motors to move at a constant velocity, reading the immediate velocity of the motors gives a highly noisy velocity curve. This is because the motor makes corrections to average at the correct velocity. As a result, using this for PID would ultimately mean that the PID algorithm would contend with another unknown control sequence and try to correct it. This would make the system less accurate and introduce noise into the calculations.

As seen in figure 21 and 22, the slip in amplitude becomes evident between the cycles without PID in place. Additionally, as seen in 21, the expected and encoder positions for the second and subsequent oscillations almost align on the downward stroke. However, on the upward stroke, there is a gap. This means that the system's movement isn't consistent and suggests that the frequency varies throughout. Within four oscillations, the maximum pitch amplitude decreases by 232 steps (4.64°), and the minimum amplitude decreases by 329 steps (6.58°). This significant error requires PID algorithms to be set up to remove this movement.

needs a legend, label axes

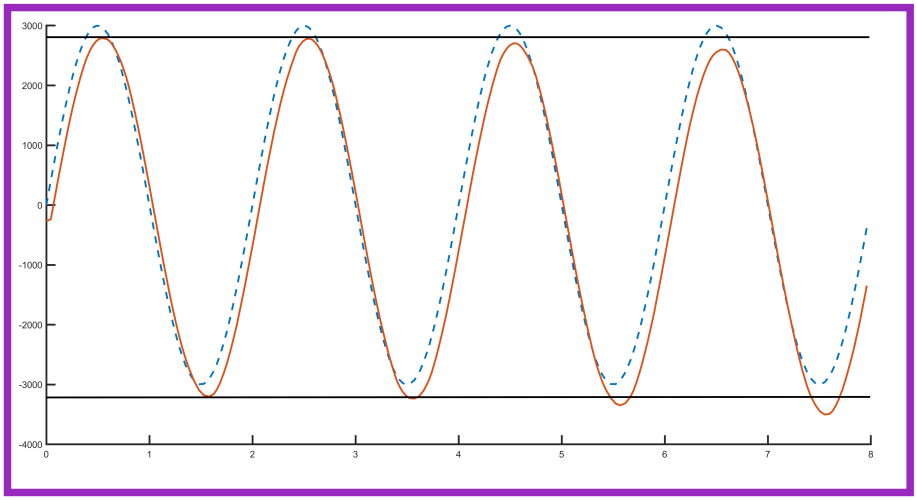


Figure 21. Pitch motion 1 without using PID control methods. The dashed line shows the motor's expected placement. The two black lines represent the first oscillation's amplitude.

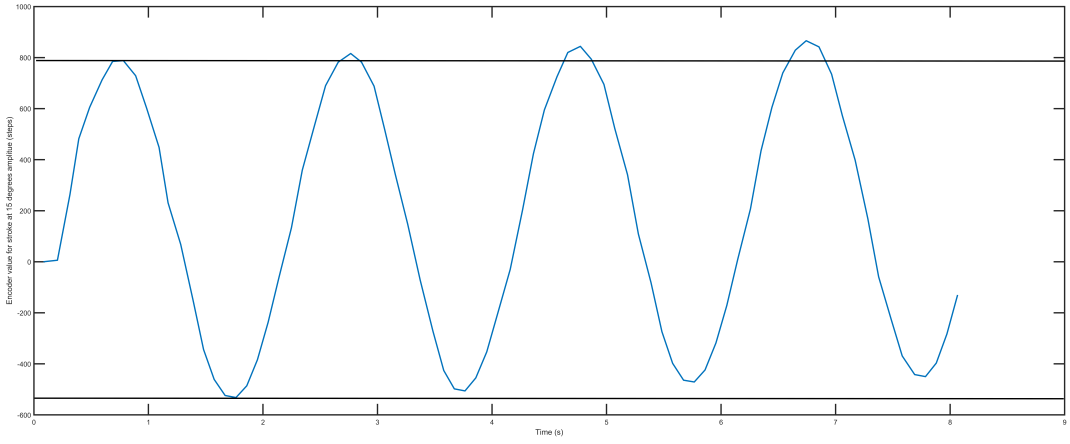


Figure 22. Stroke motion 13 without the use of PID control methods. The two black lines represent the first oscillation's amplitude.

Using the trial and error method for tuning the PID constants, it became clear that PID control was not the best method for controlling these motors. Instead, PI control was better as derivative control caused the system to go unstable even with significantly smaller K_d values than that of K_p,K_i. This was caused by the position used for the error calculations instead of velocity. As a result, the derivative constant is derived a second time in equation 11, causing the derivative constant of PID to become infinitely large and the system becoming unstable. differentiated

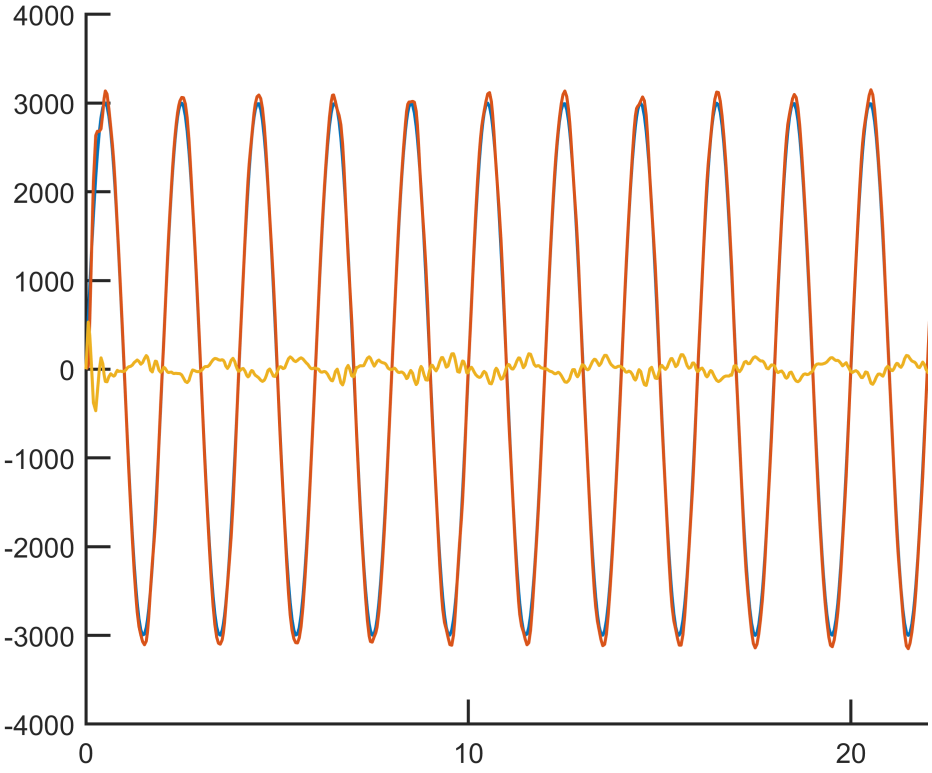


Figure 23. Proportional and Integral control using the trial and error method

The trial and error method effectively removed the slip seen in figures 21 and 22. However, in doing so, it introduced an increased amplitude. as seen in 23, the average error at the oscillation peak was 159 steps. This translates to an error of 3.18°and an average error percentage for each oscillation of 5.3%, which is a significant error. Additionally, the error graph of this cycle is incredibly noisy, with extreme error seen during the first oscillation; the error peaked at 10.6°or 17.73%. As this mechanical model's future experimentation aims to analyse the first oscillation, this error cannot be negated.

To account for this increase in amplitude, the Ziegler-Nichols method was then tested to find the correct PI system for this mechanism.

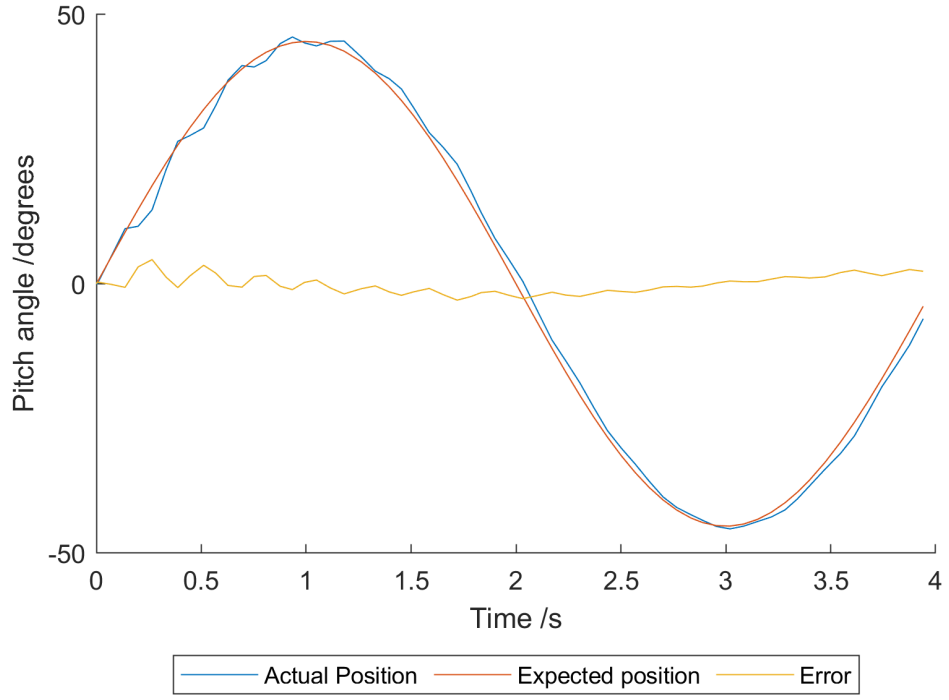


Figure 24. K_u is too low

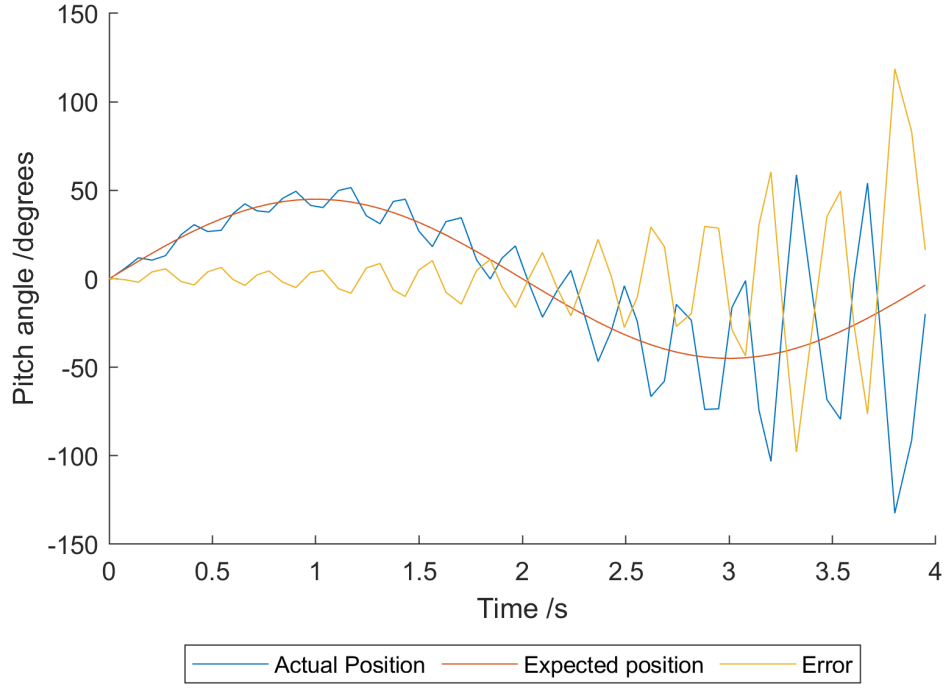


Figure 25. K_u is too low

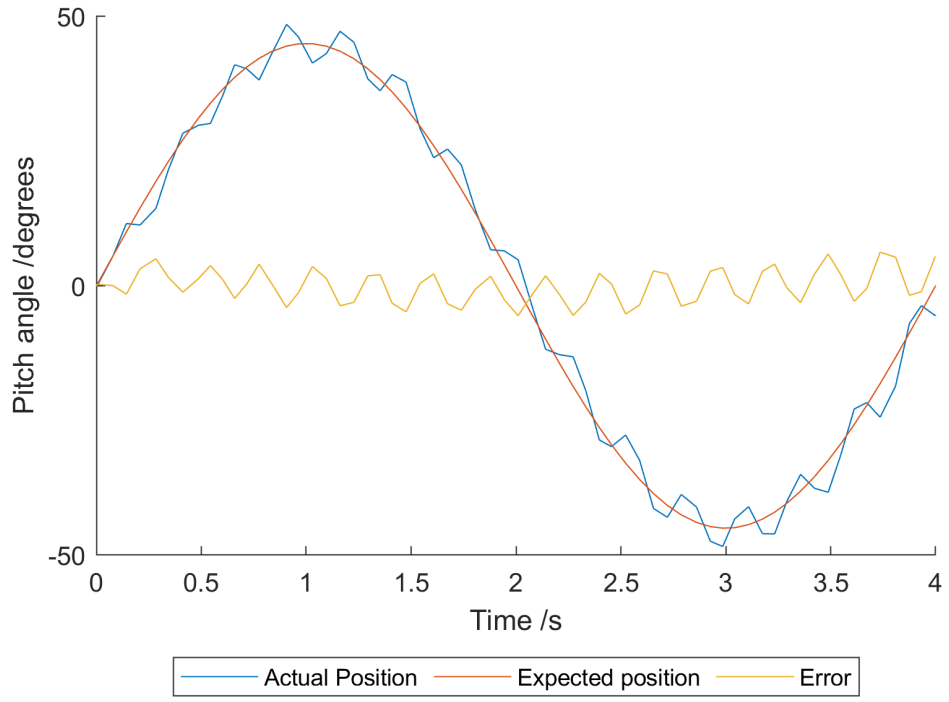


Figure 26. K_u is too low

The Ziegler Nichols method was used at an amplitude of 45 and frequency of 0.1Hz for both pitch and stroke and then confirmed over a range of frequencies and amplitudes. K_u was set at the values of K_p found from the trial and error method, and then increased by values of 0.2 until they moved from stable as seen by figure 24, to unstable as seen in figure25. Then, between these values at a higher degree of accuracy until the value of K_u) was found where they were critically stable.

Interestingly, the values of K_u were significantly different for the two motors. The K_u value for pitch is almost double that as for stroke. The most likely reason is the difference in torque and inertia being felt by the two motors.

K_u Value	Stability
17.680	Stable
17.681	Stable
17.682	Stable
17.683	Critically Stable
17.684	Unstable
17.685	Unstable

Table 3. The Ziegler-Nichols Method values: Pitch

K_u Value	Stability
8.820	Stable
8.821	Stable
8.822	Critically Stable
8.823	Unstable
8.824	Unstable
8.825	Unstable

Table 4. The Ziegler-Nichols Method values: Stroke

As a result of using the Zeigler-Nichols method, the values of P_u could be found to be 0.25s for both pitch and stroke. The values for K_d and K_i are given in table 5

Table 5. The Ziegler-Nichols Method Settings used in this mechanism

Motion	K_p	K_i
Pitch	7.9573	3.3333
Stroke	3.9699	3.3333

4.3 Motion Capture Programming

As a part of this project, a motion capture program was devised using elements of MATLAB's computer vision and image processing toolbox, allowing the motion of the wing to be analysed in further detail. The created program could detect the corners of the model, and with four corners of the wing found, six lines are drawn between them.

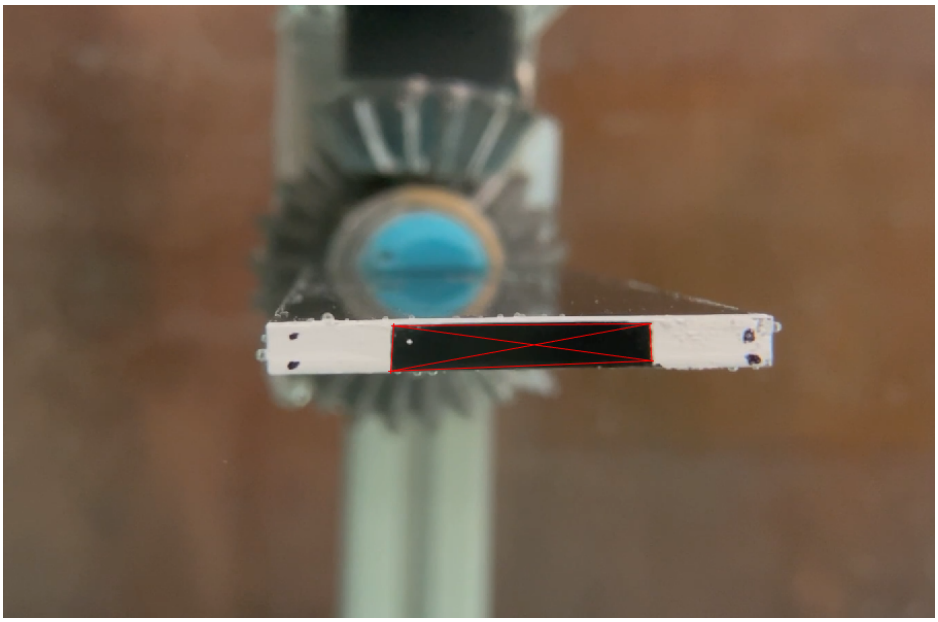


Figure 27. A frame of the motion capture video showing the six tracking lines

The gradients of these lines are calculated, explaining their initial position. From here, for each frame, these corners were re-found, the lines redrawn, and the angle change from the initial version of the line was calculated. This was completed for each frame in the video, giving a database of angles throughout the movement for 20 oscillations. Running at the lowest frequency of 0.1 Hz gave 200 seconds of running time; at 30fps, this gave 6000 data points. This is a greater clarity level than the encoders offer, who averaged 3500 data points, so it should provide a better understanding of the movement between the encoder positions.

