UNIVERSITY OF CALIFORNIA SAN DIEGO

Optimization of the FastCTD Ocean Profiler from Onboard Data and Computational Modeling

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Mechanical and Aerospace Engineering

by

Vikram Rajagopalan

2019

The Thesis of Vikram Rajagopalan is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2019

DEDICATION

I would like to dedicate this work to my family and friends. Their unconditional love and support helped me get through the most challenging few months of my life, and I am forever grateful. I would also like to thank my advisor for giving me the opportunity to work with him, and for giving me the time I needed at the beginning of the project to overcome my personal challenges.

TABLE OF CONTENTS

List of Abbreviations

FastCTD – Fast Conductivity Temperature Depth

ADCP – Acoustic Doppler Current Profiler

CTD – Conductivity Temperature Depth

EOS-80 – Equation Of State 80

TEOS-10 – Thermal Equation Of State 10

SBE – Sea Bird Electronics Inc.

IMU – Inertial Measurement Unit

GPS – Global Positioning System

PSD – Power Spectral Density

CFD – Computational Fluid Dynamics

OF – OpenFOAM

RANS – Reynolds Averaged Navier Stokes

SST – Shear Stress Transport

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Drew Lucas for his support as the chair of my committee. Through countless meetings and discussions, his guidance has always helped me make progress when I was lost.

I would also like to acknowledge the members of the Wirewalker team at Scripps Institution of Oceanography, all of whom have helped me with different aspects of this project, but especially San Nguyen, who helped with the Data Analysis chapter. I would also like to acknowledge Hieu Pham, of the MAE department, whose advice on the Computational Fluid Dynamic portion of this project was helpful.

ABSTRACT OF THE THESIS

Optimization of the FastCTD Ocean Profiler from Onboard Data

and Computational Modeling

by

Vikram Rajagopalan

Master of Science in Mechanical and Aerospace Engineering

University of California San Diego, 2019

Professor Andrew J. Lucas, Chair

The FastCTD is a shipboard system consisting of a high-speed winch and profiling body developed at Scripps Institution of Oceanography. The instrument package onboard the profiling body ("fish") can be used to measure temperature, salinity, and pressure rapidly and accurately. The winch and telemetry system allows the fish to be profiled vertically through the ocean at very high speeds (4-6 m/s). The combination enables characterization of ocean phenomena with large vertical scales that change rapidly in time, such as oceanic internal waves. Central to its performance is the ability to accurately acquire data while traveling at high velocities, without any interruptions. The fin configuration induces a rotation in a predetermined direction, curtailing vertical unpredictable and unstable flight of the FastCTD, but in doing so, results in excessive rotations. This interferes with the data telemetry through the tow cable, since the conductors can break due to extreme twisting. Evaluating the effect of the fins on the flight dynamics of the fish may lead to the development of a more robust design, and better performance.

The optimization is carried out in two steps. The first is a thorough analysis of data recorded by the onboard inertial measurement unit, in order to quantitatively and qualitatively understand the instability. The second is a first order computational fluid dynamic study of the fins and control surfaces, and their influence with varying external conditions. The simulations reveal that a steeper angle of descent is useful in mitigating the degree of rotation. Two straightforward design changes involving the positioning of the fins are proposed, and their effectiveness of influencing the angle of attack are studied, as a first step toward possible future designs of the FastCTD.

Introduction

Oceanographic Measurements

Physical Oceanography is the study of the physical attributes of the oceans such as surface tides, internal waves, and currents. Central to the evaluation of such phenomena is the analysis of the density of water in space and time. Density is an important attribute, as pressure gradients due to density differences drive ocean currents and the circulation of heat [1]. Density in the ocean is not directly measured, but derived by measuring the salinity, temperature, and pressure, and plugging them into the equation of state. The measurement of ocean density, salinity, temperature, and their fluctuations in space and time elucidates phenomena such as oceanic internal waves, oceanic turbulence, and the dissipation of meteorological events.

Internal waves are buoyancy waves caused by variations in density. They are large amplitude gravity waves, which propagate at the interface between a layer of warm water overlying a layer of cooler water. Roughly 40m below the surface, there is an abrupt change in both the water density and temperature, which are called the pycnocline and thermocline respectively. The pycnocline is the interface between two fluids of different densities, and disturbances that travel along pycnoclines are called internal waves [2]. Internal waves play a crucial role in transferring heat, energy, and momentum in the ocean, which means they are important in the fields of global climate modeling, sediment and nutrient mixing, and offshore-structural mechanics [3]. Internal waves have large vertical scales (hundreds of meters) that change rapidly in time (tens of minutes), and their characterization requires the ability to profile density with spatial and temporal scales comparable to that of the disturbances themselves. The evolution and propagation of internal waves are not well understood and research investigations are ongoing.

The irregular and variable motion of water is referred to as turbulence, although there is no simple and unambiguous definition of the term. Ocean turbulence provides a mechanism for energy dissipation and momentum transfer across time and space scales, and it also provides a means by which tracers such as heat, salt, and nutrients are irreversibly mixed [4]. Hence, turbulence in the ocean affects both the local ecosystem and global climate. Locally, turbulence controls nutrient budgets, heat content, and sea surface temperature and globally, turbulence affects the sequestration of carbon, and deep convection [5]. The study of ocean turbulence is most commonly carried out by using an ADCP to evaluate velocities at different spatial locations, which leads to the turbulent kinetic energy present in the water [4]. However, turbulence can also be studied by evaluating the variations in density, but is not common due to limitations in the profiling speed and accuracy of the available instruments.

Meteorology and oceanography are closely related, and often, meteorological phenomena are categorized, examined, and fore-casted by carrying out measurements in the ocean [6]. Characterization of the intensity, spatial scales and temporal scales of ocean storms are achieved by evaluating salinity and temperature variations across the geographical location of interest. There is however, a transient interval associated with these events, within which general turbulent mixing in the ocean has not completely dissipated their signature. This transient interval is of the order of tens of minutes. In comparison, traditional CTD instruments take several hours to profile a kilometer of water. Hence, the ability to study these events is limited, once again, by the capabilities of the measuring instrument.

There is a need for systems that can accurately profile properties such as temperature, salinity, and density rapidly enough to study internal waves, turbulence, and many other such phenomena. The FastCTD system developed at Scripps Institution of Oceanography delivers these capabilities, and is introduced in the following chapter.