However, this program has difficulties detecting the model's corners. This was likely because the image had to be binarized before the corner detection, showing only the differences in contrast. And, as the wing initially was clear, it blended into the background. To fix this, the wing was swapped out for an opaque black wing, and the white paint was used to create boxes to track the wing's tip and lower surface. This allowed the program to track high-contrast centres better.

4.4 Validation of the Experimental Set-up

To confirm the entire setup movement, values from the encoder were recorded, and videos were taken using a 30fps 1080p camera. This motion was entirely submerged under the surface to match Krishna et al. 2022 [17] and to reduce the impact of the water's surface on the camera. If done at the surface, the difference in refraction between the above and below surfaces would make analysing the videos difficult.

Experimentation was done in a 1.2m x 0.8x 1.0m tank, with the mechanism placed centrally to allow for a full range of motion without wall interference. The tank was filled to 0.15m above the height of the wing. This gave a difference of 5 chord lengths from the surface, negating surface effects on the motion capture.

The algorithm was produced so that pitch and stroke were coupled, meaning that running just pitch or just stroke would get the same results as running both pitch and stroke simultaneously. This meant that pitch and stroke movements could be verified separately to improve the motion capture accuracy.

The frequency of the honeybee's flapping motion, as seen in water, is between 40 and 120 Hz.

confusing, seems contradictory but i might just be a bit tired

$$k = \frac{\pi c}{2\phi R_2} \tag{12}$$

$$R_2 = \sqrt{\frac{\int (R_o + r)^2 dr}{R}} \tag{13}$$

The reduced frequency the mechanism was chosen to run at originated from similar experimentation [17] where a frequency of 0.37Hz was used. This was then verified for this set-up using equation 12 with c being the stroke amplitude, and R_0 being the root cut out, the distance from the wing's root to the rotation axis. Three frequencies were chosen from this: 0.1Hz, 0.25Hz, and 0.4Hz.

In the water-treading mode, the stroke and pitch motion are out of phase. This simulates the water-treading mode as seen in the [17] 2022 Krishna et al. The water treading mechanism was chosen for these verification tests to compare future force-torque data to this paper.

To confirm the pitch and stroke motion separately, the pitch was run at angles of 45,60,75,90, and the stroke was run at 15,30,45,90, chosen to match the honeybee's amplitude in oscillation [1] and also to reflect previous data sets [9, 17]. The mechanism was run at the three chosen frequencies at each amplitude, giving 24 motions to study. Each motion was repeated three times for the motion capture and their corresponding encoder values, resulting in 72 videos for analysis.

	Amplitude [°]	Frequency [Hz]
Motion 1	45	0.10
Motion 2	45	0.25
Motion 3	45	0.40
Motion 4	60	0.10
Motion 5	60	0.25
Motion 6	60	0.40
Motion 7	75	0.10
Motion 8	75	0.25
Motion 9	75	0.40
Motion 10	90	0.10
Motion 11	90	0.25
Motion 12	90	0.40

Table 6. Pitch motions used in experimentation

	Amplitude [°]	Frequency [Hz]
Motion 13	15	0.10
Motion 14	15	0.25
Motion 15	15	0.40
Motion 16	30	0.10
Motion 17	30	0.25
Motion 18	30	0.40
Motion 19	45	0.10
Motion 20	45	0.25
Motion 21	45	0.40
Motion 22	90	0.10
Motion 23	90	0.25
Motion 24	90	0.40

Table 7. Stroke motions used in experimentation

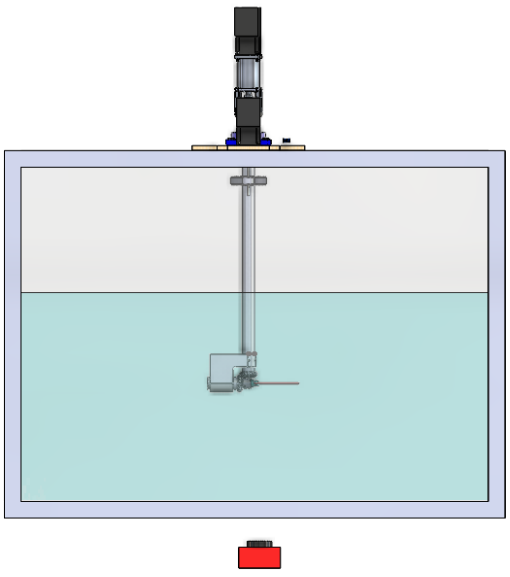


Figure 28. Camera placement for the stroke motion videos

The tank used was suspended off the floor. This allowed a camera to be placed below the tank and the wing to be filmed from below, as seen in figure 28. As from above, the mechanism would obscure the view of the wing.

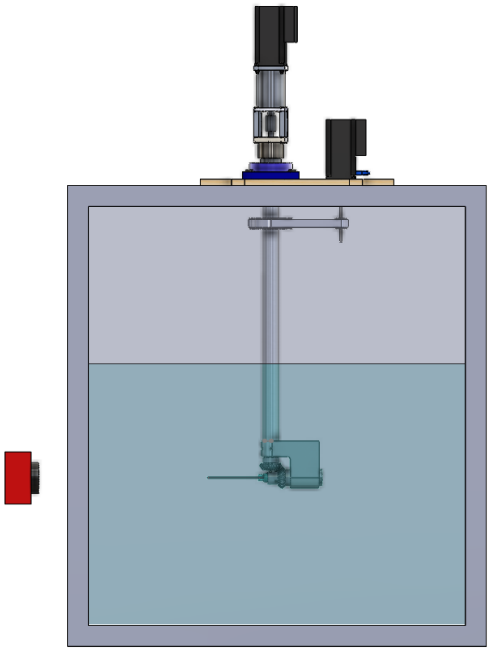


Figure 29. Camera placement for the stroke motion videos

For pitch, a mount for the camera was constructed that placed the camera at the same height as the wing to give a plan form view of the wing tip where the dots were placed, as seen in figure 29.

5. Results

The desired oscillatory pattern must be true at both the motor and mechanism levels. The encoder data has been used to confirm that the motors are moving in a sinusoidal pattern, and motion capture has been used to confirm that this translates to the mechanism.

5.1 Encoder Verification

link to start?

This section contains the results from the onboard encoders on the motors with the wing submerged. To validate this data, it must be correct in three areas: amplitude, shape, and frequency. Pitch and stroke data have been considered together in these sections as both data sets carry almost identical characteristics at the encoder level; the methods used to program both pitch and stroke are virtually identical, with the only exception being their starting position and PID constants.

5.1.1 Shape Analysis

Confirming that the motion is, in fact, sinusoidal supersedes confirmation that it is the correct sinusoidal waveform.

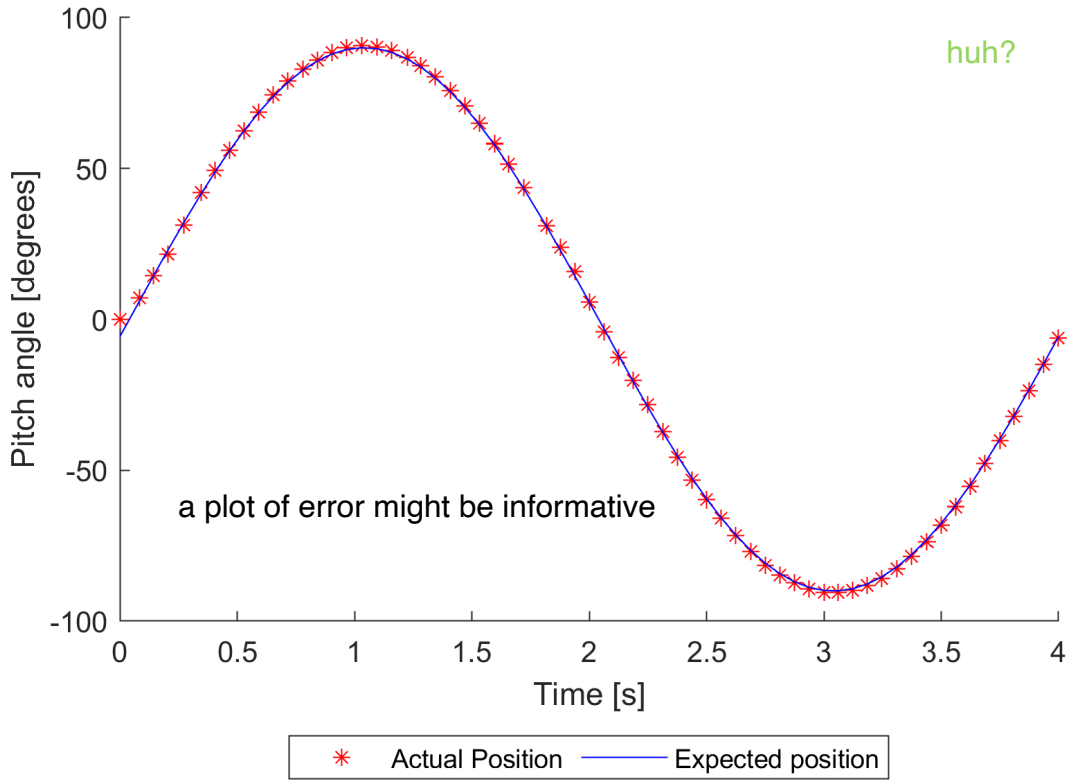


Figure 30. one oscillation at 90°amplitude, 0.25Hz frequency

From figure 30, it is clear that the motion is, in fact, sinusoidal, as the actual position points read from the encoders during motion follow a sinusoidal pattern throughout the oscillation, whilst there is some deviation

from the expected positions, this is minimal. Having this occur means that the communication rate with the motors is high enough to allow for the estimation of a sinusoidal waveform; if this were too low, then the waveform would look linear in places. The increase in points occurring around the peaks in amplitude is expected and preferred as It allows more iterations from the PID loops to occur at this crucial point in the waveform. This means the motors follow the required oscillation pattern that the honeybee uses.

This, also being the first oscillation in this run, confirms that the motors meet the expectation of consistent runs from the first oscillation, meaning that this apparatus can be used to understand the formation of the vortices and momentum in the first few oscillations.

This analysis also confirmed that the PI control effectively controlled the motors. While there is some variation in position from where expected, the overall shape is sinusoidal, which is an improvement from before PI control was enabled.

5.1.2 Frequency Analysis

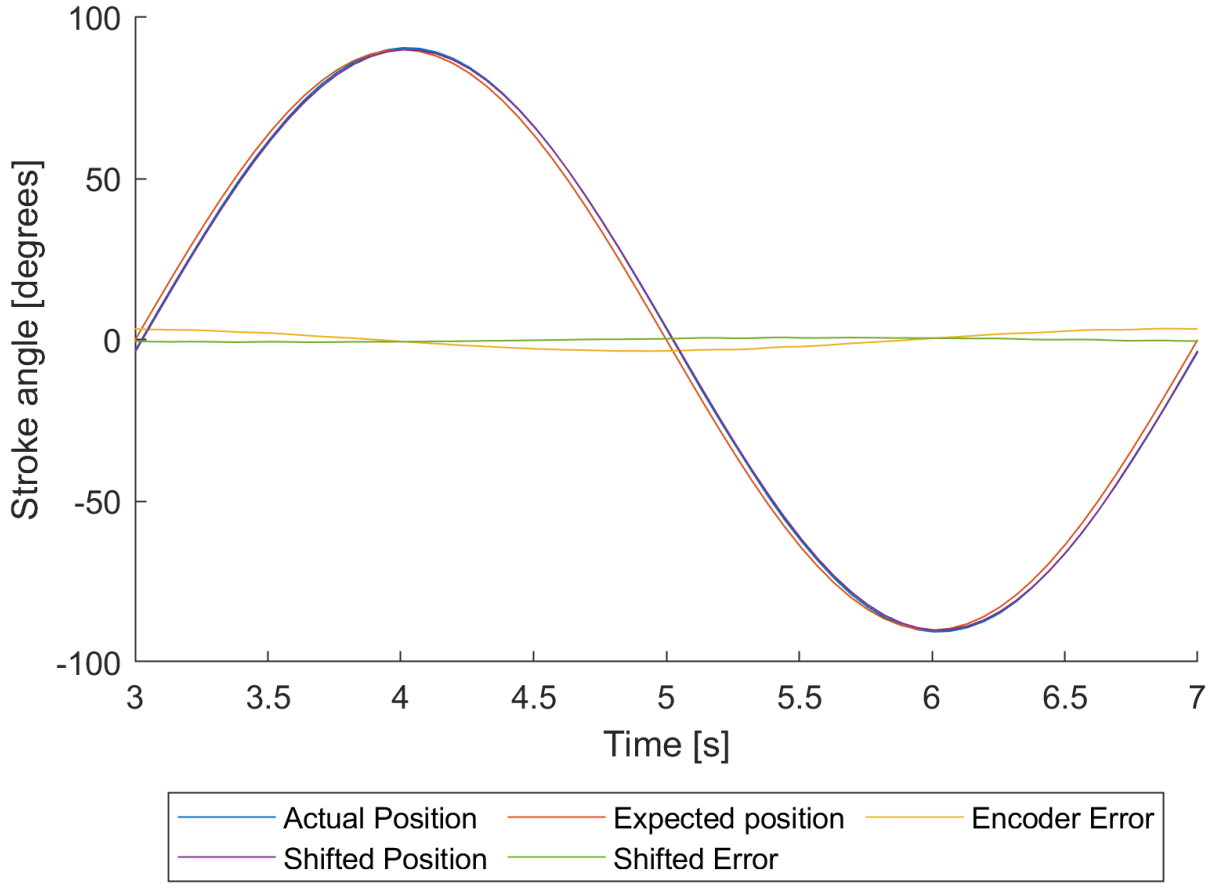


Figure 31. One oscillation in the stroke motion at 90°0.25Hz

It isn't easy to distinguish between theoretical and experimental data when 20 oscillations are shown in one figure. To account for this, only one oscillation has been shown in figure 31. This allows for a closer analysis of

the waveform.

timed difference not defined, sounds like a phase shift?

Figure 31 shows one oscillation for a 0.25Hz 90°stroke motion. In this, it is clear that there is a shift between the expected and actual positions of the encoders. The timed difference between the two curves averages 0.06s seconds for pitch and 0.03 for stroke. The difference between the two occurs from the order they are called in the program, stroke being called first. This means that, unfortunately, the program has an intrinsic delay of 0.03 seconds between the two motions. This inaccuracy in future code iterations should be accounted for using the lead/lag mechanism built into the code. To account for this, the shifted position line was used. This is a mathematical sinusoidal waveform calculated using equation 4. Where the lead/lag constant L in this case accounted for the delay in communications.

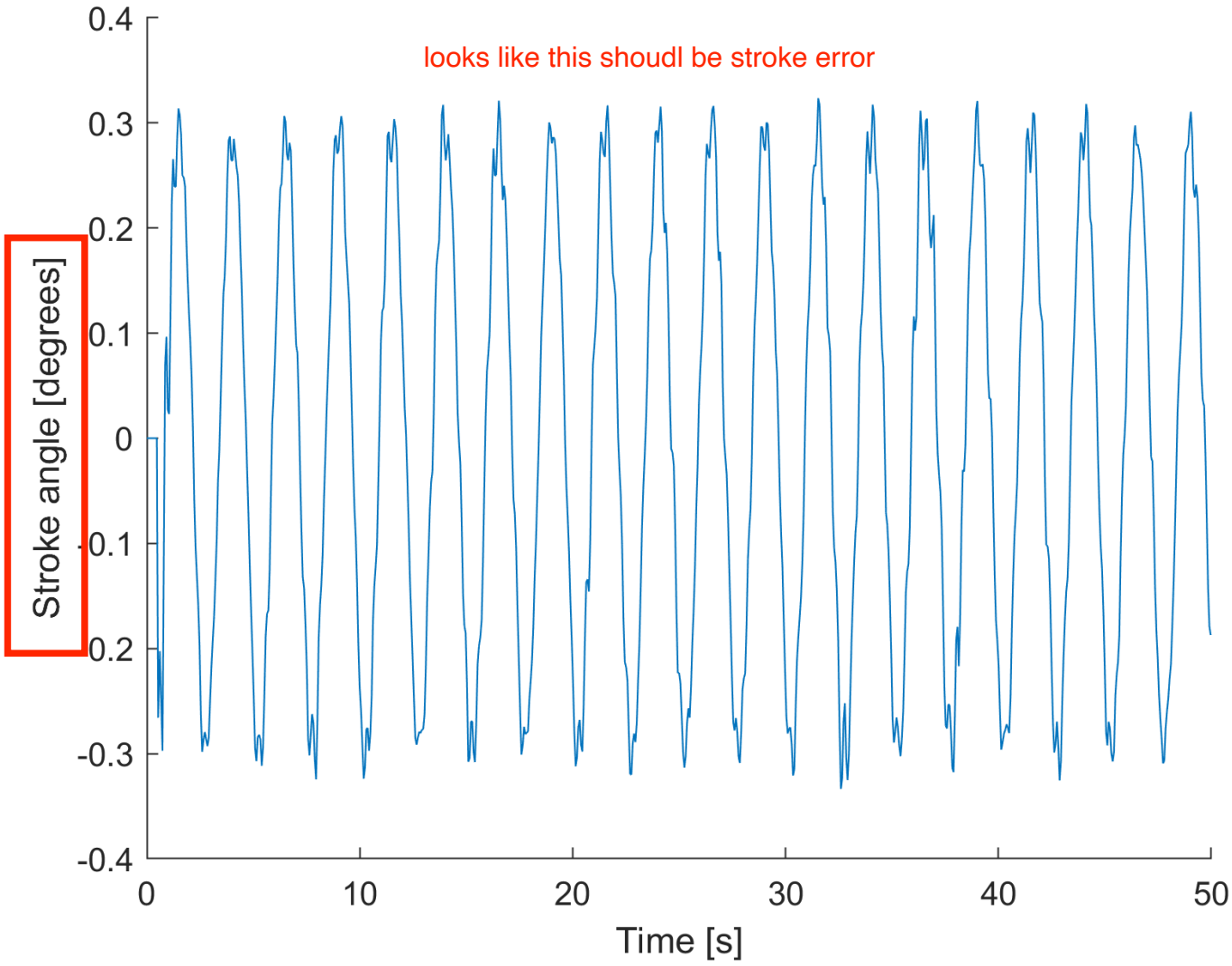


Figure 32. error experienced in 20 oscillations in the stroke motion at 15°amplitude and 0.4Hz frequency

5.1.3 Amplitude Analysis

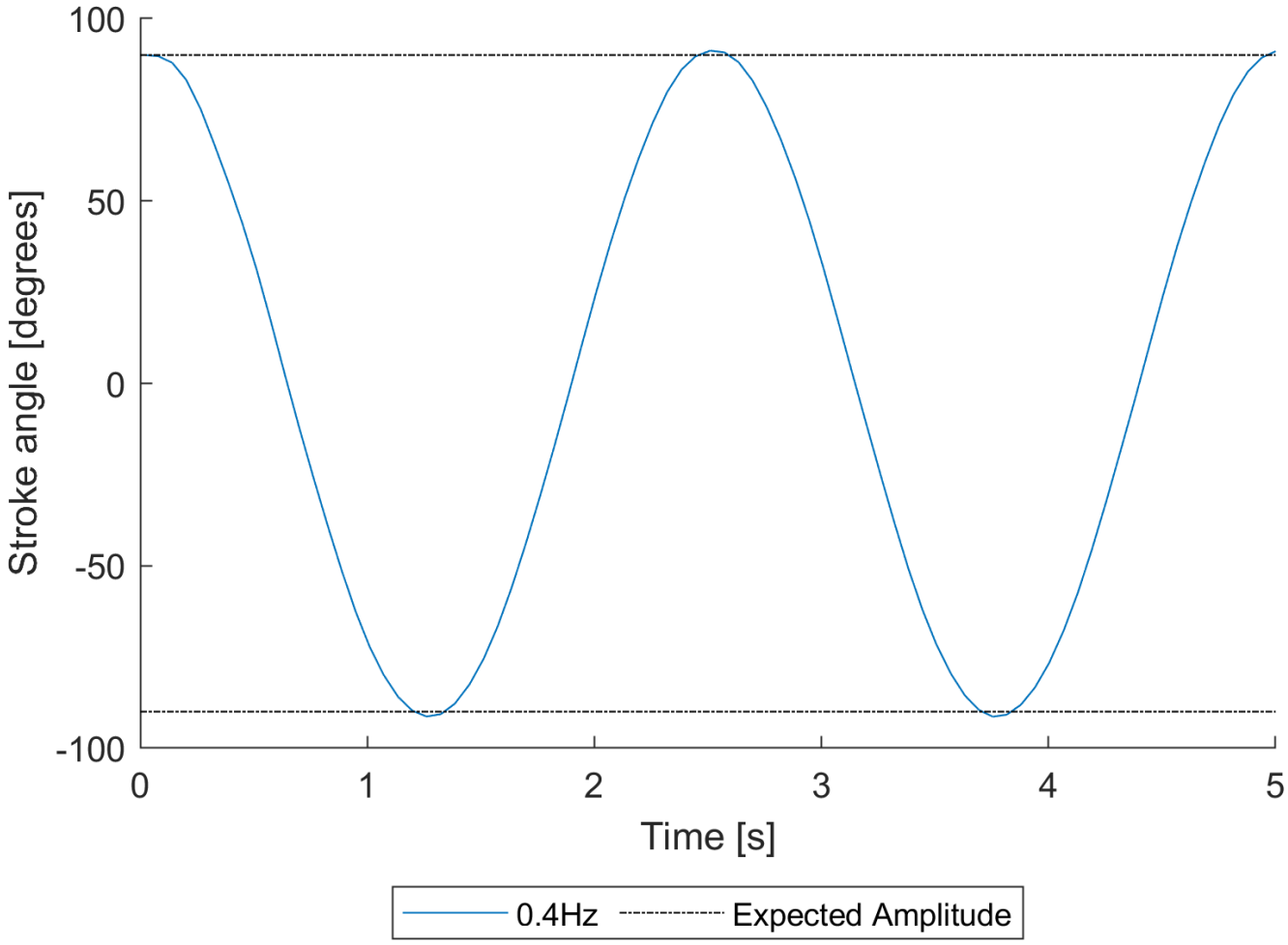


Figure 33. 20 oscillations in the pitch movement at an amplitude of 90°and a frequency of 0.1 Hz

that looks like fewer than 20 to me

In figure 33, it is clear to see overshoot present as the position line exceeds the expected amplitude. This is seen in both the pitch and stroke motion for all motions; however, it only comes pronounced enough to be graphically visible at 0.4Hz. In figure 33, it can be seen that the starting position is correct, and no overshoot is visible for this position. This is because this position is set during the start-up of the mechanism as the stroke motion starts at its maximum position to match the water-treading mode. The start position confirms that this is not a calculation error in the intended amplitude but an issue with the motion section of the program.

As this is more pronounced for the higher frequencies, this is likely caused by two sources working in tandem: the PI control algorithm and the communication delay. As a PI system was chosen over PID, the system is susceptible to overshoot, and as the higher frequencies spend shorter periods at these high amplitudes, there is less time to correct the error. Each change in speed takes 6.23ms to be carried out. Working at 0.4Hz provides 805 opportunities to read position and change velocity, whereas at 0.1Hz, this is 3220. This impact

is astronomical and likely the main cause of most algorithm-based errors in the system. Additionally, the system has no way of being more accurate at that frequency without changing the control method. The only way around such an issue would be to improve communication speed. This could be done using two USB to RS485/422 Converters, one for pitch and stroke. As this is the only line of communication to both motors, splitting this up allows for data to be read and commands sent simultaneously without having to share a connection.

Figures 34 and 35 confirm the theory that an increase in frequency is indirectly proportional to the error seen. Each line on these graphs represents the average maximum error for the various runs completed at each frequency. Interestingly, there is variation between pitch and stroke motion. Both pitch and stroke carry a 45°and 90°amplitude motion pattern. However, the average maximum error for 90°in the pitch motion at 0.4 Hz is almost 1/3rd that of its stroke counterpart, and at 0.1Hz, the pitch motion experiences around 5 times more error. This could come from the placement of the sinusoidal waveform not being accurate in some runs, creating calculation errors; however, most likely, it comes from the PI constants for pitch and stroke. The PI constants are the characteristics of the controller, so whilst the pitch and stroke require different constants, they will act slightly differently because of this. From such results it can be recommended that other methods of tuning the PI constants be explored and results compared with this study in an attempt to reduce this error.

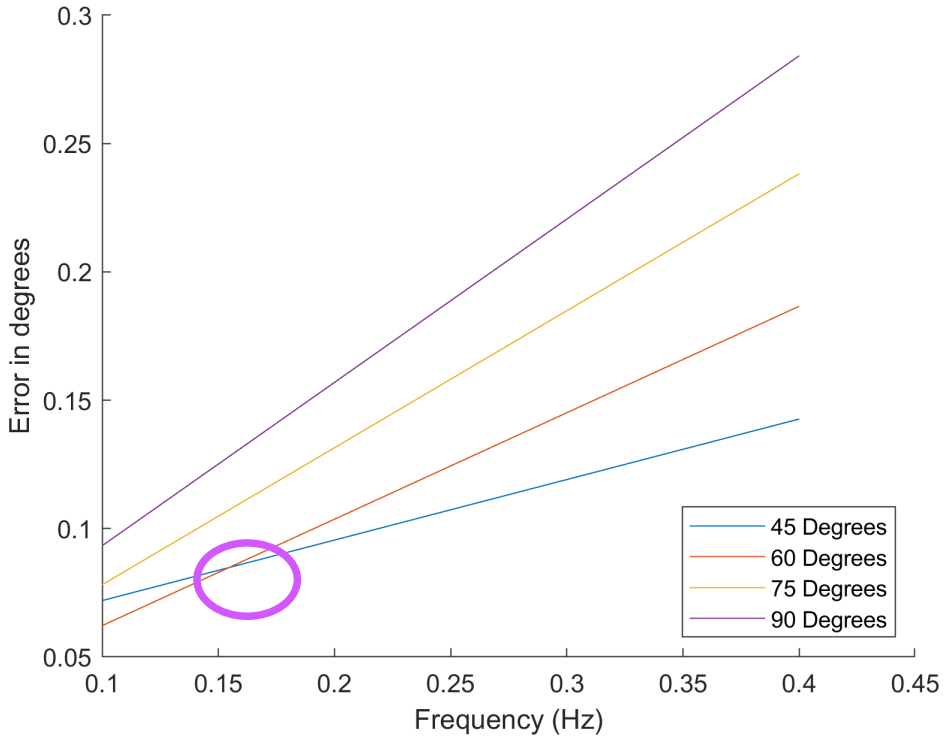


Figure 34. Average maximum deviation in pitch

i havent read all the way but if this isnt commented on it should be

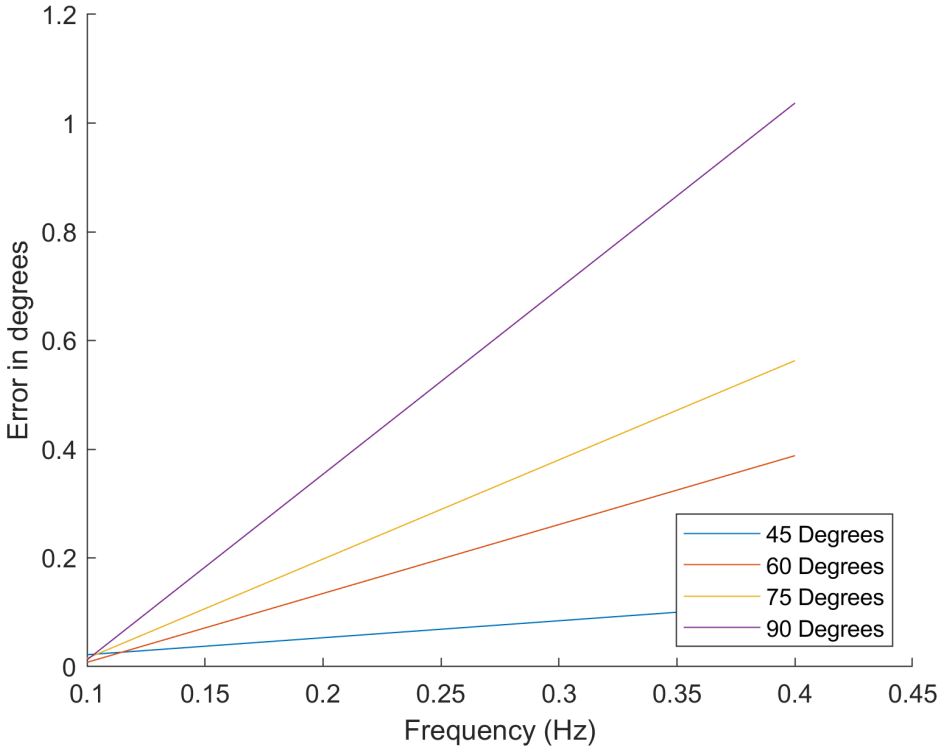


Figure 35. Average maximum deviation in pitch

Despite this, it is clear that the encoder values, whilst experiencing some error, are almost entirely accurate, with the largest average percentage error being seen in the stroke motion for 90°amplitude and 0.4 Hz frequency, having an error of 1.227%. These runs also accounted for the largest errors seen within all the runs, the maximum being a 2.26°error or a 2.511% error, which occurred at a singular point in the second trial run for this motion pattern and was not found in the repeated runs, therefore, can be deemed to be an outlier. This means that the mechanism on average falls within the 2% expected error, however, there have been cases in which it extends out of this range at the higher frequencies. This identifies the 0.4Hz frequency as the upper bound where this mechanism can run accurately at this amplitude. Identifying whether this is also the case in motion capture will cement the validity of this model.

Using the shifted encoder error as the error experienced from shape, amplitude and frequency combined, the majority of error falls below 1% at the encoder level. This is close to the desired pitch and stroke motion accuracy. However, the encoders do not account for the rest of the system, only the motors. To see if this accuracy continues in the rest of the system, the motion capture must be analysed

5.2 Motion capture Verification

For motion capture, pitch and stroke have been separated as whilst the coding is almost identical; the subsystems are not. Verifying each motion must be correct in three categories: Amplitude, frequency, and shape.

links back make it all feel more cohesive

Motion capture was completed for the 36 pitch runs using the above-explained motion capture programming. However, when translated to stroke motion, the algorithm found it difficult to keep an accurate computer model of the wing, likely because the model was further away from the camera, and the background of the shots included the mechanism, making it harder to distinguish the wing.

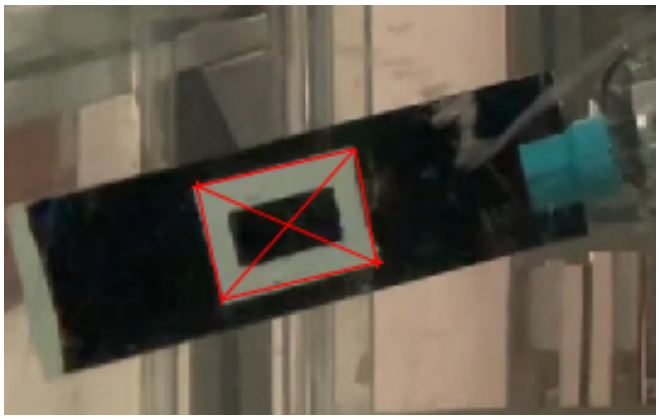


Figure 36. Average maximum deviation in pitch

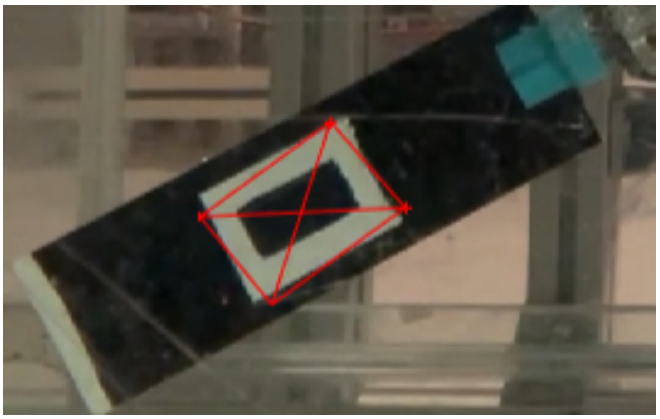


Figure 37. Average maximum deviation in pitch

This meant that, on runs where it didn't fail, the model slipped from its intended position, and all the amplitude data was decreased by a random amount each cycle. This can be seen in figure 37, where the tracking box couldn't maintain its position within the rectangle. Graphically, this gave unreliable data. As seen in figure 38, this suggests that the shifting seen before the PID is still in play and that the amplitude is wildly reduced from where it should be. This is not the case, and the details of this can be seen in the motion capture stroke amplitude analysis section of this report

However, this result is erroneous in and of itself, as almost every other motion capture run failed before the 20 oscillations were complete. This occurred when the corner it was tracking shifted so far that it fell off the wing and then attached itself to a random point in the background and started tracking that instead. This meant the tracking rectangle looked like that in figure 39. These results were completely unusable.