The FastCTD Ocean Profiler

Introduction

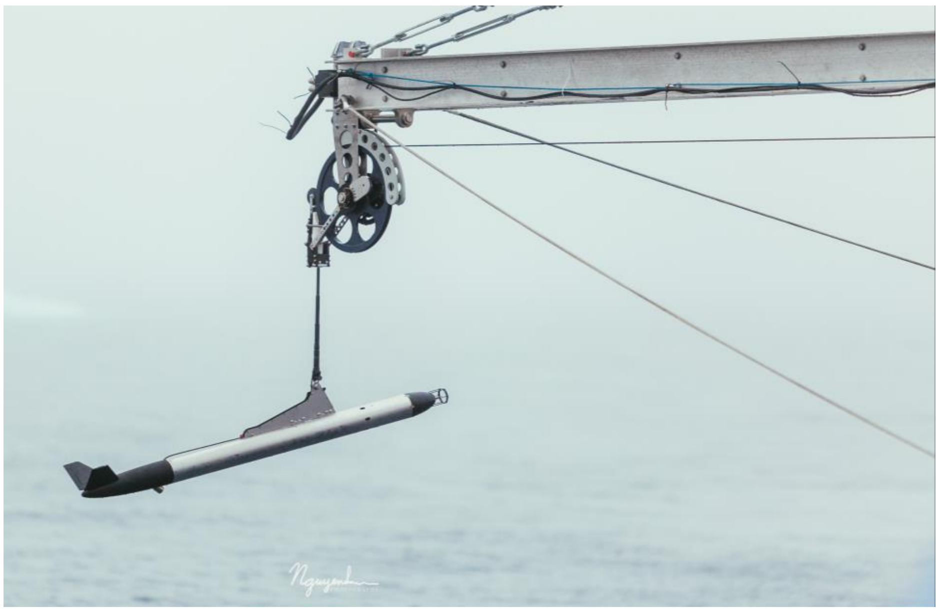


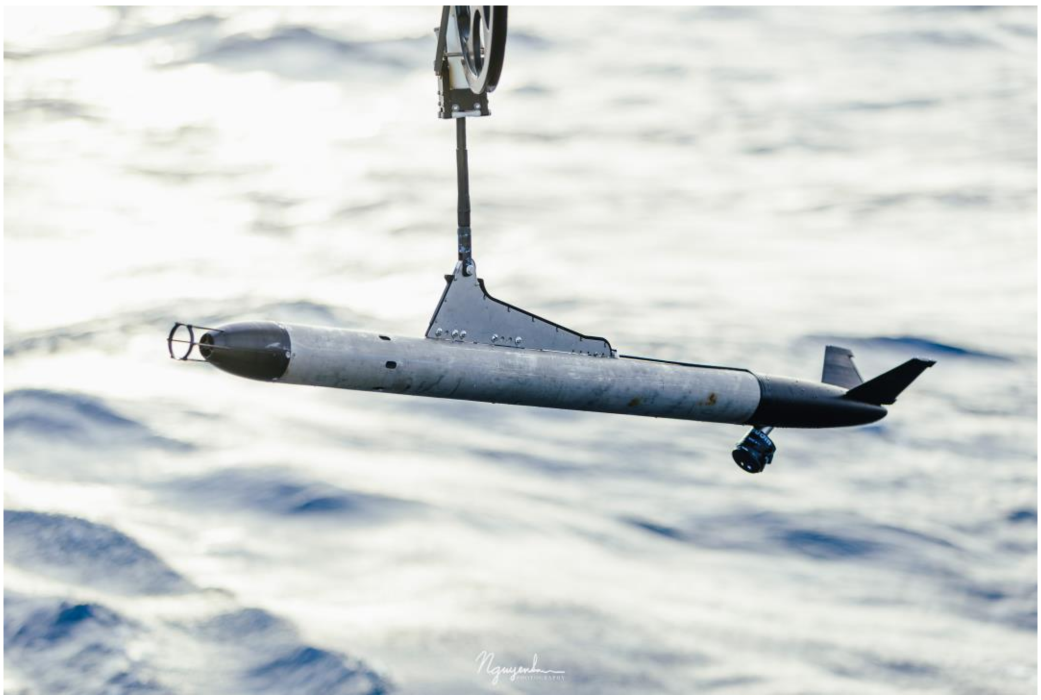
Figure 1 – FastCTD ship-board system

The FastCTD (Figure 1) is a shipboard system developed at Scripps Institution of Oceanography (SIO). It consists of a high-speed winch, a boom and pulley assembly, and a profiling body, referred to as the "fish". The system has a controller box, using which it can be operated. The instrument package onboard the fish can be used to measure temperature, salinity, and pressure rapidly and accurately. The winch and telemetry system, coupled with the streamlined design and narrow cross section of the fish, allows it to be profiled vertically through the ocean at very high speeds. This enables the study of the structure of ocean phenomena with large vertical scales that change rapidly in time, such as oceanic internal waves.

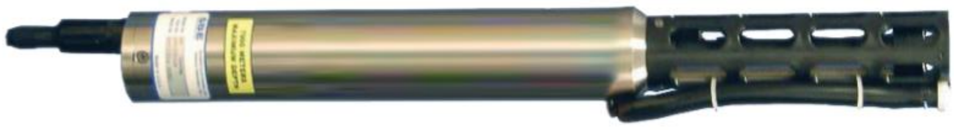
The entire system has been designed in such a way as to maximize the velocity of the fish through the water. To that end, a high-speed winch and a customized boom and pulley assembly have been designed at SIO specifically for the FastCTD. The fish is linked to the winch through a tow cable, which also serves as a data cable by which it communicates to the shipboard controller box. The fish has no propulsion mechanism, and simply free-falls on a down-cast and is winched up on an up-cast. The winch, which is powered by a 480V direct-drive electric motor, has the ability to spool out cable at upwards of 6 m/s on a down-cast. It works in conjunction with the pulley system that maintains a constant tension on the winch to enable disentangled spooling. On the up-casts, the winch pulls the fish up through the tow cable, and is smooth enough to prevent damage to the conductors within the tow cable. As a result, the fish can be profiled vertically through the water at speeds of 4-6 m/s (traditional CTD systems have a velocity of 0.5 m/s). It can profile the upper kilometer of the sea, back and forth, in 12 minutes, and the upper 2 kilometers in 30 minutes. A drawback of this profiling speed, is a decrease in resolution. The system has a resolution of approximately 2 meters, which is sufficient to study most phenomena accurately.

The key aspect of the FastCTD package, and what makes it so effective, is its robustness and ruggedness. Data acquired by the FastCTD is only useful if it can quickly and accurately survey a large section of water. As a result, the fish itself needs to repeatedly free-fall and be winched at high velocities while being subjected to chaotic ocean waves and currents, without any interruption to its data acquisition. The winch, boom, and pulley arrangement need to spool out and draw in several kilometers of cable for every profile, and be able to perform hundreds of profiles continuously. This is partially the reason for which the FastCTD contains its own specific winch and boom arrangement. A lot of care is taken to ensure that all components of the system can satisfactorily endure the fatigue they are subjected to during deployment.

Parts and Physical Aspects of the fish



2(a)



2(b)

Figure 2 – (a) FastCTD fish and (b) SBE 49 CTD module [7]

The fish (Figure 2(a)) is armed with an arsenal of sensors. The first, and most important, is the SBE-49 (Figure 2(b)), an integrated CTD sensor intended for use as a modular component in underwater vehicles. It requires external DC power, and a mechanism for onboard data logging or telemetering to the towing vehicle. In the case of the fish, the data is telemetered back to the ship through the towing cable. The SBE-49 module uses the CTD data in TEOS-10 (Thermal Equation Of State) to directly provide the density. Apart from the SBE-49, the fish is equipped with a Yost-Labs 9-axis IMU to evaluate its flight through the water, and an altimeter. The altimeter indicates the distance to the bottom of the ocean, and is used to control bottom depth of the profile through the shipboard controller box.

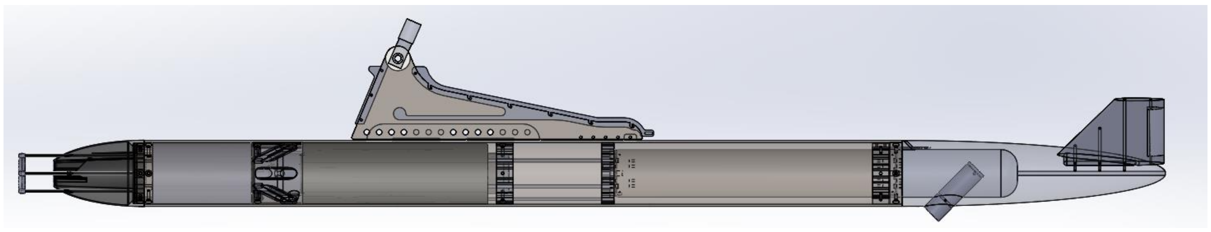


Figure 3 – Fish housing

The housing of the fish (Figure 3) consists of three main parts, a nose cone, a main body, and a tail cone. The nose cone contains the hydrofoil, flow channels, and pump, which are responsible for guiding the water past the sensors of the CTD module. They ensure that the water being sampled represents the surrounding water, as the fish moves vertically at high speeds, which would otherwise result in a stagnation point at the nose. The body houses all the instruments such as the SBE-49, the telemetry, as well as the dorsal fin assembly. The dorsal fin has an articulating arm through which data is passed to the tow cable, which needs to be able to withstand the forces sustained during winching. The tail cone contains the altimeter, and serves as the base for the two tailfins. The tailfins are rigidly attached to the tail cone, but each have control surfaces that can be manipulated by hand.

Problem Statement

Deployment Limitation – Rotation

The key feature of the FastCTD package is its ability to continuously, repeatedly, profile in harsh conditions with no interruptions due to its design or operational reasons. The cost associated with shipping, deployment, operation and manpower in obtaining data using the FastCTD, predictably, is very high. Hence, the reliability of the system is of utmost importance. In order to achieve this reliability, the flight of the fish through the water must be stable. To that end, two tailfins, each with a control surface, are fitted on the tail cone of the fish. They are responsible, primarily, for generating enough lift to drive the nose of the fish down. The nose of the fish must be pointing towards the direction of travel in order to guide the incoming water over the sensors of the CTD module, and to move through the water as fast as possible. The tailfins are also responsible for preventing undue rotations and maintaining the heading of the fish.

Field experiments have revealed that a neutral position of the control surfaces results in an unpredictable rotation in the fish, which could damage the instrument package within the body or interrupt the data sampling, both of which have high cost penalties associated with them. In order to curtail unpredictable rotation, both control surfaces are angled toward the same direction in order to induce rotation in a pre-determined direction (seen in Figure 4). This approach is advantageous as it ensures safe flight and quality data can be obtained, but disadvantageous as it results in the accumulation of rotations of the fish. Excessive rotation of the fish, in itself, does not affect the experiment in anyway. However, since the coupling between the fish and tow cable does not allow for independent rotation, rotation of the fish results in twisting of the cable. This limits the number of continuous profiles that can be carried out, due to the conductors within the tow cable, which are responsible for the telemetry of data. Excessive rotation can break the conductors at an unknown point in the cable, and many kilometers of the cable may come to be discarded. The cable costs approximately $5 per meter.