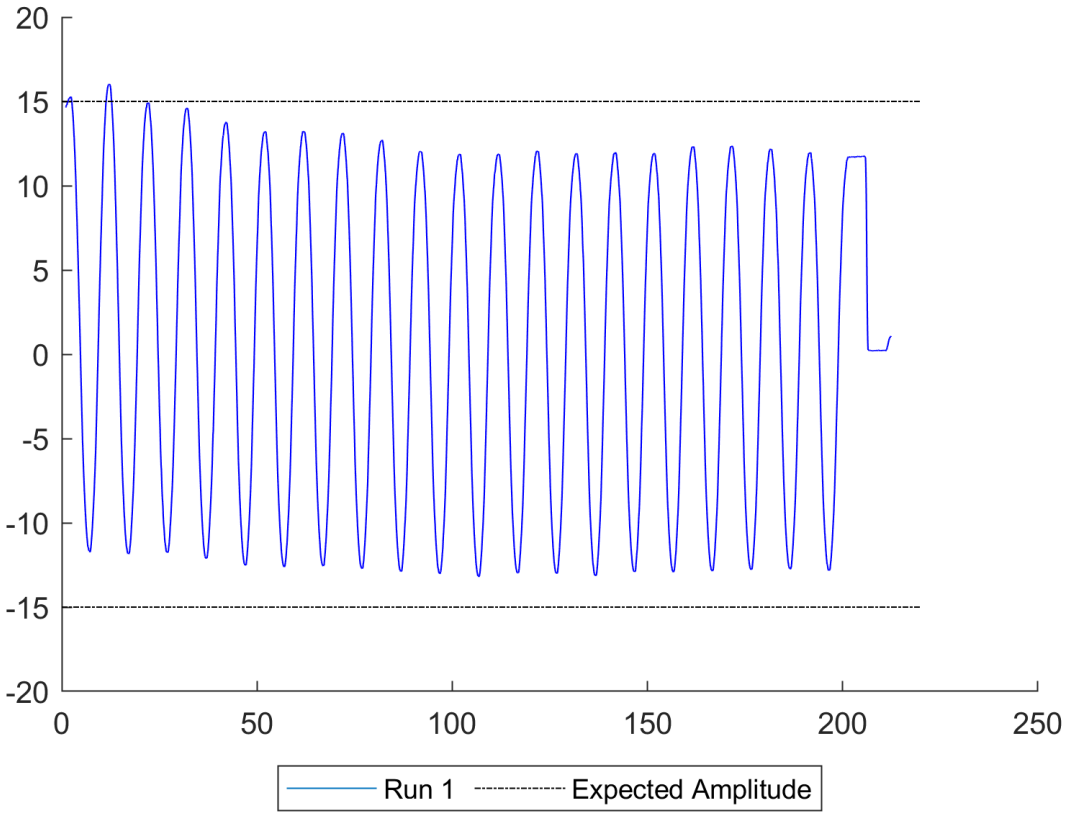


Figure 38. Average maximum deviation in pitch

lmao

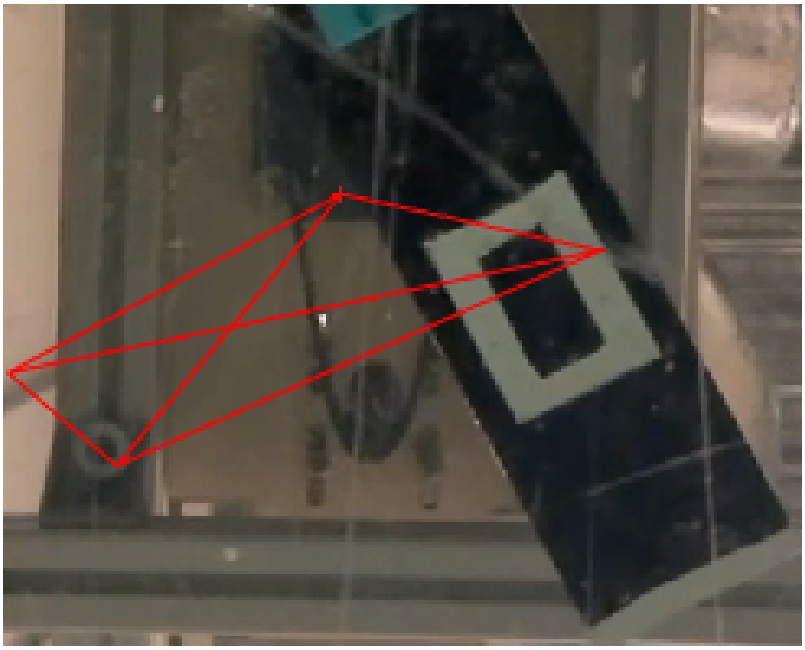


Figure 39. Average maximum deviation in pitch

To allow for some analysis of the stroke motion to occur, a secondary user-based program was created. This allowed the user to go through each video frame by frame to select the frames in which the peak amplitude

occurs and calculate the frequency. As a result, the shape of the oscillation could not be analysed, as it would not have been feasible to compute each frame by hand. Future motion capture will need to be completed to verify the results of this cruder version of motion capture. The user-based version had an accuracy of ($\pm 0.4^{\circ}$) when calculating amplitude due to the user interaction. In addition, as this is seen as an issue for the stroke data, it can be assumed that the motion capture program is not as robust as is needed for this level of analysis; therefore, it must also be considered a point of error within the pitch analysis.

5.2.1 Shape Analysis

Due to the failure of the motion capture, the shape of the motion can only be considered in the pitching motion.

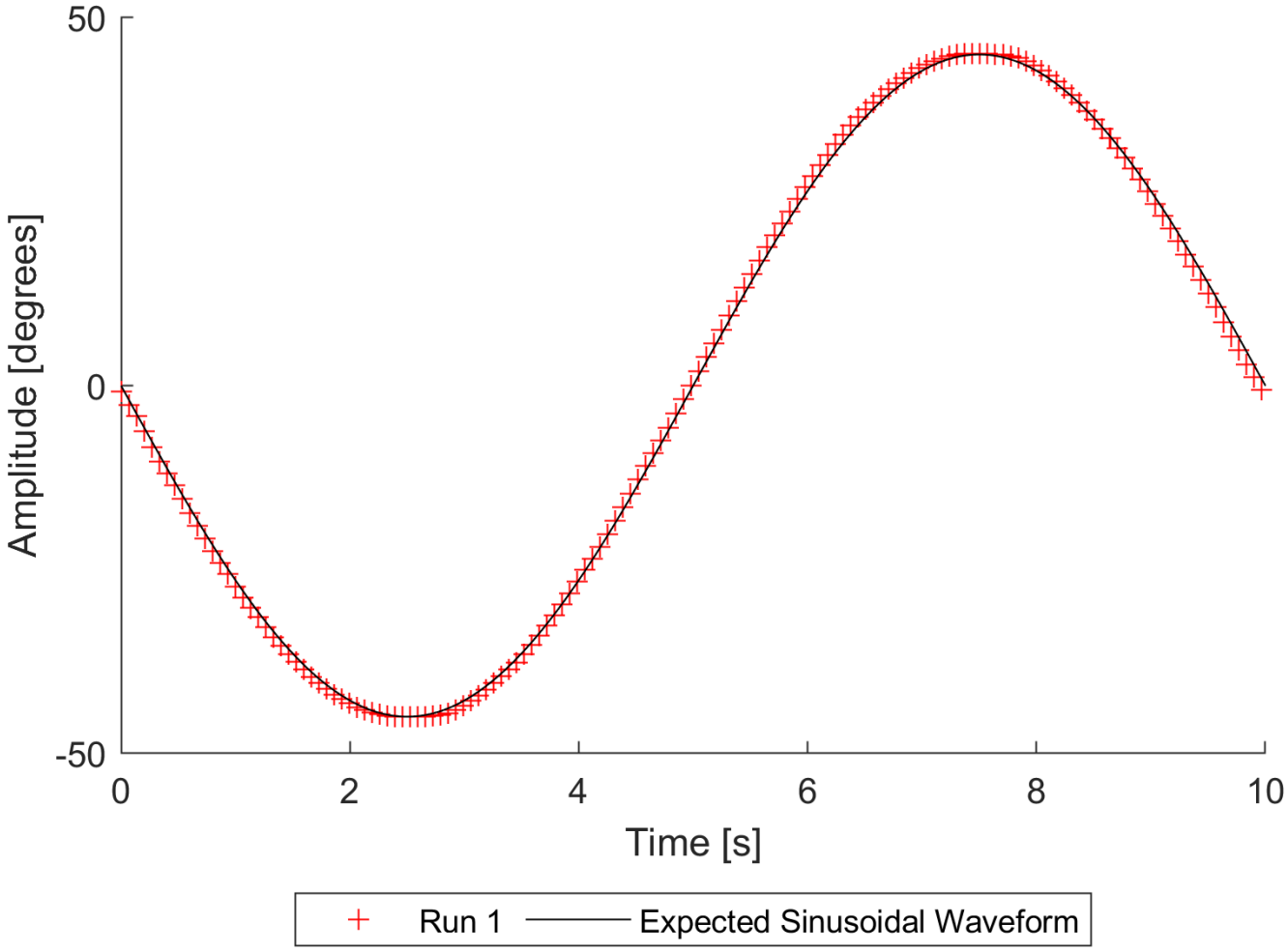


Figure 40. Average maximum deviation in pitch

Figure 40 makes it clear that the motion in the pitch direction is definitely sinusoidal in motion. Confirming that the motors are able to covey this oscillatory motion throughout the rest of the mechanism. This also confirms that the system remains sinusoidal even between the velocity changes the motors provide. This means the system's motion is as close to sinusoidal as possible with this motor control method. This figure also eludes

to a frequency different from the expected 0.1Hz as the curve occurs before the expected waveform in most places. This is further analysed below.

5.2.2 Pitch Frequency Analysis

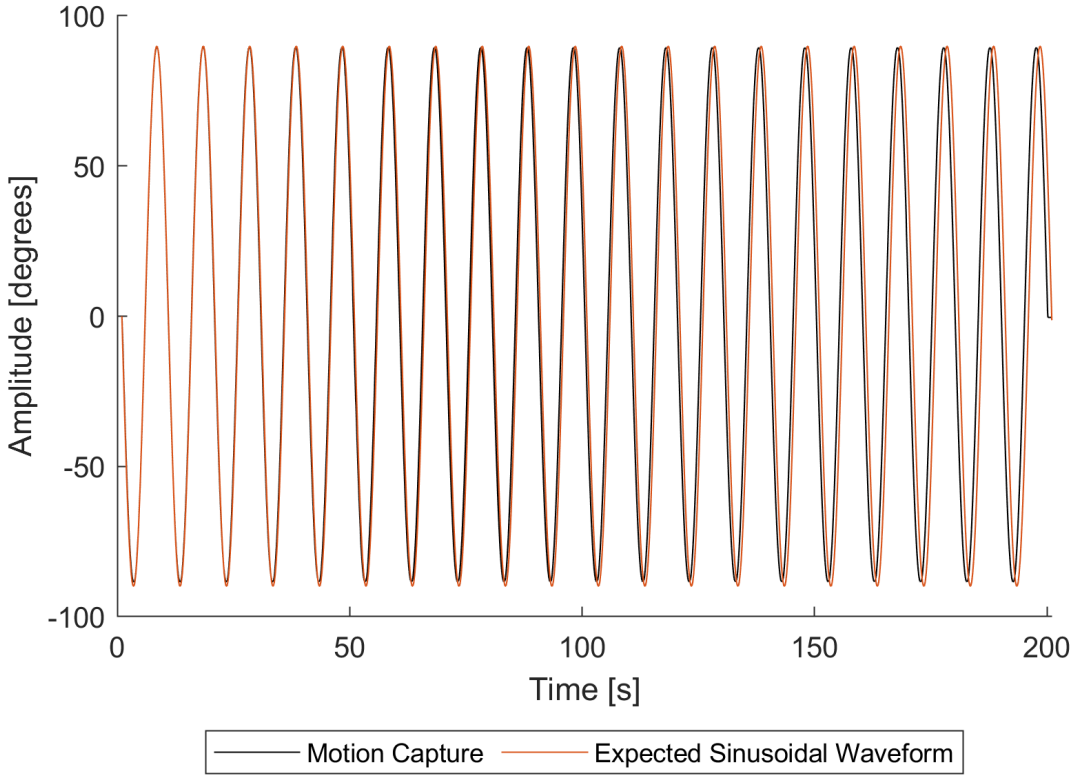


Figure 41. Motion 1 showing the frequency of the motion capture oscillation against the expected frequency

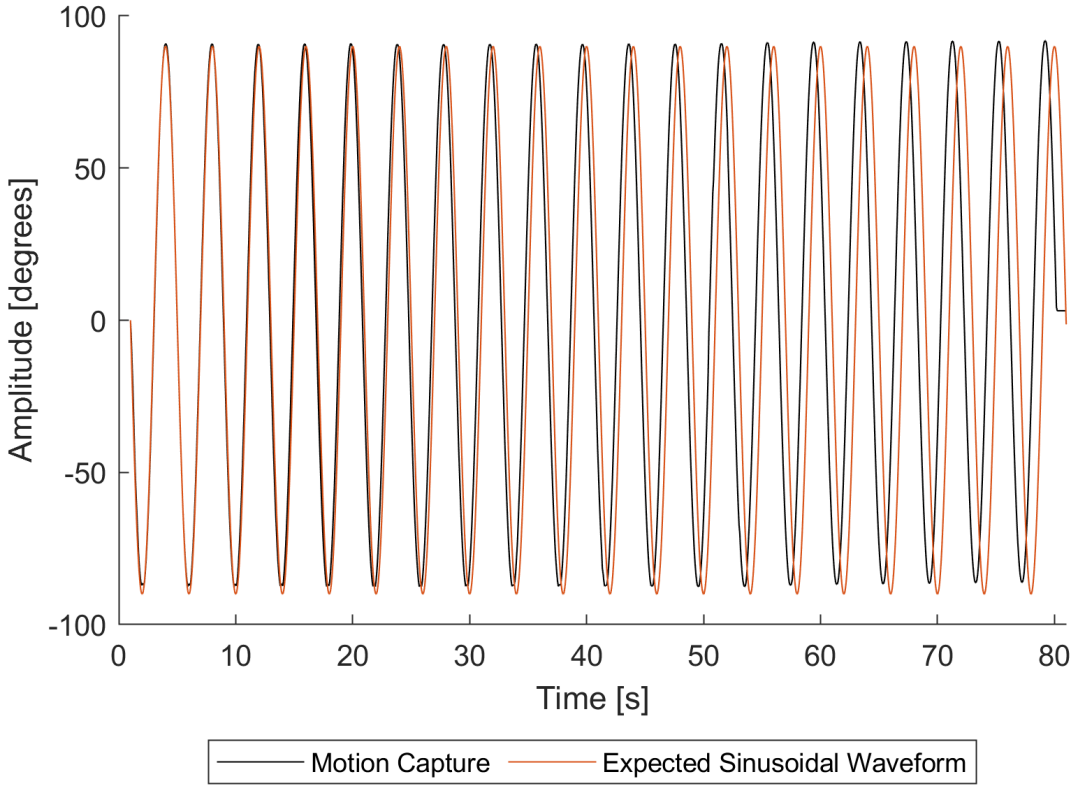


Figure 42. Motion 2 showing the frequency of the motion capture oscillation against the expected frequency

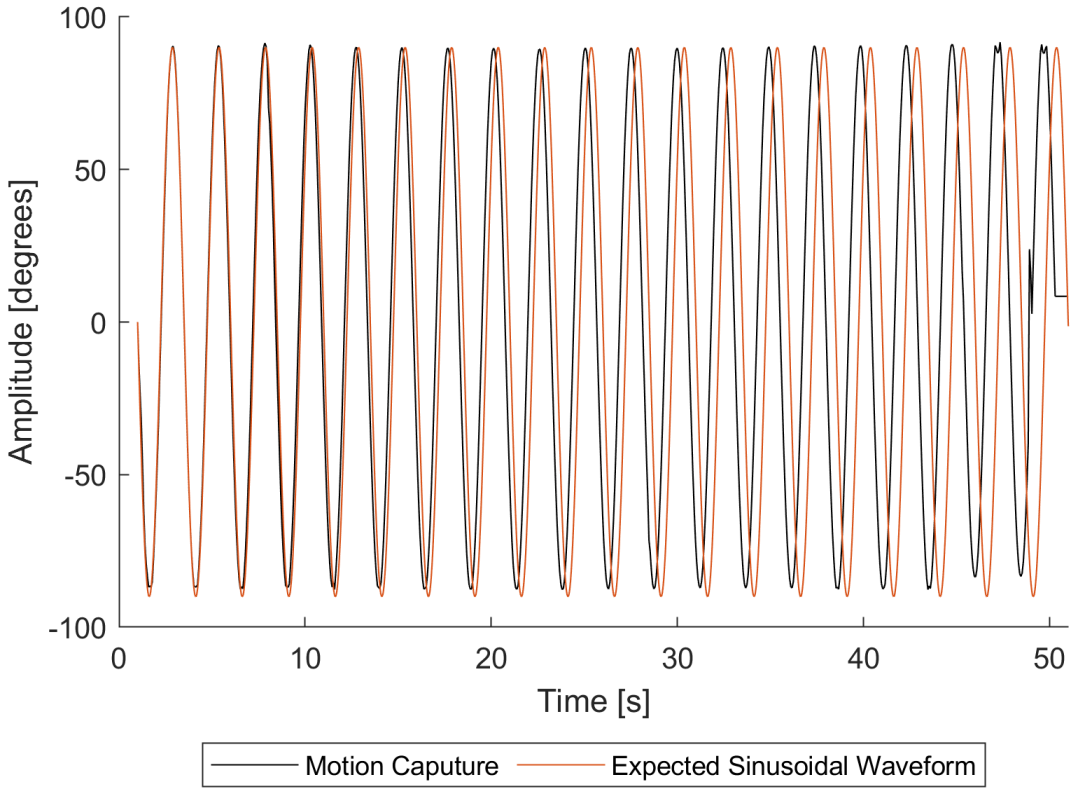


Figure 43. Motion 3 showing the frequency of the motion capture oscillation against the expected frequency

Figures 41, 42, 52 show the motion capture oscillations against a standard sinusoidal waveform with the correct frequency. The error rates for these are 0.0110%, 0.3223%, and 1.0198%. The first oscillations almost line up, meaning this isn't the same issue seen in the encoder data where it needed to be shifted to forwards in time to remove the error. However, this shift grows with time, suggesting the frequency is too fast for all three. The error is clearly worse for the high-frequency variation than the low variation. However, such a claim that the mechanism is going faster than it is being commanded makes little sense. This is not an issue seen at the encoder level. The shifted line matches the encoder data for every oscillation. This suggests that the mechanism moves to positions before the motor controlling it has moved to that position. This categorically isn't possible. Therefore, confirming another flaw within the motion capture, as this issue gets worse with the higher frequency, is not the frame rate of the video being miscalculated, as the error would be proportional to the length of the video.