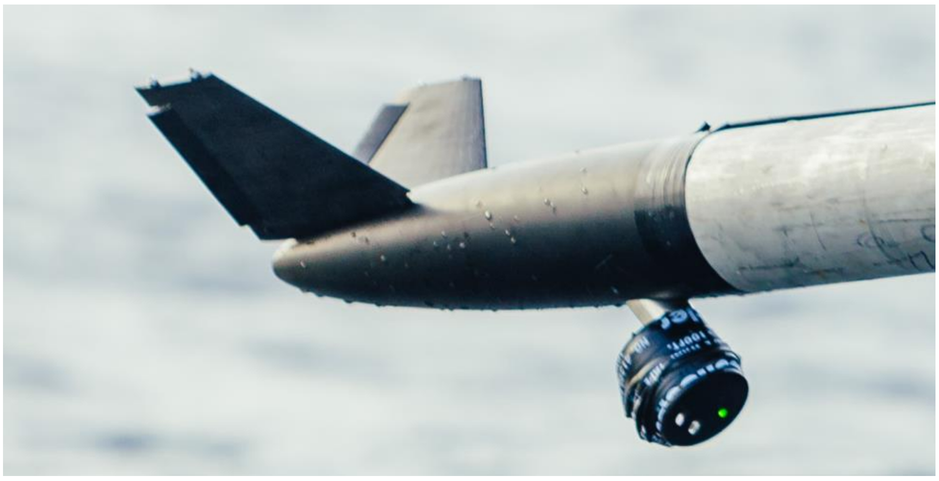


Figure 4 – Control surfaces angled in same direction

As shown later, in Section 1.5, the fish (and thereby the cable), undergo approximately 30 rotations roundtrip on a one kilometer long profile. The cable, from field observations, has been observed to deteriorate in performance beyond a few thousand rotations per kilometer length. This means that every 100 profiles or so, the fish needs to be completely removed from the water and have the control surfaces reversed to unwind the cable. This represents a serious practical disadvantage in the deployment of the FastCTD system, which this work aims to address.

Optimization Approach

The occurrence of excessive rotations in the fish during deployment is addressed by an evaluation of the fins and the control surfaces, and their effect on the flight dynamics of the fish. An optimization of the design of these components could lead to a more robust fish, with an improved maximum number of profiles per round of deployment. This optimization is approached in two steps. First, data from the onboard inertial measurement unit is analyzed in order to qualitatively and quantitatively understand the nature of the rotations occurring. Initial objectives in terms of design changes are developed with the results obtained from the data analysis. Second, a first order computational fluid dynamic analysis is carried out to understand the forces and moments caused by the fins and control surfaces, for varying external conditions. Based on these results, two preliminary design changes are also simulated as a first step toward possible future designs of the FastCTD.

Chapter 1 - Data Analysis

1.1 - On-board Inertial Measurement Unit

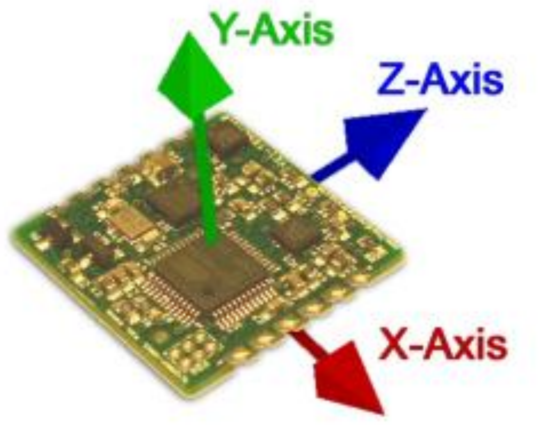


Figure 5 – Yost Labs Inertial Measurement
Unit [8]

The objective of this section of the analysis is to obtain a qualitative and quantitative understanding of the rotations occurring during deployment of the fish using data from the several sensors that the fish is equipped with. One of these sensors, is the Inertial Measurement Unit (IMU) mounted in the pressure case within the fish. This IMU, developed by Yost Labs, consists of a tri-axial gyroscope, an accelerometer and a compass, in conjunction with advanced processing and on-board quaternion based filtering algorithms to determine orientation relative to an absolute reference in real time. This data can be used to acquire information of the rotations such as direction, frequency, down-cast and up-cast variations and axes of rotation.

The axes of measurement of the IMU are as depicted in Figure 5. A peculiar characteristic of this IMU is that it follows a left-hand rule coordinate system, not the traditional right-hand rule system. However, this IMU boasts several strengths such as best in-class form factor and weight, fast sampling and filter rates, and high customizability, among many other features, at a competitive price of $62.5 [8]. These attributes make it a suitable choice for this application. Given below in Table 1 are the technical specifications of this IMU.

Table 1 – IMU Technical Specifications [8]

Sensor	Scale	Resolution	Sensitivity
Accelerometer	±8g	14 bit	0.00024-0.00096g/digit
Gyro-meter	±250°/sec	16 bit	0.00833°/sec/digit
Compass	±1.3Ga	12 bit	0.73mGa/digit

The mounting of the IMU in the fish is not a straightforward procedure. The IMU is hard mounted along with other electronics within the pressure case, and thus the orientation with respect to the pressure case can be specified. However, the pressure case itself is mounted into the main body of the fish, which can be done in multiple orientations. As a result, the orientation of the IMU with respect to the fish itself is only quasi-predetermined. The orientation of Y-axis, which is mounted pointing from the nose to the tail of the fish in the pressure case, is unaltered by the assembly process, however, the orientation of the X and Z axes are not specified. Thus we are only aware of the transverse plane in which both the X and Z axes lie, as shown in Figure 6 (This aspect along with the left-hand rule peculiarity of the IMU are corrected in the following sections).

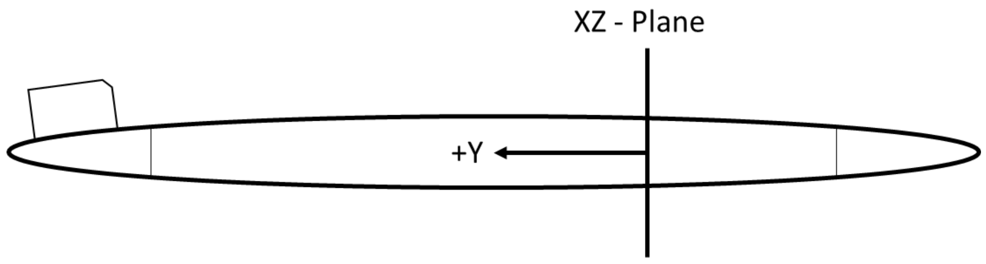


Figure 6 – IMU Coordinate System in fish

1.2 - Pressure and Conductivity Data

The CTD module in the fish provides several types of data such as pressure, conductivity, salinity and temperature. For the purpose of this analysis, the pressure and the conductivity data prove to be very useful. The pressure provides an almost exact estimate of the instantaneous depth of the fish, which can also be differentiated with respect to time to obtain the vertical velocity. The conductivity data, provides an exact reference for points in the experimental data during which the fish is out of the water. As indicated in the problem statement section, the fish needs to frequently be completely removed from the water, to reverse to control surfaces, which will unwind the tow cable on subsequent profiles. At these points, the fish is suspended at a constant angle and direction off the back of the ship. These points in the dataset are pivotal to deciphering the information obtained from the IMU.

1.3 - The dataset

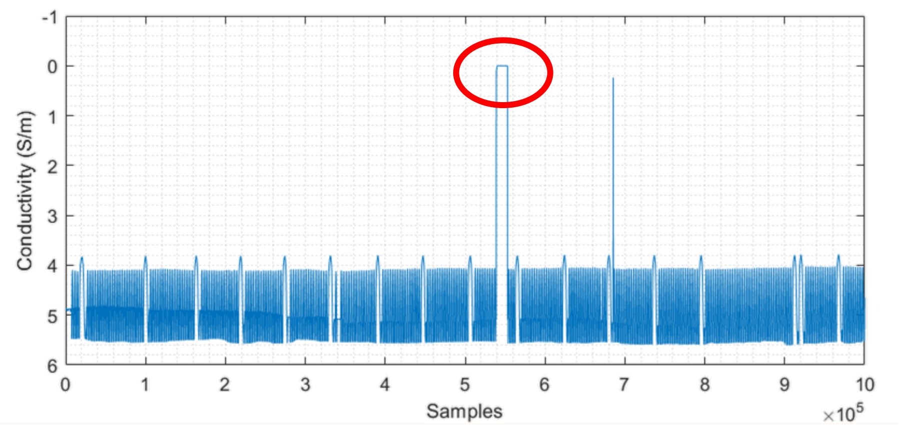
The final dataset, which only includes data that is useful and pertinent to the analysis, are the following:

• X,Y,Z axes of accelerometer	- g
• X,Y,Z axes of gyro-meter	- °/sec
• X,Y,Z axes of compass	- Ga
• Pressure data	- dBar
• Conductivity data	- S/m
• Time	- sec

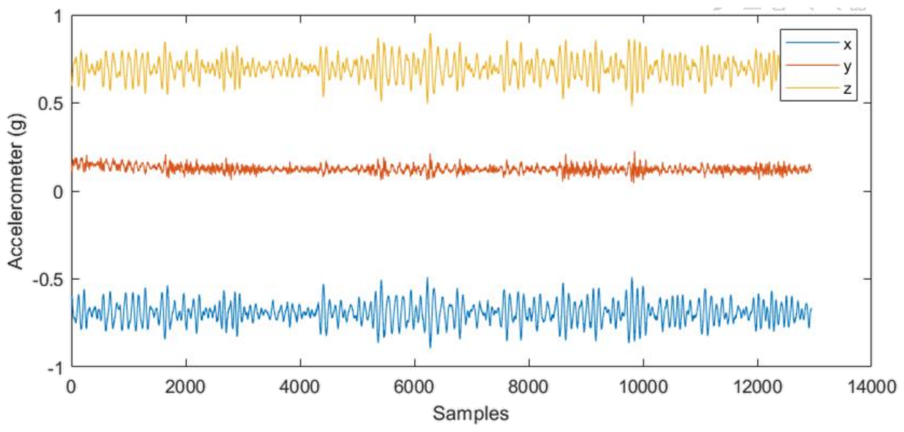
1.4 - Coordinate System Rotations

1.4.1 - Rotation to Base Coordinate System

In order to make physical sense of plots obtained from the IMU data, it greatly helps to rotate the coordinate system to a more sensible mode, such as Z - vertical, and X - horizontal in a right-hand rule coordinate system, with respect to the body of the fish. This requires rotation of the X and Z axes, and flipping of the direction of the Y axis. The angle of rotation of the X and Z axes is obtained from points in the experiment where the fish is winched out of the water and "parked" off the back of the ship. The important point here is that the fish was suspended vertically (i.e. the two tail fins were symmetric with the vertical, the dorsal fin was perfectly in-line with the vertical). These points of the experiment can be found by plotting a graph of the conductivity, as the conductivity of air is zero (Figure 7(a) shows a plot conductivity as measured by the fish). Since the only force experienced by the accelerometer at this point is gravity, the angles of the axes can be found by analyzing the X, Y and Z axes of the accelerometer. The accelerometer when the fish is parked is shown below in Figure 7(b).



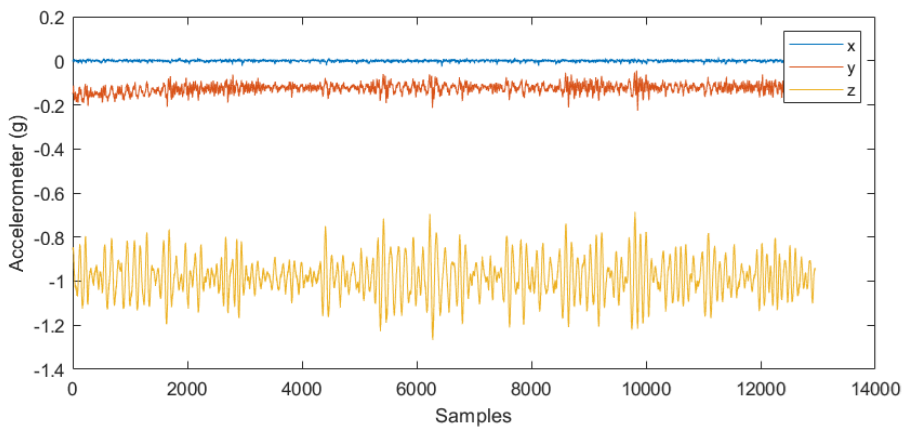
7(a)



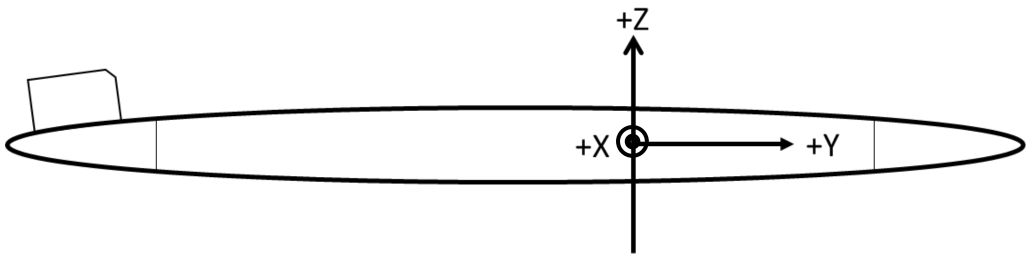
7(b)

Figure 7 – (a) Parked fish conductivity data and (b) Parked fish accelerometer data

From the accelerometer graph, it is observed that the position of the fish when it is suspended is with its nose slightly above the horizontal (as the Y component is positive and small), and the remaining component of gravity is divided between the X and Z axes (the noisy behavior of the data is associated with the motion of the ship). The X and Z axes will also be offset by the same angle as the Y axis from the vertical plane. Accounting for this pitch in the fish, the positions of the X and Z axes with respect to gravity are found using cosine relations, and are subsequently rotated in order to establish a new coordinate system for the IMU with respect to the fish, where X is horizontal and Z is vertical. The Y coordinate readings from the IMU are also multiplied by negative one, to convert from a left-hand rule to a traditional right-hand rule system. This modified, intuitive system helps in making sense of the data. Figure 8(a) shows the new accelerometer data and a sketch of the new coordinate system. Now it can clearly be seen that the acceleration due to gravity is only reflected in the Y and Z axis, since the X axis is horizontal when the fish is parked. As the nose of the fish points marginally upward, opposite to gravity, it has a small, negative component (which was previously positive), and the majority of gravity is captured by the Z axis (also negative in value, as the Z axis points upward, opposite to gravity).



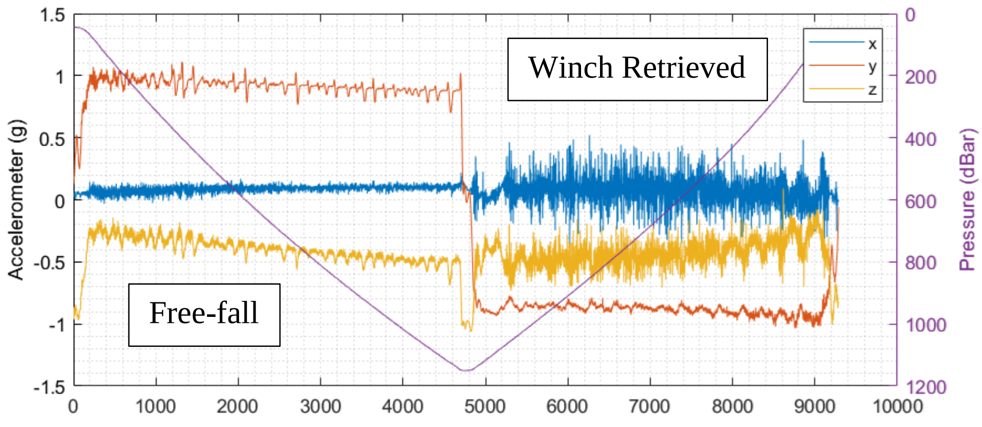
8(a)



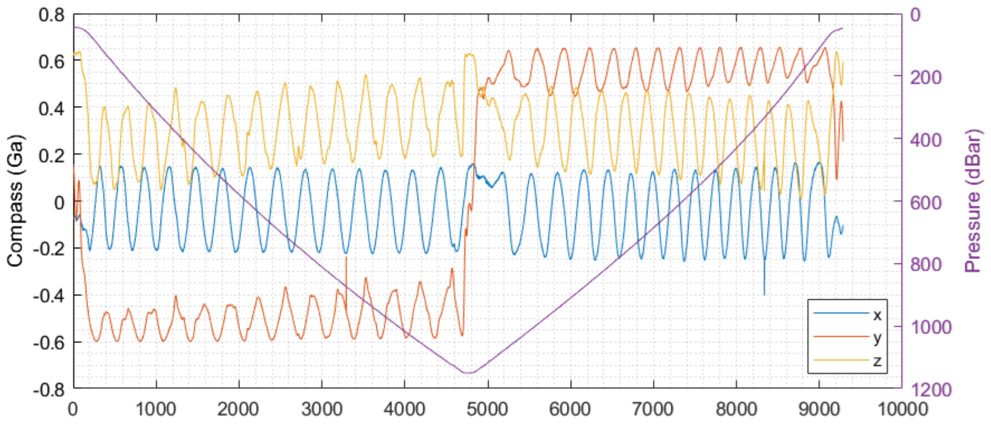
8(b)

Figure 8 – (a) Rotated parked fish accelerometer data and (b) Base coordinate system sketch

1.4.2 - Rotation to Horizontal Coordinate System



9(a)



9(b)

Figure 9 – (a) Accelerometer data and (b) compass data for one profile

Now that the coordinate system has been appropriately adjusted, graphs of the compass and accelerometer data become more interpretable. It is important to note that, unless otherwise specified, all the following figures of dive profiles of the fish correspond to a particular configuration of the control surfaces on the fin. As mentioned in the introduction, the control surfaces are periodically flipped to the opposite direction during the experiment to undo the twisting of the tow cable. In Figure 9 are two graphs depicting each axis of the accelerometer and compass along with pressure for one particular dive. The accelerometer and compass data scale is represented by the left-side Y axis, and the pressure data scale is represented on the right-side Y axis, a trend followed on multiple graphs in this work. During the downcast, where the fish is free-falling, it can be seen from the accelerometer graph that gravity acts only on the Y and Z axes, and not on the X axis. From the constant, near-zero value of X-axis of the accelerometer, it is clear that there is no rotation about the Y axis occurring, which is referred to as *rolling*. However, on the compass graph, periodic fluctuations can be seen on all three axes. This indicates that the only instability occurring is rotation about a vertical axis, which is referred to as *twisting*. It is unclear from these graphs whether the fish is revolving in a helical manner or simply rotating about one vertical axis as it descends. Also observable from the accelerometer plot is that with depth, the component of gravity through the Y axis decreases and the component through the Z axis increases. Physically, this represents a decreased angle with respect to the horizontal. Angles with the horizontal, are referred to as angles of attack or angles of descent in this work. An angle such that the nose of the fish is below the horizontal and the tail above the horizontal is considered to be a negative angle of attack, or a positive angle of descent. The steady increase in the angle of attack (or decrease in the angle of descent) through a downcast is expected as the drag that the fish experiences from the tow cable will increase with depth, as more cable is spooled out into the water. A graph of the variation of angle of attack across one dive can be seen in Figure 10. Note that the angle with the horizontal throughout the down-cast is approximately 60°.