To confirm this is an error within the motion capture, the videos were individually analysed; the time for 19 oscillations was recorded, and the frequency was calculated using $f = 19/t$. The final oscillation was ignored as the system is programmed to stop after the time taken to complete 20 oscillations, regardless of its position in the cycle.

Table 8 dictates the video version of the frequency and the motion capture.

Table 8. A table showing the pitch frequency error

Motion	Amplitude [°]	Expected Frequency [Hz]	Video Frequency [Hz]	Motion Capture Frequency [Hz]	Video Error [%]	Motion Capture Error [%]
1	45°	0.1000	0.1000	0.1000	0.0110	0.0110
4	60°	0.1000	0.1000	0.1002	0.0950	0.2104
7	75°	0.1000	0.1000	0.1006	0.033	0.6100
10	90°	0.1000	0.1000	0.1004	0.086	0.4060
2	45°	0.2500	0.2499	0.2508	0.0250	0.3223
5	60°	0.2500	0.2498	0.2522	0.0649	0.8811
8	75°	0.2500	0.2498	0.2517	0.0667	0.6905
11	90°	0.2500	0.2501	0.2522	0.0432	0.8852
3	45°	0.4000	0.3997	0.4041	0.0667	1.0198
6	60°	0.4000	0.3997	0.4056	0.0697	1.3882
9	75°	0.4000	0.3992	0.4032	0.2002	0.7997
12	90°	0.4000	0.3992	0.4057	0.2001	1.4110

Both motion capture and video analysis suggest that there is no correlation between amplitude and expected frequency. Only that it is reliant on the frequency it is trying to emulate. The difference between the two supposed error rates for frequency is around a factor of 10 higher; while both are minimal, this is a significant difference between the two. Additionally, they suggest completely different trends. Motion capture suggests the motion was too quick in almost every case, and video analysis suggests it is, on average, too slow.

Given that the motion capture suggests that the mechanism is moving quicker than the motors by 0.71s in motion 12, this is likely incorrect, as this isn't physically possible. The values from the video analysis should be taken as a more accurate representation of the motors' motion.

The error rate is below 0.2002% for all motion patterns; there seems to be no correlation to the amplitude in which the motion was run as to which the error occurred. The only correlation to the frequency at which it occurs is weak. However, most are slower than their expected frequency. This is not an error seen in the encoders and was verified using the original videos, not the motion capture algorithm, ruling this out as the cause. The error seen here is minimal and is not a concern for future testing. As the frequency relies more on the motors than the mechanism, this is likely to cause this error; if so, the stroke frequency analysis results should match that of the stroke.

this bit is so much fun to read and your frustration comes across really well. idk if thats a good thing for the report but its got me engaged in the rest of it. consider writing a book if you get bored of engineering

5.2.3 Stroke Frequency Analysis

Table 9. A table showing the pitch frequency error averaged across 3 oscilations

Motion	Amplitude	Expected Frequency [Hz]	Video Frequency [Hz]	Video Error [%]
Motion 13	45°	0.1000	0.1003	0.2793
Motion 16	60°	0.1000	0.1000	0.0167
Motion 19	75°	0.1000	0.1002	0.2001
Motion 22	90°	0.1000	0.0999	0.0500
Motion 14	45°	0.2500	0.2508	0.3085
Motion 17	60°	0.2500	0.2501	0.0058
Motion 20	75°	0.2500	0.2507	0.2805
Motion 23	90°	0.2500	0.2499	0.0250
Motion 15	45°	0.4000	0.4005	0.1334
Motion 18	60°	0.4000	0.4000	0.0130
Motion 21	75°	0.4000	0.4000	0.0200
Motion 24	90°	0.4000	0.3998	0.0500

Frequency analysis was again completed for the stroke motion; this has a slightly wider variation in frequency, with values both above and below their expected target, with no correlation to amplitude. One reason for this could be a rounding error. The velocity of the motors has a resolution of 0.0042 revs/second. In calculating the velocity, a higher resolution is provided in steps per second. This gives a resolution for the stroke motion of 0.378°/second. Losing this accuracy becomes more prevalent at the lower gear ratio seen in the stroke motion and causes slight alterations in frequency, as seen here. Despite this, the largest error seen was 0.3085%, so it is hardly concerning.

5.2.4 Stroke Amplitude Analysis

Figure 44 shows the range of amplitudes from the 60 oscillations done for each of these three motion profiles. In figure 38, a frequency of 0.1Hz and an expected amplitude of 15°was shown. This correlates to the first line in figure 44. The difference between the two is monumental. Figure 38 suggests a range in maximum amplitude of 4°compared to the video analysis where a range of 0.38°which is within the tolerance of the user-based motion capture. Again, this highlights why the motion capture data for stroke was deemed too inaccurate.

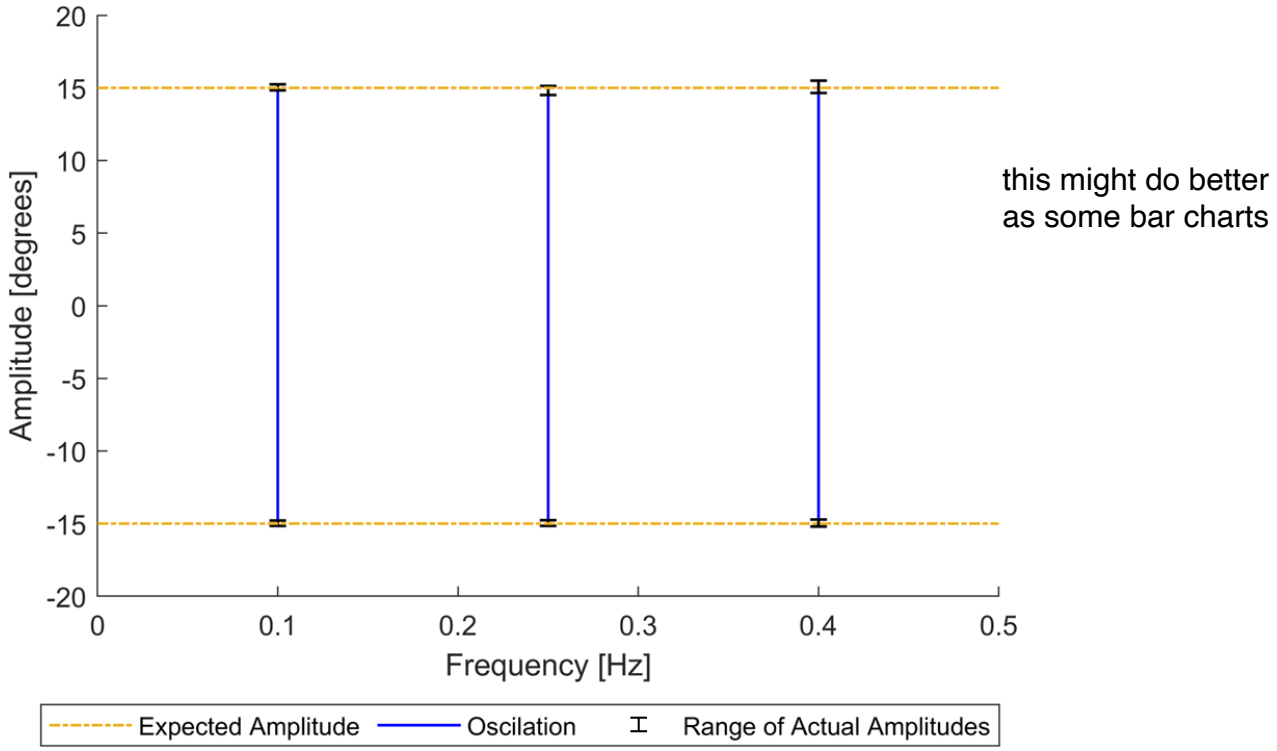


Figure 44. Amplitudes from motions 13,14,15. With error bars to denote the variation in amplitude

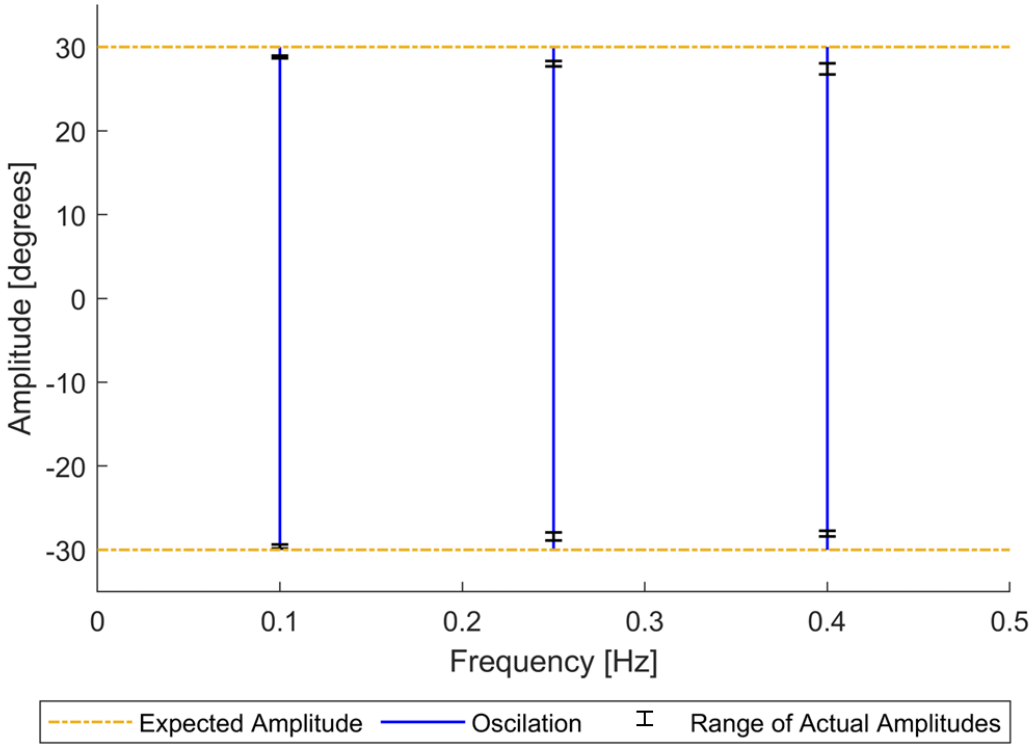


Figure 45. Amplitudes from motions 16,17,18. With error bars to denote the variation in amplitude

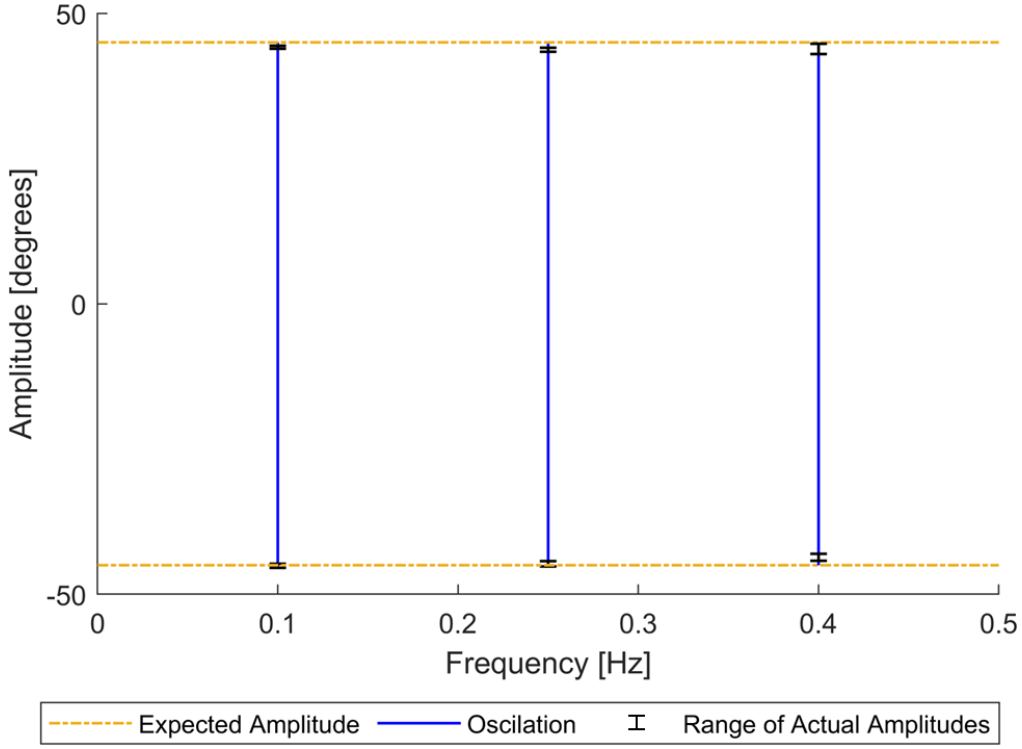


Figure 46. Amplitudes from motions 19,20,21. With error bars to denote the variation in amplitude

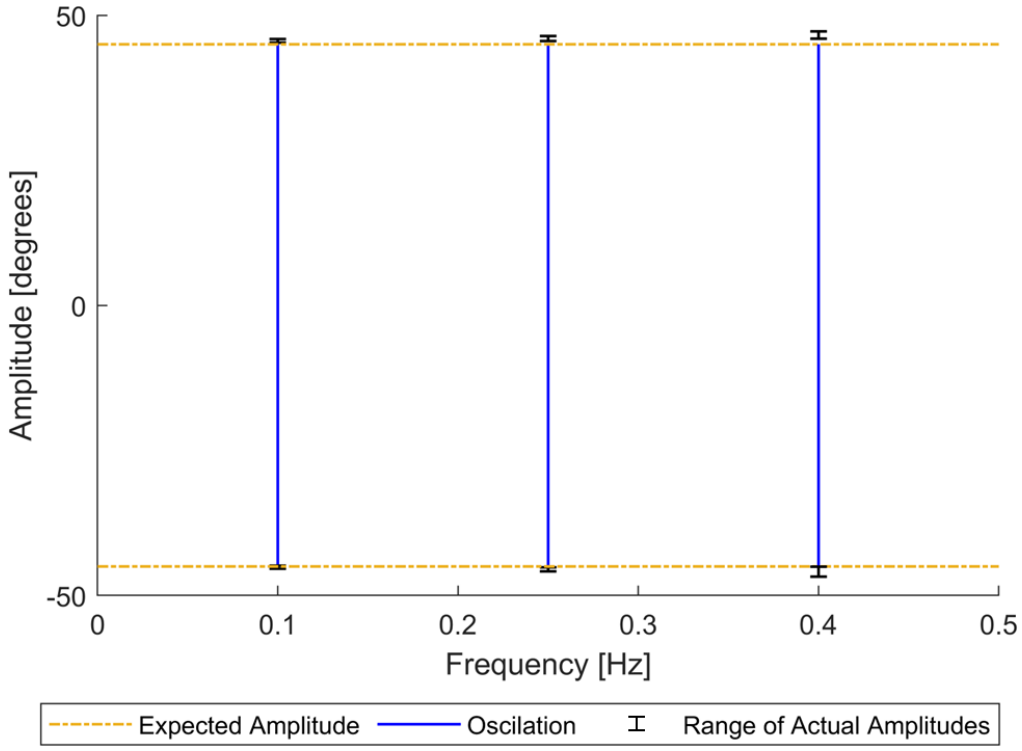


Figure 47. Amplitudes from motions 22,23,24. With error bars to denote the variation in amplitude

The data shown in figures 45, 46, 47 shows a fluctuation in amplitude at different frequencies with a conclusive pattern; All experience a reduction in amplitude of $2.5^\circ \pm 0.5^\circ$ when operating at 0.4 Hz and a reduction of 1 $\pm 0.3^\circ$ at 0.25Hz. Having overshoot was expected, given the results of the encoder verification. However, the $3^\circ$of undershoot seen in figure 45 was not expected; this is a major 10% amplitude error, and as it falls outside the scope of the motion capture accuracy can be believed to be true. As this isn't seen at the motor level, this can attributed to the mechanism.