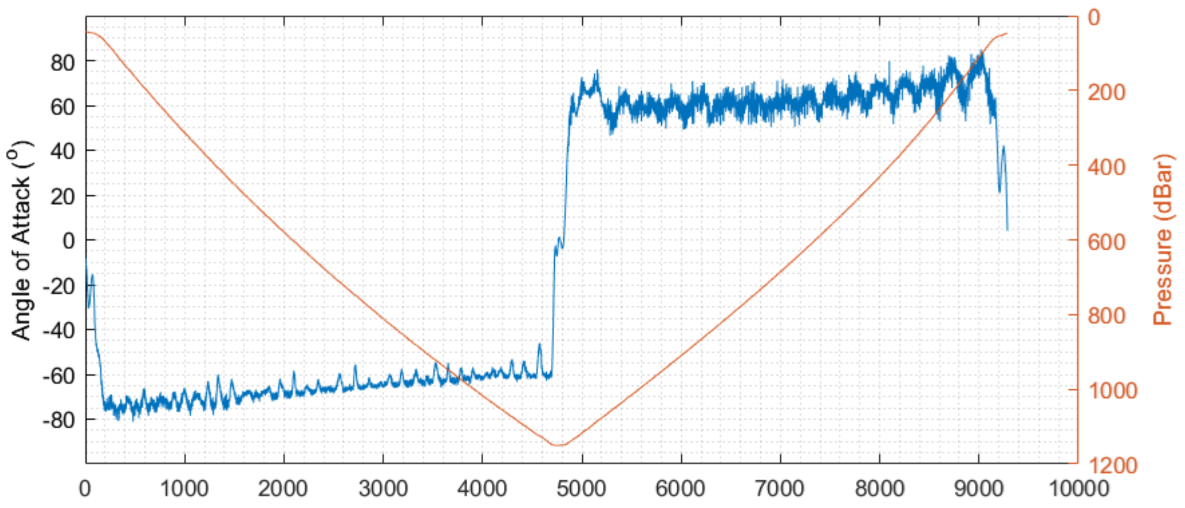


Figure 10 – Angle of attack variation

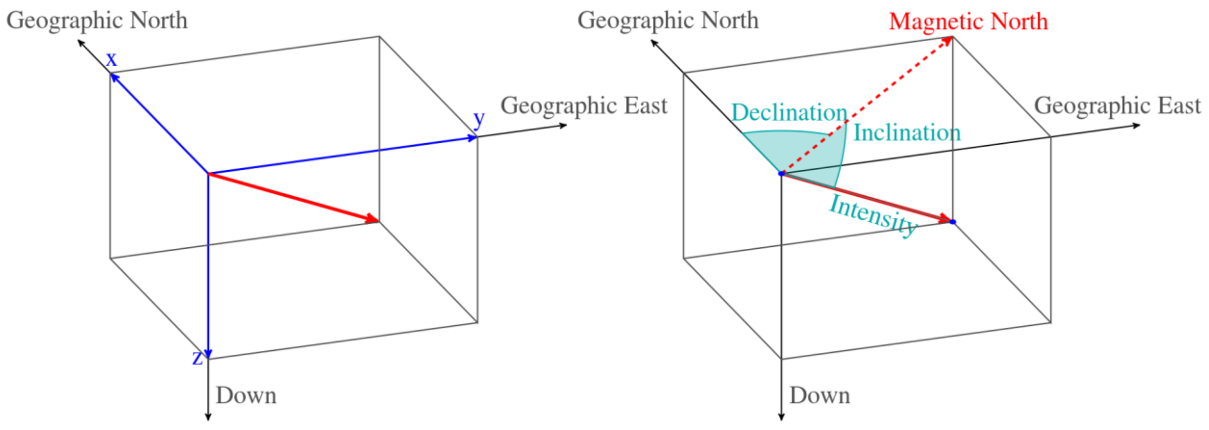


Figure 11 – Components of Earth's magnetic field

Figure 11 depicts the components of the Earth's magnetic field and their nomenclature. The magnetic inclination (the angle that Earth's magnetic flux lines make with the horizontal plane) at the location that these measurements were carried out is 71°. However, it should be noted that the experiment was in the southern hemisphere, where the magnetic field lines are outward from the earth's surface. Each axis of the compass captures both the vertical and the horizontal component of magnetic field. As the accelerometer graph confirms that there is no rolling and the only rotation occurring is twisting about the vertical, fluctuations in the compass profiles are due to changes in the angle between the compass axes and the direction of the horizontal component of magnetic field. The vertical component does not affect the profiles, it only results in a constant offset.

In order to better understand this mode of rotation of the fish, it is useful to perform another rotation on the coordinate system. On the down-casts, it can be assumed that the only force on the fish is gravity. The fish is subject to tension from the cable, which, as can be seen from Figure 10 depicting the angle of attack, results in the fish becoming more horizontal with depth. However, in comparison to gravity, this cable tension is negligible. (Although this assumption is not valid for the up-casts, it is still useful in analyzing the flight of the fish). The Z axis can then be rotated to the opposite direction of gravity for each time-step. This would create a new coordinate system, with Z always vertical, Y pointing in the direction of the nose in the horizontal plane, and X pointing perpendicular to the nose in the horizontal plane. With such a coordinate system, the fluctuations in the magnetic field in the horizontal plane during the rotation of the fish will be captured only in the X and Y axis, and the direction of rotation can also be identified.

This rotation is performed by using a tan inverse relation to find the angle between the Z axis and the gravity vector for every time-step, followed by a cross product between a unit vector pointing towards positive Z and the gravity vector. The purpose of the cross product is to find an arbitrary unit vector about which the rotation can be performed. The matrix for rotation of a vector by a specified angle about an arbitrary vector is given by Rodrigues' Rotation Formula [9], as follows.

$$\theta = tan^{-1}\left(\left|\left|\boldsymbol{u} \times \boldsymbol{v}\right|\right|, \boldsymbol{u} \cdot \boldsymbol{v}\right)$$

$$R = I + (\sin\theta) + (1 - \cos\theta)K^2$$

Where: $K = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}$

And, $k = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix}$

R denotes the rotation matrix, *K* is called the *cross-product matrix*, *k* is the arbitrary vector about which the rotation is performed, and θ is the counter-clockwise angle of rotation. Following this transformation of the coordinate system, the compass data in Figure 9(b) is plotted again in Figure 12.

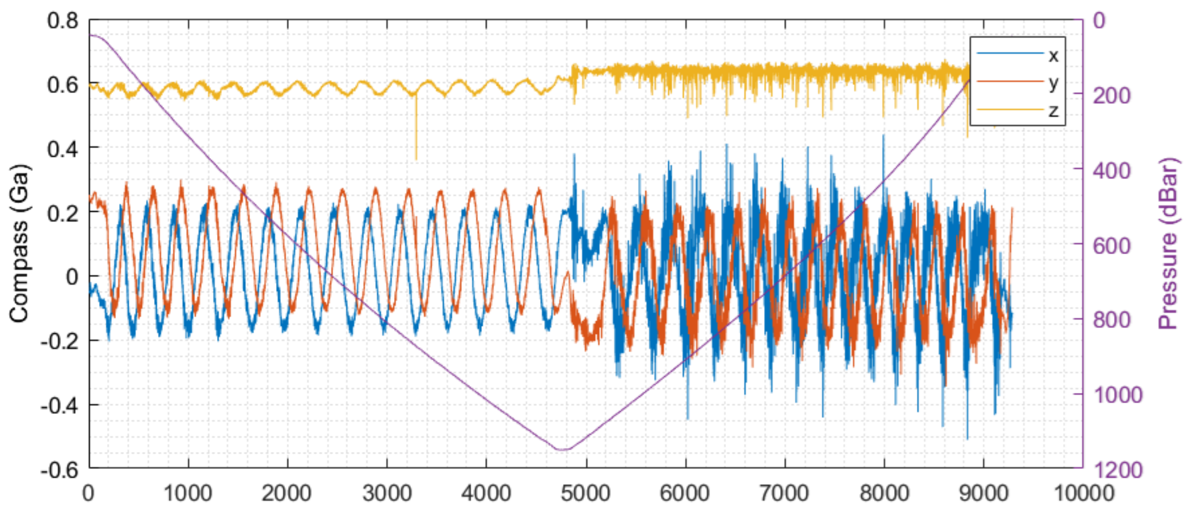


Figure 12 – Horizontal coordinate system compass data

This plot of the compass shows the periodic variation of the X and Y axes of the IMU, with an amplitude from approximately +0.2 Ga to -0.2Ga. The horizontal component of Earth's magnetic field at this particular location is also 0.2 Ga [10]. However, as a result of the inherent sensitivity of the IMU, the data is very noisy. In order to interpret this data more easily, it is useful to apply a high and low-pass filter to remove extraneous content.

1.5 - Spin Counter

A useful utility to arise from the rotation of the IMU data to a horizontal coordinate system, is that the rotated gyro-meter readings can be used to count the number of twists occurring on down-casts and up-casts, individual profiles, or even across a series of profiles. This helps to understand the rate of twisting and can quantitatively relate the manipulation of the control surfaces to the rate of rotation of the fish to the twisting of the tow cable. Shown in Figure 13 is a plot of gyro-meter readings over one particular profile.

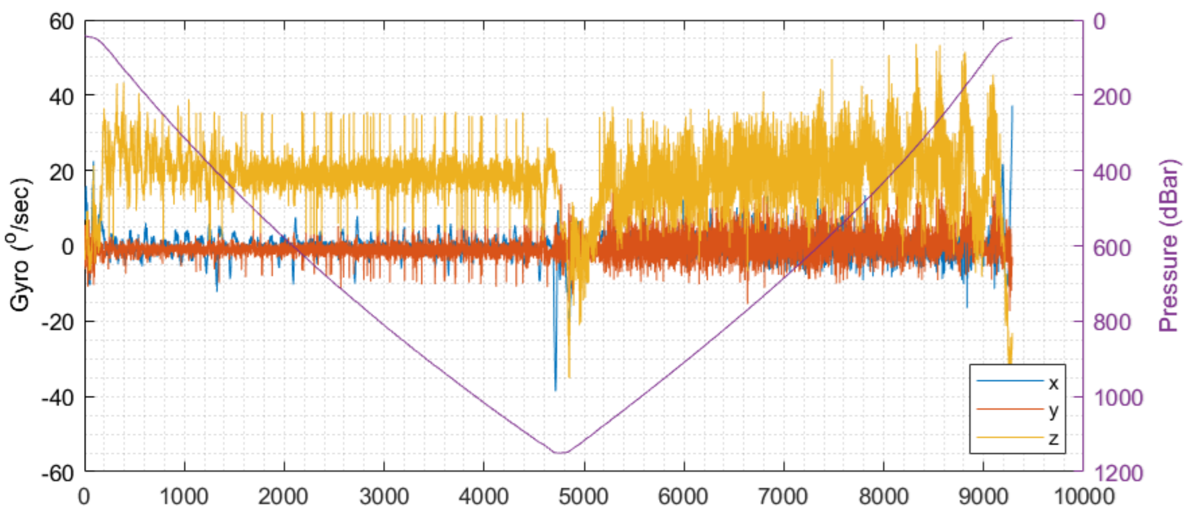


Figure 13 – Gyro-meter data for one profile

The method of calculating the number of spins from the gyro-meter data after it has been rotated to the horizontal frame of reference is straightforward. For the profile shown in Figure 13, the Z-axis gyro-meter values are summed up, multiplied by the mean time-step (simple form of integrating to find the angular displacement across that time interval), and divided by 2π to obtain the number of rotations. The calculated number of rotations for the profile under consideration was 30.23. In this manner, the number of rotations of the fish, and the hence the towing cable, can be found. This utility will be included in the monitoring lab software on the controller box, to have a real-time estimate of the rate of rotation of the fish, and the rotations on the tow cable during deployment.

1.6 - High-pass/Low-pass Butterworth Filter

In order to suitably filter the IMU data, we need to know the upper and lower cut-off frequencies associated with the rotation of the fish. The peak-to-peak distance on the compass X axis profile on Figure 12 is approximately 350 samples on the X axis of the graph. Given that the data was sampled at 16Hz, this gives a time period of rotation of about 20sec, or 0.05Hz. This is the mean frequency of the rotation of the fish. To find the upper and lower cut-off frequencies, it helps to perform a spectral analysis on the compass data. We know that there should be an energy peak around the 0.05Hz frequency, and an upper and lower cut-off frequency can be selected from the power spectral density distribution with frequency.

To compute the power spectral density (PSD) with respect to frequency, Matlab's *pwelch* function has been used. It is an overlapped segment averaging estimator, and directly returns the PSD of an input signal [11]. Shown in Figure 14 is a PSD of the compass X and Z axis.

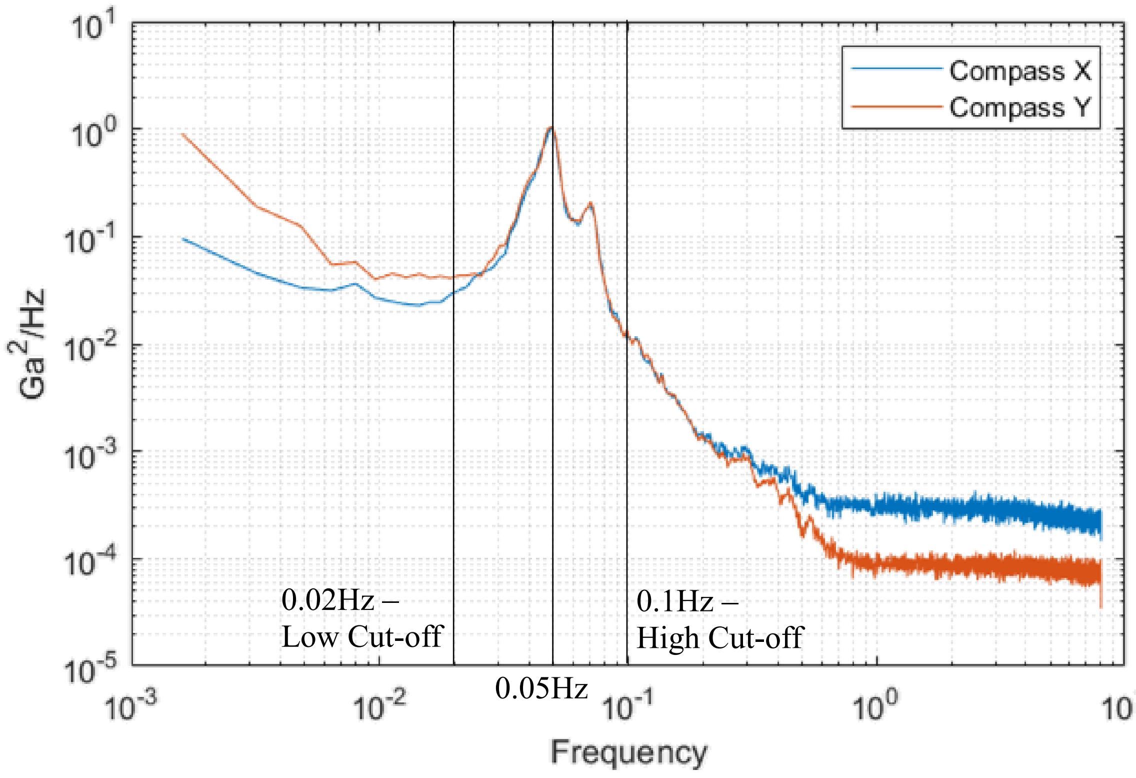
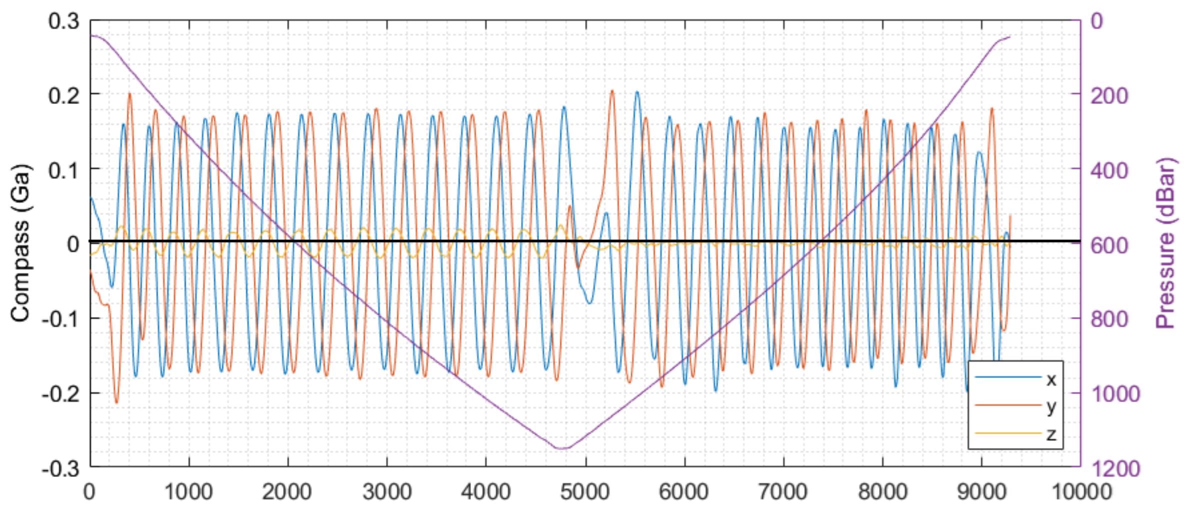