The full extent of this flaw will need to be studied in further motion capture, but an undershoot of $2.5^\circ \pm 0.5^\circ$ is considerably higher than the $\pm 0.45^\circ$ expected from the stroke mechanism. This is a consistent drop across almost all amplitudes attempted, suggesting this is a constant loss. This could due to the tension in the timing belt not being high enough; slack in the belt could cause erroneous results as the belt goes slack on one side and is taut on the other when changing direction, meaning that the full extent of the motor's rotation would not be translated. However, if this were the case, one would expect this also to be visible at 15°amplitude. The 0.25Hz frequency follows this trend, but the 0.4Hz frequency does not. This could be a user error, or this issue with the belt only becomes apparent at higher amplitudes. No conclusive confirmation can be made with the current data set. Further, more rigorous motion capture is required to understand this issue and whether it is a factor of amplitude.

In the 60 oscillations analysed for each motion shape, there is variation in these as well as seen by the error

bars in figures 44, 45, 46, 47. As expected, the variation in oscillation amplitude increases with an increased frequency; the error bars at the higher frequencies exceed the error from the user-based motion capture, so the impact can be believed to be a good representation of the impact on frequency on the accuracy of the stroke motion.

If, after further analysis, the pulley system continues to produce erroneous results, it would be best to replace it with a two-spur system; this would better translate torque but would require rebuilding and revalidating the stroke mechanism. The future work section of this report further explores this change.

5.2.5 Pitch Amplitude Analysis

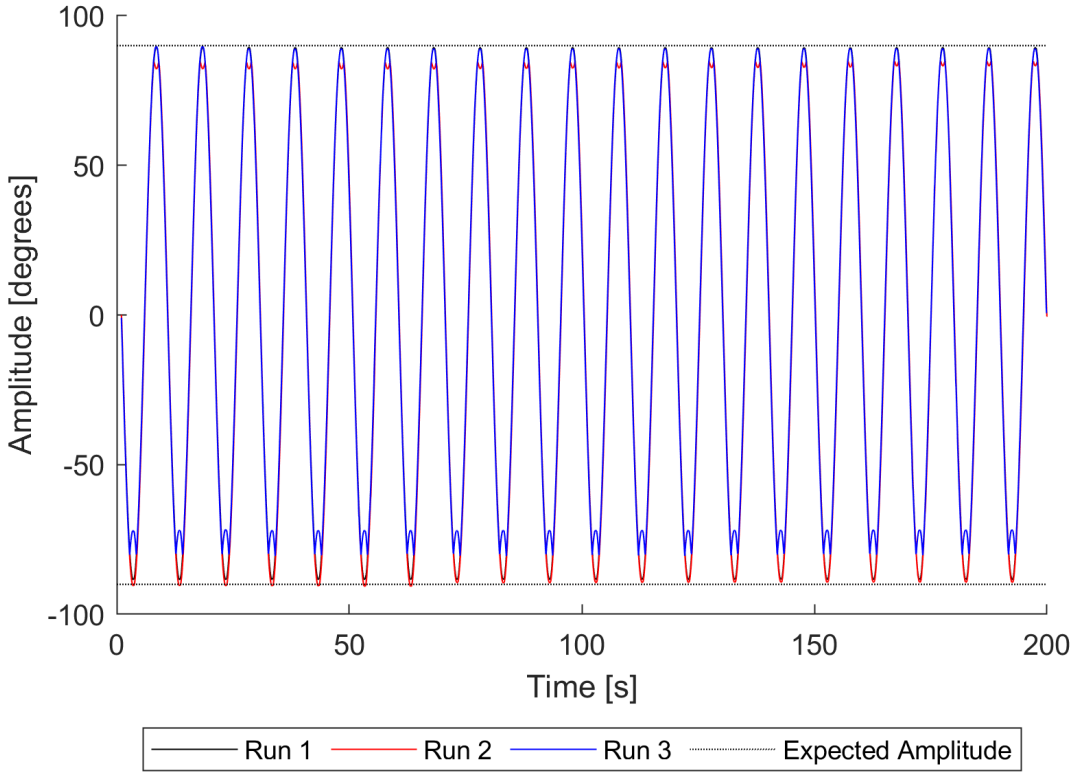


Figure 48. Motion capture results from motion 10 for 20 oscillations

Figure 48 shows the outcome of the motion capture; this figure contains the 3 runs analysed; each run's waveform is comprised of the average of the six lines formed in motion capture, any erroneous lines where the corners weren't traceable throughout the video removed.

Figure 48 clearly depicts a variation in amplitude between the runs, with run 2 and run 3 having significantly different shapes at the peaks.

This, in all probability, is the fault of the motion capture. Where it struggled to keep track of the corners accurately. Run two has a maximum amplitude short of the positive peak amplitude; in run three, the waveform looks to have two peaks at the negative amplitude, coming from the point detaching and attaching itself. As

the motion capture was not robust enough to validate the stroke motion, and in about 20% of pitch motion capture videos analysed, at least one point of contact on the model failed, it is almost definitely the cause of this error. very long sentence

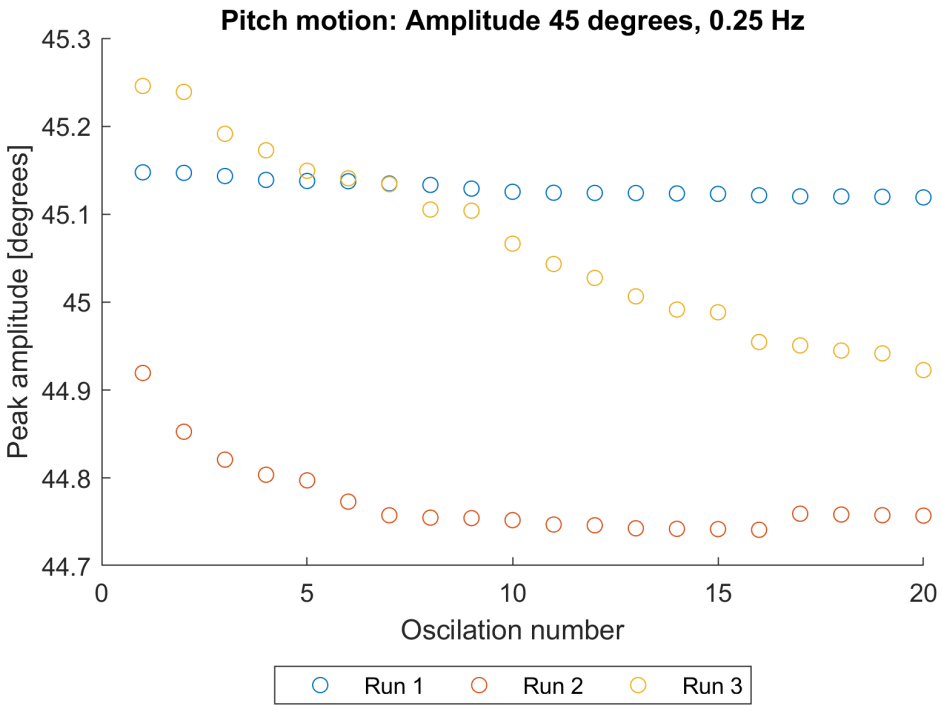


Figure 49. Variation in amplitude seen for 3 runs for motion shape 2

While it failed in some runs, the runs in which it didn't can still be used to show the variation in amplitude occurring. Figure 49 shows the variation in amplitude experienced during motion 2. This shows all values were within 0.3°of their intended target. The motion studies done at 0.1Hz have an accuracy close to that of the bevel gears' 0.2°. Additional errors are likely from the motor program or the motion capture itself.

As the frequency and amplitude increased, the errors associated with them also increased. Motion 12 has larger variations in maximum amplitude, as shown in Figure 50. Run 3 was omitted from its findings as the motion capture struggled to accurately maintain the computer model for this run at the peaks. The error experienced in motion 12 is around ± 1.4 degrees. There also does not seem to be a correlation between the number of oscillations occurring and the error from the peak amplitude. This suggests that the errors do not impact each other or accrue with time. This means the mechanism can be run for high numbers of oscillations without impacting its performance and accuracy.

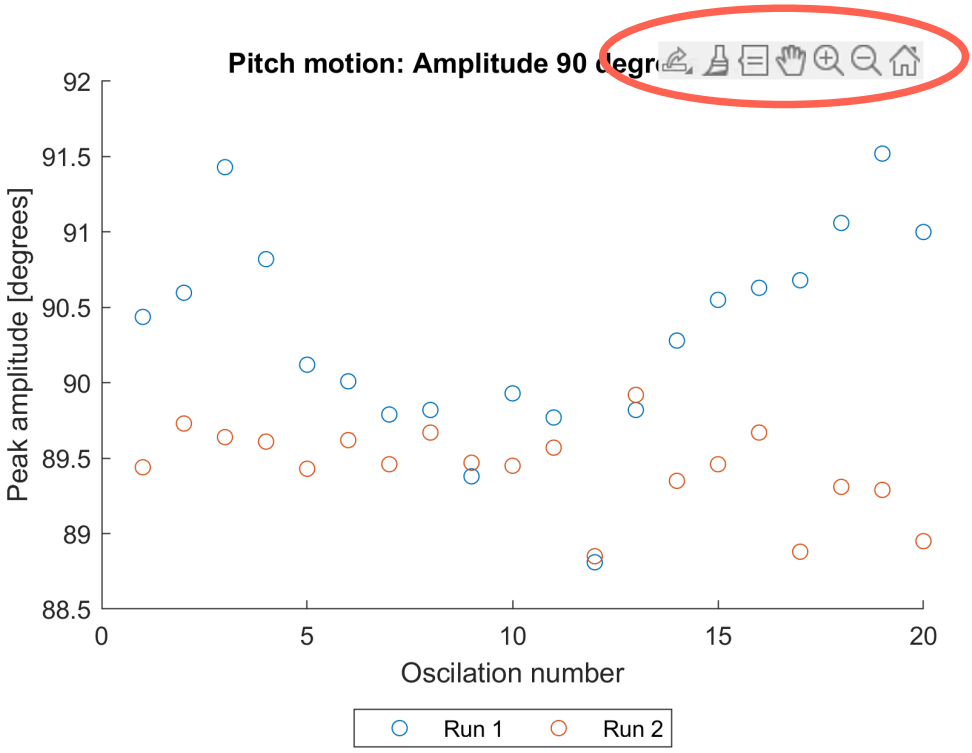


Figure 50. Variation in amplitude seen for 3 runs for motion shape 11

Table 10. Table showing the average amplitude error and the variance around the expected value in degrees

Motion	Average A Error [%]	Variance Error[%] [0.5ex]
45°, 0.1Hz	0.3308	1.1213
60°, 0.1Hz	0.2109	1.3883
75°, 0.1Hz	0.6113	1.7733
90°, 0.1Hz	0.9982	1.2000
45°, 0.25Hz	0.6151	1.5556
60°, 0.25Hz	1.1600	3.600
75°, 0.25Hz	0.4302	1.8400
90°, 0.25Hz	1.1743	2.8333
45°, 0.4Hz	0.5502	2.1556
60°, 0.4Hz	1.1119	3.1500
75°, 0.4Hz	0.7101	1.4346
90°, 0.4Hz	0.7634	3.3453

The errors are shown in table ten 10. The average error seen in the cycle is smaller than the desired error for

the system, which is good. However, the variance error is the error between the highest and lowest oscillations, which is higher and, on average, increases with amplitude and frequency. The difference in error between motion 1 and 12 is the same trend as seen from the motors. however, the variance error is larger than that seen in the encoder by 1% $\pm 0.4\%$. This means that there is an error coming from the motors that is being amplified in the mechanism. Despite this, the average error remained low within all runs. With further, more rigorous motion capture, the likelihood is that the variance in amplitude seen here will be reduced to the levels seen in the encoder.

6. Conclusions

This project aimed to produce an accurate mechanical model that could produce reliable and accurate sinusoidal motion in both pitch and stroke. This goal was met by retrofitting model elements and replacing components with more reliable alternatives, using PID control methods to improve the accuracy of the motor positioning, and validating using encoder and motion capture methods.

The motion capture developed for this project experienced some accuracy and reliability issues that impacted the validity of its results. Further testing will be required to validate the motions analysed in this report more accurately.

In conclusion, the pitch motion is accurate in both code and mechanism. With both the frequency and amplitude errors accounted for, a maximum error of 3.6451% was experienced during motion 12 (45°amplitude and 0.4Hz frequency), and the maximum average error was 1.17% seen in motion 6. The average error is well below the target error set out in the objectives for this mechanism and is consistent with previous mechanical methods [9]. Whilst the maximum error is higher than this report aimed for,2%, it is not a significant increase. The pitch motion gained from this mechanism is continuous and consistent, meaning the system is also reliable within its motion. This is a key requirement for implementing the force-torque sensor as it ensures the safety of equipment whilst the mechanism is in use.

The stroke motion will need additional motion studies to understand further the timing pulley's effects on the amplitude and to confirm that the shape of the motion remains sinusoidal at all frequencies. The stroke motion can be confirmed in its frequency; its amplitude, whilst more inaccurate than is acceptable at $\pm 3°$, is consistent. The mechanism is reliable and won't perform movements beyond its expected motion, just at a reduced amplitude. Methods to negate this have been discussed, and a recommendation based on future motion capture has been made.

With further motion study and the stroke mechanism properly verified, the system can be fitted with a nano-17 force torque sensor. This milestone will allow for findings to be made about the force produced using this honeybee motion in still and later moving water. As the pitch and stroke motion have been verified for higher amplitudes than the honeybee, this allows for the opportunity to optimise the mechanism past the conventional water-treading mode.

Applying this to MAV and UAV technology will improve our ability to monitor and control bodies of water, allowing for more accurate readings in remote places. A dynamic monitoring method will give a better understanding of the effects of pollution and climate change on the waterways surrounding us, allowing for more effective policies and protection methods.

7. Recommendations for Future Work

A timing belt that was not fully taut had a larger impact than expected and could be remedied by using a small gain constant in the code for the stroke motion or by replacing the stroke mechanism with a 2 spur system. Using a 2 spur system would mean that the base plate would have to be re-cut to move motor 2 closer to the main axle, and fitting spur gears to both the main and secondary axle with a gear ratio of 1:4. This would negate the need for a timing belt, reducing the moment placed on the motor, and would help improve accuracy.

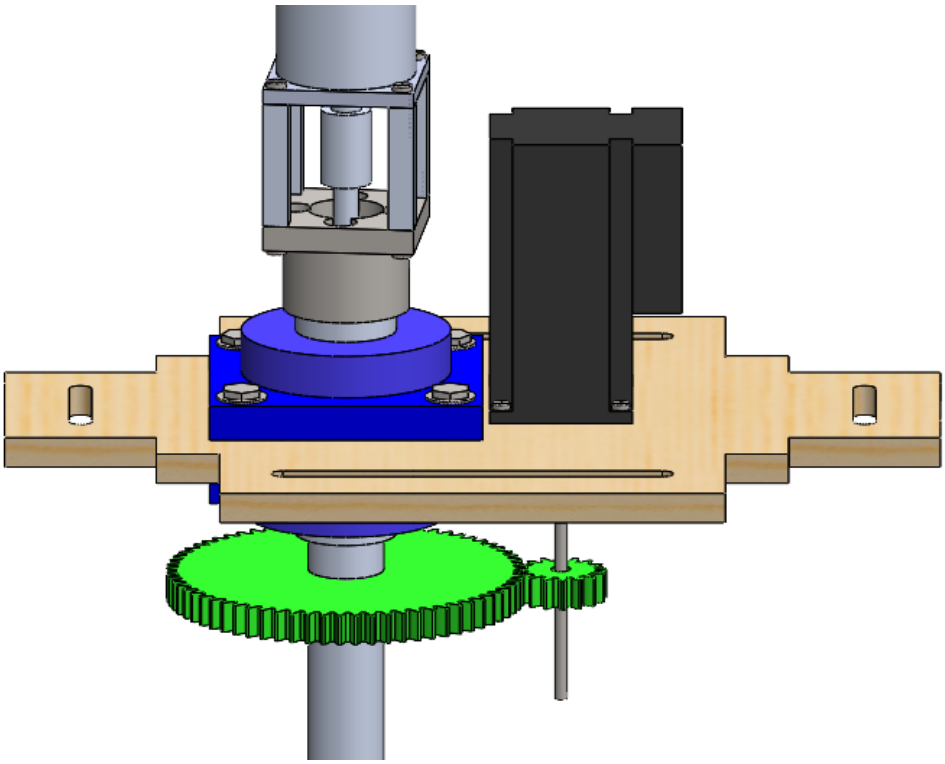


Figure 51. An alternative stroke control method using a 2 spur system

This was designed using the SolidWorks Design Library Toolbox to be laser-cut out of 12mm acrylic sheets. Alternatively, a thinner version could be machined from 6082 T6 Aluminium stocked in the EDMC. However, aluminium has a density of 2700 Kg/m^3 compared to a density of 1180Kg/m^3 for acrylic; a part like this would be heavy and unnecessarily strong for this component as a torque force of 4.228Nm is well within the strength range of acrylic which has a young modulus of 3.2 GPa and would be cheaper and simpler to manufacture. Engineering design drawings for these parts have been included in the appendix.

The experimental runs were done during two consecutive days, with the set-up left submerged during that time as the tank had to be emptied by hand. This meant that the system spent around 24 hours in total in the tank; during this time, the bevel gears developed a layer of rust; this seemed to have little to no impact on the data, as tests from both days correlated with no significant differences. However, for future testing, it would be

recommended it's not left in the tank for extended periods of time or that the bevel gears be swapped out with rustproof alternatives, as this can cause the bevel gears to erode, decreasing accuracy with time.

The next set of experiments will repeat testing with the force toque sensor at the water's surface in still water. This experiment was done in a tank that would have rebound waves that interfered with force-toque readings if surface experiments were repeated. Therefore, It is recommended that this experimentation be done in such a way as to negate this. Surface waves can be damped using foam on the sides and floating on the surface. Alternatively, there is a method that would fully negate surface waves: using a recirculating water tank.

already discussed

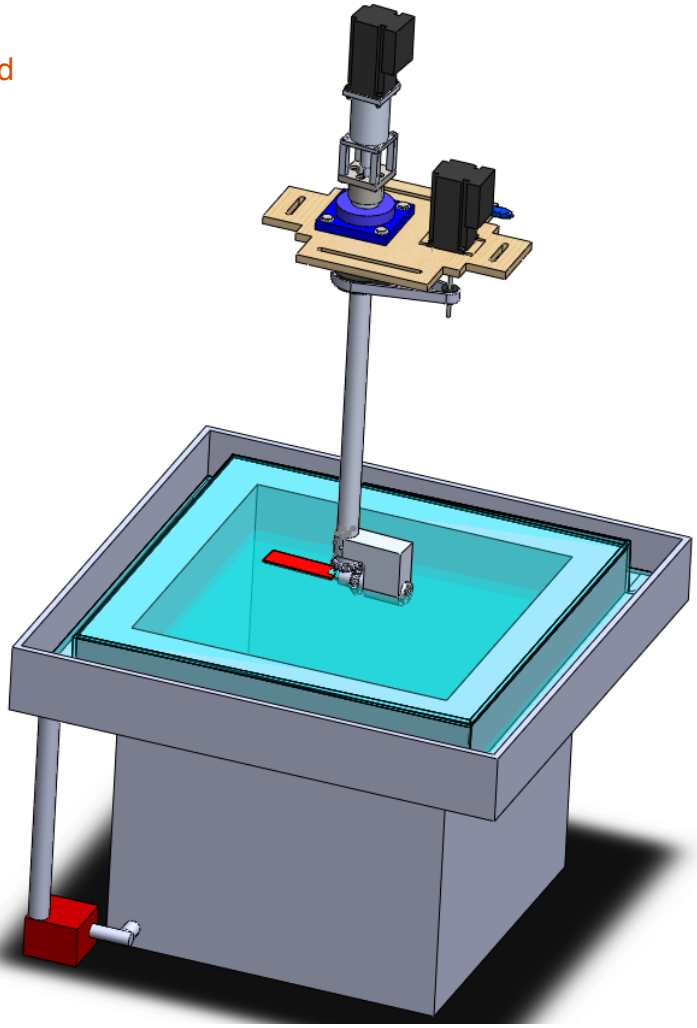


Figure 52. A recirculating water tank to reduce the impact of surface waves on experimentation

Much like an infinity swimming pool, having a tank that is constantly lightly overflowing into a reservoir that is then recirculated back into the tank, the water level would remain the same, as it would drain at the same rate as the pool is filled. If the fluid flow is maintained at a low level, the impact on the force-torque readings can be zeroed out as inertia would be within the model. This means that as ripples on the surface

reach the edge of the tank, they go over the edge instead of reflecting back. Though mildly mechanically difficult, it should completely negate surface waves in the tank and could be used for future experimentation on water-surface locomotion methods.

would this not only affect surface waves and stil cause waves below the surface to have impacts?

References

[1] Chris Roh and Morteza Gharib. Honeybees use their wings for water surface locomotion. *Proceedings of the National Academy of Sciences*, 116(49):24446–24451, 2019.

[2] Dusanpetkovic. Saving life concept. bee saving life at water surface., 2018.

[3] Bharat Sharma Acharya, Mahendra Bhandari, Filippo Bandini, Alonso Pizarro, Matthew Perks, Deepak Raj Joshi, Sheng Wang, Toby Dogwiler, Ram L. Ray, Gehendra Kharel, and et al. Unmanned aerial vehicles in hydrology and water management: Applications, challenges, and perspectives. *Water Resources Research*, 57(11), Nov 2021.

[4] William R. Roderick, Mark R. Cutkosky, and David Lentink. Touchdown to take-off: At the interface of flight and surface locomotion. *Interface Focus*, 7(1):20160094, 2017.

[5] Onur Ozcan, Han Wang, Jonathan D Taylor, and Metin Sitti. Stride ii: A water strider-inspired miniature robot with circular footpads. *International Journal of Advanced Robotic Systems*, 11(6):85, 2014.

[6] Jul 2023.

[7] Anam Tahir, Jari Böling, Mohammad-Hashem Haghbayan, Hannu T. Toivonen, and Juha Plosila. Swarms of unmanned aerial vehicles — a survey. *Journal of Industrial Information Integration*, 16:100106, Dec 2019.

[8] Andre Farinha, Julien Di Tria, Raphael Zufferey, Sophie F. Armanini, and Mirko Kovac. Challenges in control and autonomy of unmanned aerial-aquatic vehicles. *2021 29th Mediterranean Conference on Control and Automation (MED)*, Jun 2021.

[9] Swathi Krishna, Melissa A. Green, and Karen Mulleners. Flowfield and force evolution for a symmetric hovering flat-plate wing. *AIAA Journal*, 56(4):1360–1371, 2018.

[10] John W.M. Bush and David L. Hu. Walking on water: Biolocomotion at the interface. *Annual Review of Fluid Mechanics*, 38(1):339–369, 2006.

[11] J. W. Glasheen and T. A. McMahon. A hydrodynamic model of locomotion in the basilisk lizard. *Nature*, 380(6572):340–342, 1996.

[12] Yelong Zheng, Hongyu Lu, Jile Jiang, Dashuai Tao, Wei Yin, and Yu Tian. Walking of spider on water surface studied from its leg shadows. *Chinese Physics B*, 27(8):084702, 2018.

[13] Haripriya Mukundarajan, Thibaut C. Bardon, Dong Hyun Kim, and Manu Prakash. Surface tension dominates insect flight on fluid interfaces. *Journal of Experimental Biology*, 219(5):752–766, 2016.

[14] David L. Hu, Brian Chan, and John W. Bush. The hydrodynamics of water strider locomotion. *Nature*, 424(6949):663–666, 2003.

[15] James H. Marden, Brigid C. O'Donnell, Michael A. Thomas, and Jesse Y. Bye. Surface-skimming stoneflies and mayflies: The taxonomic and mechanical diversity of two-dimensional aerodynamic locomotion. *Physiological and Biochemical Zoology*, 73(6):751–764, 2000.

[16] P. Freymuth. Thrust generation by an airfoil in hover modes. *Experiments in Fluids*, 9(1–2):17–24, 1990.

[17] Swathi Krishna, Alexander Gehrke, and Karen Mulleners. To tread or not to tread: Comparison between water treading and conventional flapping wing kinematics. *Bioinspiration amp;amp; Biomimetics*, 17(6):066018, 2022.

[18] K Isaac, Jessica Rolwes, and Anthony Colozza. Unsteady flow features of a flapping and pitching wing at low reynolds number. *25th AIAA Aerodynamic Measurement Technology and Ground Testing Conference*, 2006.

[19] K. M. Isaac, Jessica Rolwes, and Anthony Colozza. Aerodynamics of a flapping and pitching wing using simulations and experiments. *AIAA Journal*, 46(6):1505–1515, 2008.

[20] Peter Freymuth. Propulsive vortical signature of plunging and pitching airfoils. *AIAA Journal*, 26(7):881–883, 1988.

[21] K. Gustafson, R. Leben, J. McArthur, and M. Mundt. Qualitative features of high lift hovering dynamics and inertial manifolds. *Theoretical and Computational Fluid Dynamics*, 8(2):89–104, 1996.

[22] S. Sunada, K. Kawachi, A. Matsumoto, and A. Sakaguchi. Unsteady forces on a two-dimensional wing in plunging and pitching motions. *AIAA Journal*, 39(7):1230–1239, 2001.

[23] Robert J. Wood. The first takeoff of a biologically inspired at-scale robotic insect. *IEEE Transactions on Robotics*, 24(2):341–347, 2008.

[24] Sui Zhou, Weiping Zhang, Yang Zou, Xijun Ke, Feng Cui, and Wu Liu. Piezoelectric driven insect-inspired robot with flapping wings capable of skating on the water. *Electronics Letters*, 53(9):579–580, 2017.

[25] Steve H Suhr, Yun Seong Song, Sang Jun Lee, and Metin Sitti. Biologically inspired miniature water strider robot. In *Robotics: Science and Systems*, volume 2005, pages 319–326, 2005.

[26] Je-Sung Koh, Eunjin Yang, Gwang-Pil Jung, Sun-Pill Jung, Jae Hak Son, Sang-Im Lee, Piotr G Jablonski, Robert J Wood, Ho-Young Kim, and Kyu-Jin Cho. Jumping on water: Surface tension–dominated jumping of water striders and robotic insects. *Science*, 349(6247):517–521, 2015.

[27] Yogesh M. Chukewad, Johannes James, Avinash Singh, and Sawyer Fuller. Robofly: An insect-sized robot with simplified fabrication that is capable of flight, ground, and water surface locomotion. *IEEE Transactions on Robotics*, 37(6):2025–2040, 2021.

[28] Christopher B. Freelance. To regulate or not to regulate? the future of animal ethics in experimental research with insects. *Science and Engineering Ethics*, 25(5):1339–1355, 2018.

[29] Alexander Gehrke, Guillaume Guyon-Crozier, and Karen Mulleners. Genetic algorithm based optimization of wing rotation in hover. *Fluids*, 3(3):59, 2018.

[30] ATI Industrial Automation. F/t sensor: Nano17.

[31] F.S. Hover, Haugsdal, and M.S. Triantafyllou. Effect of angle of attack profiles in flapping foil propulsion. *Journal of Fluids and Structures*, 19(1):37–47, 2004.

[32] Field Manar, Albert Medina, and Anya R. Jones. Tip vortex structure and aerodynamic loading on rotating wings in confined spaces. *Experiments in Fluids*, 55(9), 2014.

[33] Xavier Ling, Zhi Quan Leong, Christopher Chin, Michael Woodward, and Jonathan Duffy. Comparisons between seabed and free surface effects on underwater vehicle hydrodynamics. *International Journal of Naval Architecture and Ocean Engineering*, 14:100482, 2022.

[34] Ho-Young Kim, Jun-Seong Lee, Han-Lim Choi, and Jae-Hung Han. Autonomous formation flight of multiple flapping-wing flying vehicles using motion capture system. *Aerospace Science and Technology*, 39:596–604, 2014.

[35] Pranay Seshadri, Moble Benedict, and Inderjit Chopra. A novel mechanism for emulating insect wing kinematics. *Bioinspiration amp;amp; Biomimetics*, 7(3):036017, 2012.

[36] Rakesh P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar. A review of pid control, tuning methods and applications. *International Journal of Dynamics and Control*, 9(2):818–827, Jul 2020.

[37] P. M. Meshram and Rohit G. Kanojiya. Tuning of pid controller using ziegler-nichols method for speed control of dc motor. In *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, pages 117–122, 2012.

[38] Libretexts. 9.3: Pid tuning via classical methods, Mar 2023.

[39] Applied Motion, 2021.

[40] Applied Motion. Stm23s-3re nema 23 integrated drive+motor w/ encode.

Appendix

Main Motor Code

$$t_M = \frac{c}{f} \tag{14}$$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% CLEARING VARIABLES %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This section resets and clear all stored values in the program

clc %clears the command window
clf % clears the figures section
clear all % clear all variables
close all % closes all the figures


%%%%%%%%%%%%%%%%%%%%%%%%%%%% VARIABLES TO CHANGE %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This contains all vaiables tf
sheet_num = 5;
sheet_name = 'Encoder 24.xlsx';

serial_port = 'COM3'; % The serial port on the user's computer

gear_change_pitch = 9; %multiplier by which the gear effects the pitch
gear_change_stroke = 4;%multiplier by which the driving belt impacts stroke

pitch = 0; %degrees - Max pitch looking to achieve
stroke = 90; %degrees - Max stroke looking to achive
frequency = 0.4; %hertz
pitch_acceleration = 1; %increasing this increases the responsivness
stroke_acceleration = 1; %degrees per -----
phase_control = 1; %the pitch follows the stroke - leading trailing, insinc
force_data_rate = 1; %how often to read the force data
number_of_cycles = 10; %number of cycles to run
start_pitch = 0; %expected starting picth at the begginning of the program

%%%%%%%%%%%%%%%%%%%%%%%% INITIALISING THE SERIAL PORT %%%%%%%%%%%%%%%%%%%%%%%%
%This initialises the serial port and sets up communication with the motors
```

```
%%{
disp('Initialising serial port:')
disp(serial_port)
s1 = serialport(serial_port, 9600);
fopen(s1);                    % Open the port
pause(0.25);
configureTerminator(s1, "CR");
s1.Terminator;
%}



%%%%%%%%%%%%%%%%%%%%%%%% MANIPULATING INPUTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This section calculates nessisary values from the inpuyt variables to be
%used later in the program


motor_steps_per_rotation = reading_motor_values(s1,'1EG') ;
writeline(s1,['2EG',num2str(motor_steps_per_rotation)])
%total steps for one rotation
one_rotation_pitch = gear_change_pitch*motor_steps_per_rotation;
one_rotation_stitch = gear_change_stroke*motor_steps_per_rotation;
%adjusting how many steps per degree is needed when motor is used
steps_pitch = pitch*motor_steps_per_rotation*gear_change_pitch/360;
steps_stroke = stroke*motor_steps_per_rotation*gear_change_stroke/360;

%calculating amplitude in terms of motor steps
PITCH_amplitude = pitch*motor_steps_per_rotation*gear_change_pitch/360;
STROKE_amplitude = stroke*motor_steps_per_rotation*gear_change_stroke/360;
max_time = number_of_cycles/frequency;
%%%%%%%%%%%%%%%%%%%%%%%%%%% MOVEMENT PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%this section starts communication with the motors to recieve position
%points
%%{
writeline(s1,'IFD') % returns encoder positions in decimels incread of hex

%Stroke is motor 1
```

```
writeline(s1, ['1AC',num2str(stroke_acceleration)]);
writeline(s1, ['1DE',num2str(stroke_acceleration)]);
writeline(s1,'1VE0.5');
writeline(s1,'1FP0'); %returns the motor to its initial position
writeline(s1,'1EP0'); %resets the encoder postion to this position
writeline(s1,'1SP0'); %checks the reset worked
writeline(s1,'1DI200')
%Pitch is motor 2
writeline(s1, ['2AC',num2str(pitch_acceleration)]);
writeline(s1, ['2DE',num2str(pitch_acceleration)]);
writeline(s1,'2VE0.5');
writeline(s1,'2FP0'); %returns the motor to its initial position
writeline(s1,'2EP0'); %resets the encoder postion to this position
writeline(s1,'2SP0'); %checks the reset worked
writeline(s1,'2DI200')%sets the direction for future jogging motion
%}
%writeline(s1,['2FP',num2str(PITCH_amplitude)]);
correction = 0;
x = input('change pitch? give a positive or negative number of steps to turn ',"s");
while x ~= 'n'
    correction = correction + str2double(x);
    writeline(s1,['2FP',num2str(correction)])
    x = input('change pitch? give a positive or negative nmber of steps to turn ',"s");

end
correction = 0;
x = input('change stroke? give a positive or negative number of steps to turn ',"s");

while x ~= 'n'
    correction = correction + x;
    writeline(s1,['1FP',num2str(correction)])
    x = input('change stroke? give a positive or negative number of steps to turn ',"s");

end
pause(2)

writeline(s1,'1EP0'); %resets the encoder postion to this position
```

```
writeline(s1,'1SP0'); %checks the reset worked
writeline(s1,'2EP0'); %resets the encoder postion to this position
writeline(s1,'2SP0'); %checks the reset worked

disp('set up complete')

pause(2)
disp('moving to initial position')
writeline(s1,['1FP',num2str(round(STROKE_amplitude))])
%writeline(s1,['2FP',num2str(PITCH_amplitude)])