Figure 14 – PSD of rotation phenomena

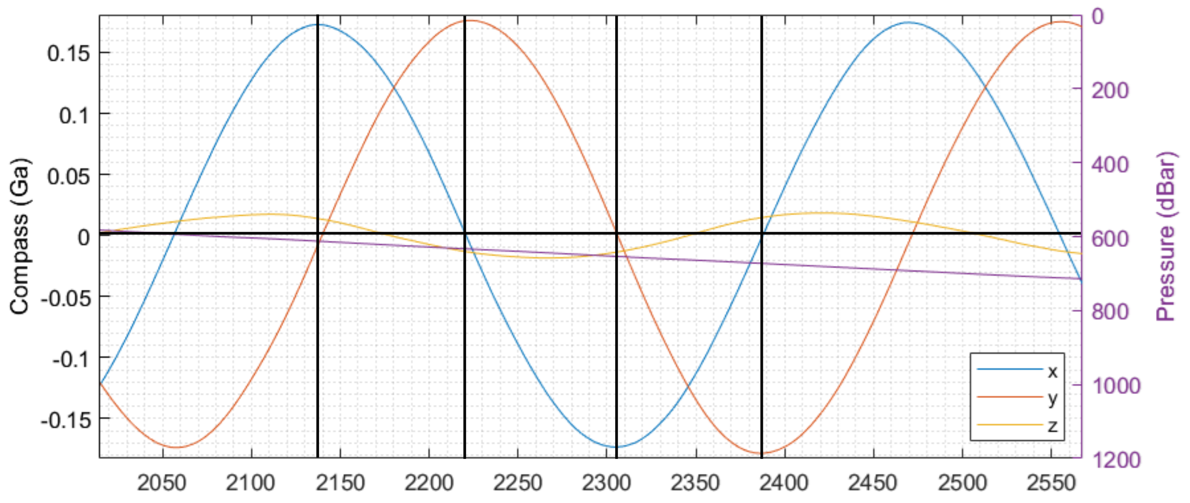
As inferred from Figure 12, there is a peak at a frequency of 0.05Hz. From the PSD plot, a lower cut-off frequency of 0.02Hz and upper cut-off frequency of 0.1 Hz has been selected, providing some cushion between the peak and the cut-off frequencies. Using this, a high-pass and a low-pass Butterworth filter was constructed and the compass data was filtered, and re-plotted.

The variation in power between the compass X and Y axes in the higher frequencies (from 1Hz to 10Hz) is due to high frequency oscillations occurring on the up-casts. These oscillations, as seen in the up-cast of the accelerometer profiles in Figure 9(a), are captured on the X and Z axes but not on the Y axis of the IMU, hence the lower high-frequency power of the Y axis component in the PSD.

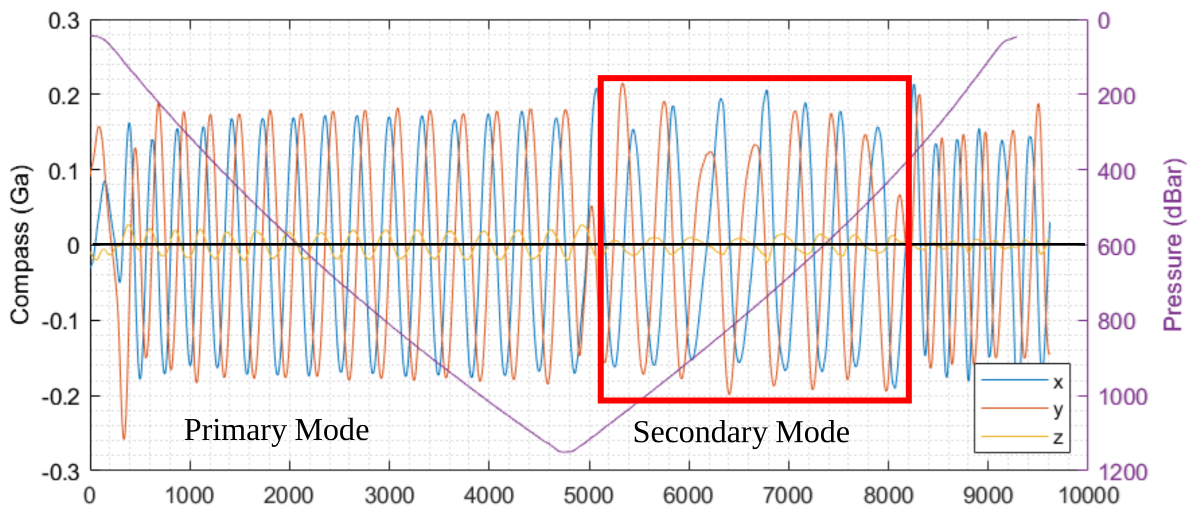
The same profile from Figure 12, is filtered and shown in Figure 15(a). From this plot, the fluctuations from +0.2Ga and -0.2Ga in the X and the Y axes of the compass can be clearly seen. Looking at the down-cast portion of Figure 15(a), and selecting any positive peak of the X component of the compass (see Figure 15(b)), it is observed that the Y component at that point is zero, and that a positive peak in the Y component shortly follows, with the X component at that point being zero. This phenomena is then exactly repeated in the negative side of the graph, by the negative peaks of the X and Y components of the graphs. This pattern is repeated throughout the down-cast, and on this particular profile, throughout the up-cast as well. Physically, this is representative of the X-axis of the fish initially pointing towards magnetic north (and Y towards west), followed by the rotation to a position where the X-axis points towards east (and Y towards north), followed by the X-axis pointing south (and Y east) and finally the X axis pointing west (and Y south). This indicates a clockwise rotation of the fish (looking down) which repeated throughout the entire dive. For this particular profile, the fish follows a remarkably similar mode of rotation on the up-cast when compared to the down-cast, in terms of both direction of rotation, as well as frequency. However, upon analyzing other profiles from the same section of the experiment (shown in Figure 15(c)), interesting observations are found. The plot suggests that on the up-cast, there are two modes of rotation, the first, and dominant one being a clockwise rotation (looking down), but also a second mode wherein the Y axis of the compass assumes magnetic north first, followed by the X axis, indicating a counter-clockwise rotation (looking down). This second mode also has a longer period, or a lower frequency. On some up-casts, the fish snaps into the counter-clockwise mode for a few rotations and then snaps back to the clockwise mode.



15(a)



15(b)



15(c)

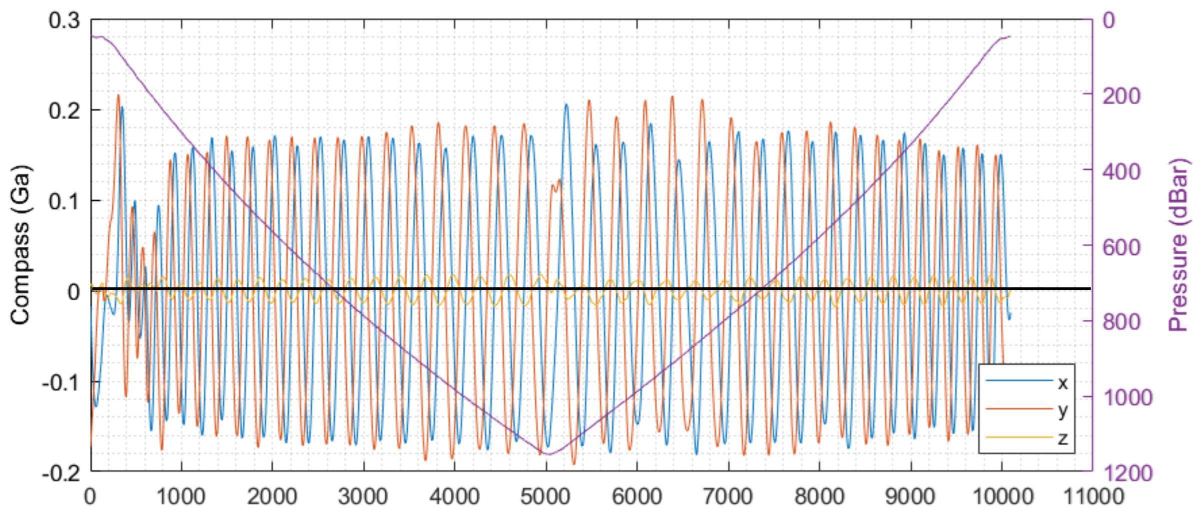
Figure 15 – (a) Filtered compass data with only one mode, (b) Compass data for one rotation (c) Filtered compass data with a second mode of rotation

In order to generalize the nomenclature of the rotations, the dominant mode of rotation will be referred to as a *primary* mode and the other mode, which only occurs on some of the up-casts, will be referred to as the *secondary* mode. For the control surface configuration of the profile in Figure 15, the primary mode is clockwise and the secondary mode is counter-clockwise.