%%%%%%%%%%%%%%%%%%%%%%%%%%%% TIMED MOVEMENT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this method uses position from the encoders and feeds back velocity
% changes with a really high accelleration

tic
while toc < 5
    disp('waiting')
    pause(1)
end
%%{
disp('Motor 1 position is')
disp(reading_motor_values(s1,'1IP'))
disp('Motor 2 position is')
disp(reading_motor_values(s1,'2IP'))



disp('Starting movement')
[actual_pos, error,expected_pos,timey,lead_vel,i] = sin_wave_2(s1,PITCH_amplitude, ...
    frequency,one_rotation_pitch,max_time,STROKE_amplitude,one_rotation_stitch);


%%%%%%%%%%%%%%%%%%% Plotting graphs%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
actual_angle = actual_pos/one_rotation_stitch*360;
expected_angle = expected_pos/one_rotation_stitch*360;
error_angle = error/one_rotation_stitch *360;
```

```
 hold on
scatter(timey,actual_angle(1:length(timey)))
scatter(timey,expected_angle(1:length(timey)))
scatter(timey,error_angle(1:length(timey)))
plot(timey,actual_angle(1:length(timey)))
plot(timey,expected_angle(1:length(timey)))
plot(timey,error_angle(1:length(timey)))
%plot(timey,encoder_v)
hold off;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Writing data to excel%%%%%%%%%%%%%%%%%%%
names_array = {'Actual Position','Error', 'Expected Position','Time', 'Actual Angle',
    'Expected Angle', 'Error Angle'};
writecell(names_array,sheet_name,'Sheet', sheet_num,'Range','B1:H1');

writematrix(actual_pos(1:i),sheet_name,'Sheet', sheet_num,'Range','B2');
writematrix(error(1:i),sheet_name,'Sheet', sheet_num,'Range','C2');
writematrix(expected_pos(1:i),sheet_name,'Sheet', sheet_num,'Range','D2');
writematrix(timey(1:i),sheet_name,'Sheet', sheet_num,'Range','E2');
writematrix(actual_angle(1:i),sheet_name,'Sheet', sheet_num,'Range','F2');
writematrix(expected_angle(1:i),sheet_name,'Sheet', sheet_num,'Range','G2');
writematrix(error_angle(1:i),sheet_name,'Sheet', sheet_num,'Range','H2');

%}

%%%%%%%%%%%%%%%%%%%%%%%% MOTOR COOL DOWN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%returns motor to home position of it is not there and then disengages the
%serial port
disp('Back to zero')
writeline(s1,'1FP0')
writeline(s1,'2FP0')


disp('Closing the serial port')
%clear(s1)
```

```
%delete(s1)
%}


disp('finish')
disp(max(error_angle(30:end)))

%%%%%%%%%%%%%%%%%%%%%% Function Definitions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function readSerialData(src,~)
    data = readline(src);
    disp(data);
end
```

---

sin_wave_2 Function

```
   function [actual_pos_s,error_s,expected_pos_s,timey_s,lead_vel_s,i] =
       sin_wave_2(s1,amplitude_p,freq,motor_steps_per_rotation_p,max_time,amplitude_s,motor_steps_per_rotat
%% Sin_wave_2 controls the movement of both pitch and stroke simultaniously
%% as long as the time reached is below the macx time for n oscilations

i = 1;

%%%%%%%%%%%%%%%  PITCH   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%% starting motor at 0 velocity%%%%%%%%%%
%Motor 2 - Pitch
writeline(s1,'2JA3000') %max acceleration
writeline(s1,'2JD3000')
writeline(s1,'2JS0')%sets initial jogging speed to zero to complete set up
writeline(s1,'2CJ')% starts jogging

%%%%%%%%%%%%%% PID constants%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ku_p = 18;
tu_s = 0.25;
kp_p = 0.45*ku_p;
ki_p = 0.54*ku_p/tu_s;%1.2*kp_p*freq;
```

```
kd_p =0%0.0001;

%%%%%%%%%%%% Setting space to place values into %%%%%%%%%%%%%%%%%%%%%%
error_p = zeros(200000,1);
actual_pos_p = zeros(200000,1);
expected_pos_p = zeros(200000,1);
lead_vel_p = zeros(200000,1);
timey_p = zeros(200000,1);
error_p(i) = 0;

%%%%%%%%%%%%%%%%%% Reading initial values %%%%%%%%%%%%%%%%%%%%
actual_pos_p(i) = reading_motor_values(s1,'2IP'); % reading initial values
expected_pos_p(i) = amplitude_p*cos(0*2*pi*freq); %calculating expected inital position

%calulating initial velocity
lead_vel_p(i) = -amplitude_p*pi*4*freq*sin(0*2*pi*freq)/motor_steps_per_rotation_p;




%%%%%%%%%%%%%%%%%%%%%%%% STROKE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%% starting motor at 0 velocity%%%%%%%%%%%
writeline(s1,'1JA3000')%max acceleration
writeline(s1,'1JD3000')
writeline(s1,'1JS0')
writeline(s1,'1CJ')
error_s = zeros(2000,1); % setting space to place values into
actual_pos_s = zeros(2000,1);
expected_pos_s = zeros(2000,1);
lead_vel_s = zeros(2000,1);
timey_s = zeros(2000,1);

%%%%%%%%%%%%%% PID constants%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ku_s =9;% 1.2;
tu_s = 0.25;
kp_s = 0.45*ku_s;
```

```
ki_s =0.54*ku_s/tu_s;
kd_s =0%0.0001*ku_s*tu_s;

%%%%%%%%%%%% Setting space to place values into %%%%%%%%%%%%%%%%%%%%
error_s(i) = 0;
actual_pos_s(i) = reading_motor_values(s1,'1IP');
expected_pos_s(i) = amplitude_s*cos(0*2*pi*freq);
lead_vel_s(i) = -amplitude_s*pi*4*freq*sin(0*2*pi*freq)/motor_steps_per_rotation_s;

integral_term = 0;
pause(2)



%%%%%%%%%%%%%%%% MOVEMENT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic % starts the timer
while toc < max_time %max time is the total time to complete all the oscilations
    i = i+1;
   [error_p(i),lead_vel_p(i),actual_pos_p(i),expected_pos_p(i),timey_p(i)] =
       sin_wave_4(s1,amplitude_p,freq,motor_steps_per_rotation_p,2,error_p(i-1),timey_p(i-1),kp_p,ki_p,kd_p
   [error_s(i),lead_vel_s(i),actual_pos_s(i),expected_pos_s(i),timey_s(i)] =
       sin_wave_3(s1,amplitude_s,freq,motor_steps_per_rotation_s,1,error_s(i-1),timey_s(i-1),kp_s,ki_s,kd_s
end

%%%%%%%%%%% Stopping motion%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
writeline(s1,'2SJ')% stops the motion
writeline(s1,'1SJ')%stops the motion
%Pitch = table(error_p(1:i),lead_vel_p(1:i),actual_pos_p(1:i),expected_pos_p(1:i));
%Stroke = table(error_s(1:i),lead_vel_s(1:i),actual_pos_s(1:i),expected_pos_s(1:i));
%disp(Pitch)
end
```

Functions sine_wave_3 and sine_wave_4 are identical except for a 180 degree phase shift in both the the expected_pos functions and the lead_vel functions that can be changed to introduce lead and lag.

```
function [error1,lead_vel,actual_pos,expected_pos,timey1] =
    sin_wave_3(s1,amplitude,freq,motor_steps_per_rotation,motor,error2,timey2,kp,ki,kd)
```

```
% calls position, determins the error from the expected position and then
% computes the corrected velocity to stay on the correct sine wave
    timey1 = toc;
    actual_pos = reading_motor_values(s1,[num2str(motor),'IP']); %reads the actual position in
        steps
    expected_pos = amplitude*cos(timey1*2*pi*freq);%calculates the expected position from the
        time
    error1 = expected_pos-actual_pos;% error between the two values
    lead_vel = -amplitude*pi*4*freq*sin(timey1*2*pi*freq)/motor_steps_per_rotation; %
        calculates the expected velocity

    %Computing PID values
    prop = error1*kp; % proportional control in position in steps
    inter = ki*(error1+error2)/2*(timey1-timey2);% intergrative control in position in steps
    deer = (error1-error2)/(timey1-timey2)*kd;% differential control in position in steps

    %velocity being fed to the motors with the pid being tranlated into velocity in rps
    feed_vel = lead_vel + (((prop +deer + inter)/(timey1-timey2))/motor_steps_per_rotation);
    writeline(s1,[num2str(motor),'CS',num2str(feed_vel)])
end
```

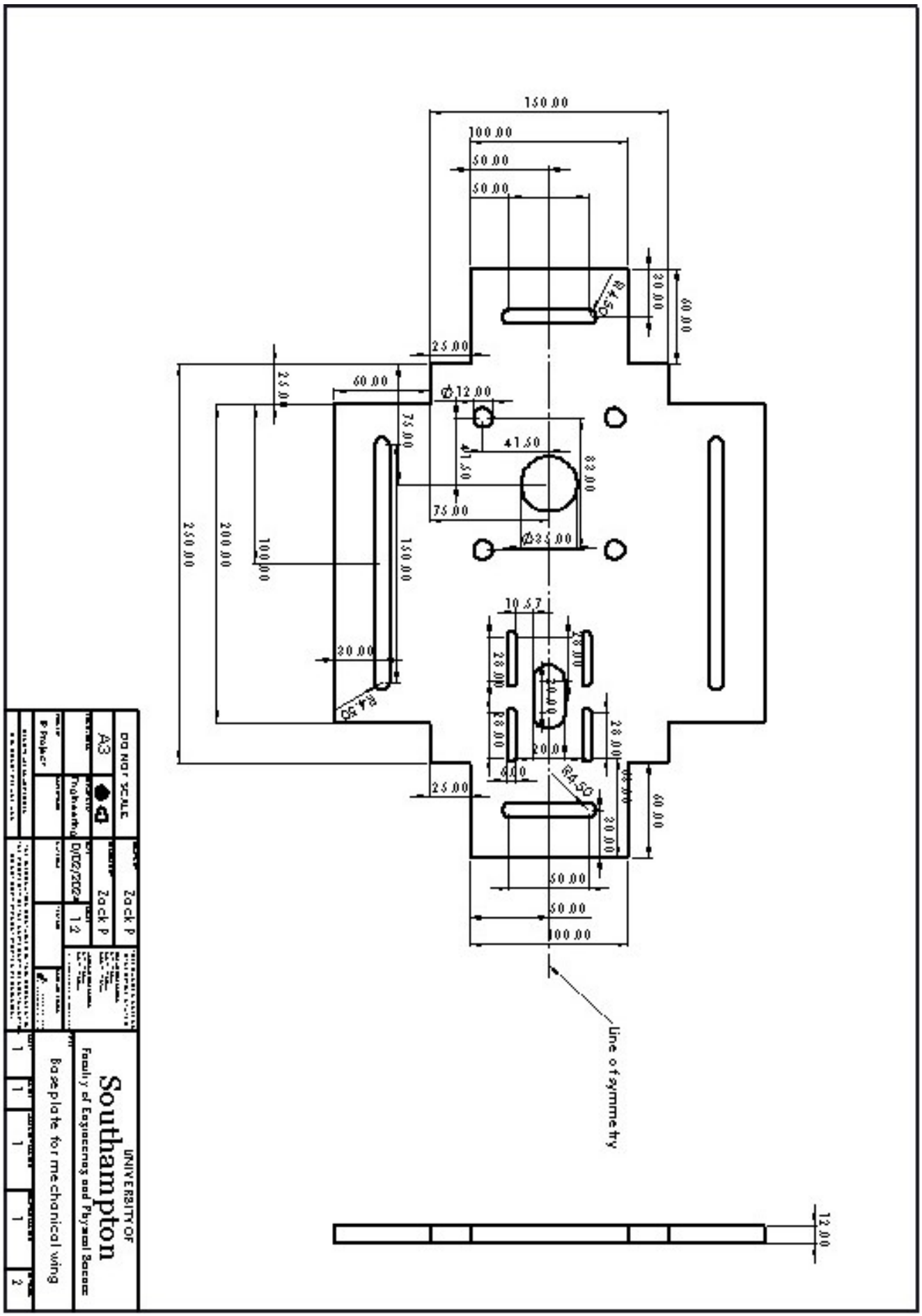


Figure 53. New baseboard design

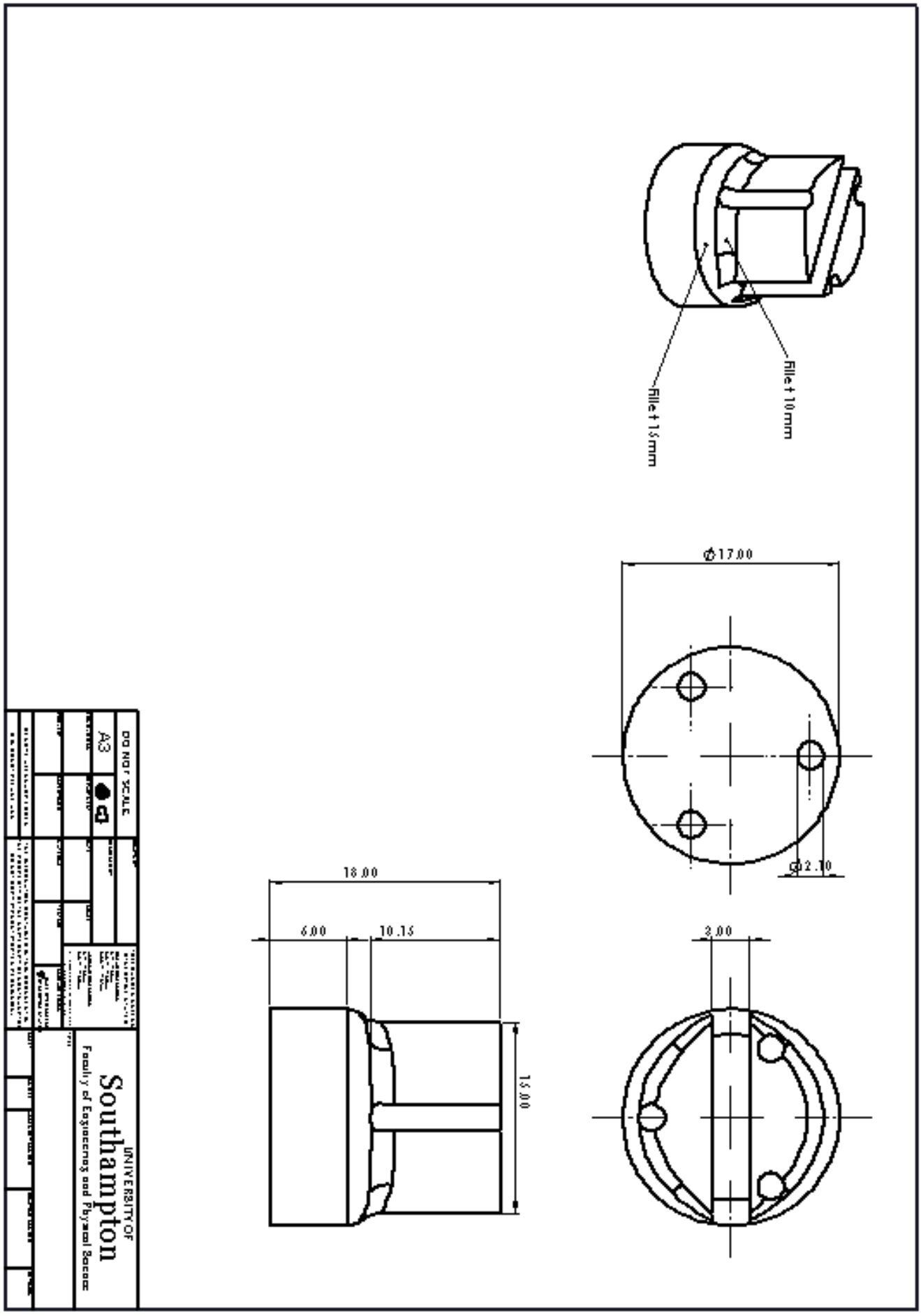


Figure 54. New baseboard design