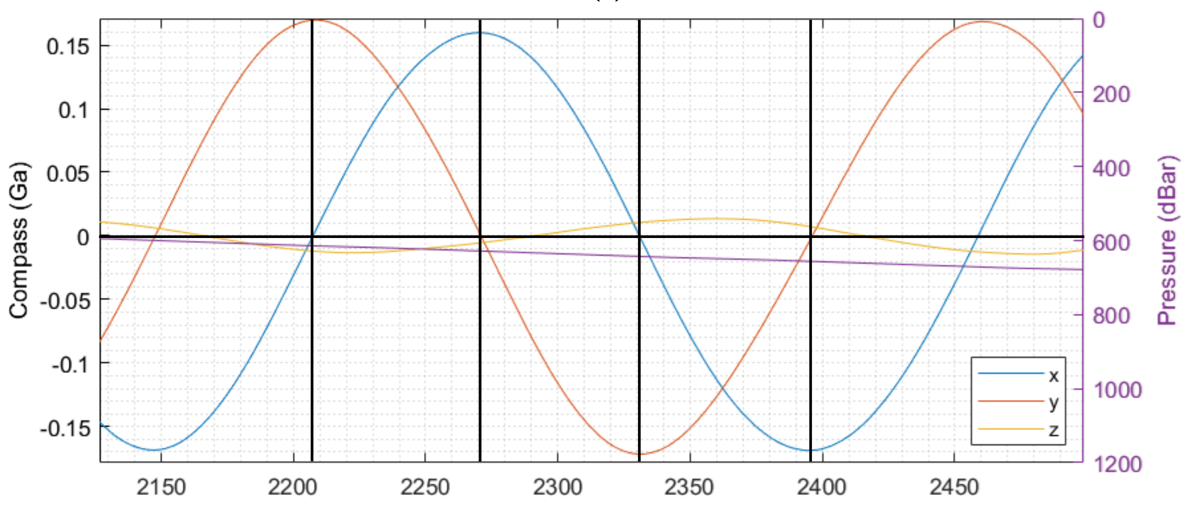
Revisiting the plot of gyro-meter data in Figure 13, which is from the same profile as Figure 15(a), there is a quasi-constant positive reading from the Z-axis of the gyro-meter. A positive value indicates that the rotation is counter-clockwise, while looking from the origin to the positive side of the axis. Hence, in our frame of reference, the rotation is counter-clockwise looking up, or, clockwise looking down. This checks out with our inference of clockwise rotation (looking down), that can be seen in Figure 15(a).

The occurrence of two modes of rotation can also be seen in a different section of the experiment, where the tail-fin control surface is flipped to the other direction. This is illustrated in Figure 16(a) and 16(b). However, for this opposite control surface configuration, it can be seen that the primary mode of rotation is not clockwise as observed earlier, but counter-clockwise (looking down), as the Y-axis assumes magnetic north first, shortly followed by the X-axis. Also, the secondary mode of rotation, occurring on some up-casts, is in fact clockwise. This mirrored behavior of rotation is attributed to the direction of tilt of the control surfaces, and is further characterized by spectral analysis.

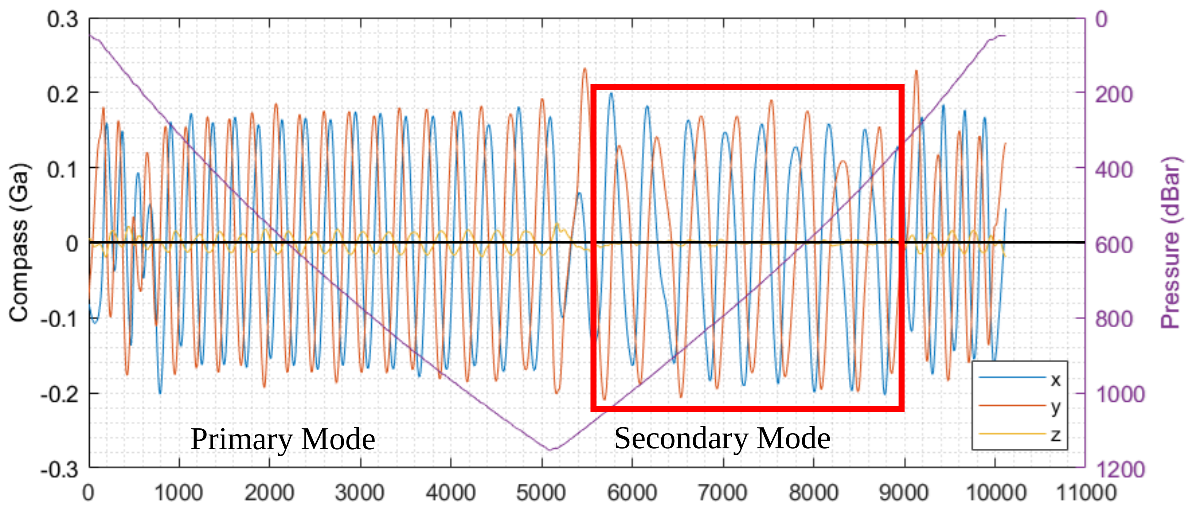
The asymmetrical and erratic distribution of compass components seen in the secondary mode of rotation are attributed to the inaccuracy of the assumption that gravity is the dominant force experienced by the fish on up-casts. Since the force exerted by the tow cable on the fish is comparable to gravity, the rotation of the coordinate system is not as precise as on the down-casts. This means that the X and Y axis of the IMU are not perfectly in the horizontal, and capture some portion of the vertical component of earth’s magnetic field. As a result, they do not perfectly oscillate from +0.2Ga to -0.2Ga, and are occasionally not symmetrically placed about the zero line. The direction of rotation, however, can still be inferred by the order in which the X and Y peaks occur.



16(a)



16(b)



16(c)

Figure 16 – (a) Filtered compass data with only primary mode, (b) Compass data for one rotation and (c) Filtered compass data with secondary mode of rotation

1.7 - Spectral Analysis

1.7.1 - Rotary Spectra Analysis

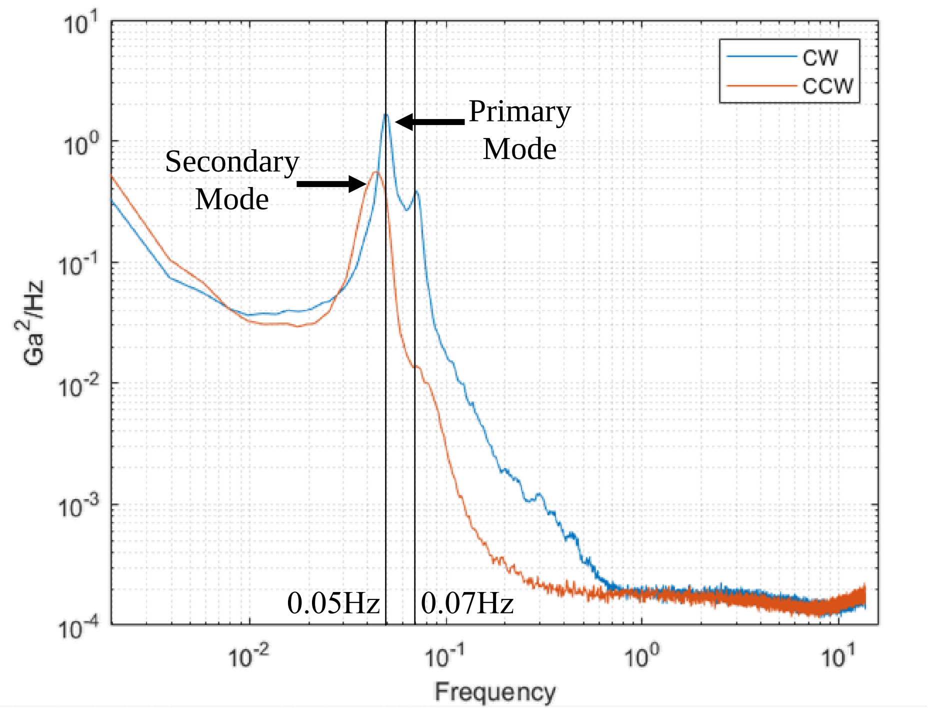
A useful tool in the characterization of multiple modes and directions of rotation is *rotary* spectral analysis. It can provide a power spectral density estimate of each direction of rotation, clockwise or counter-clockwise. A comparison of the power estimate and the frequency at which the peak occurs should reflect the inferences made in the previous sections of the analysis.

In rotary spectra, the parameters of interest, the X and Y axes of the compass in this case, are used as the real and imaginary parts of a complex number. The fluctuation in the intensity of the parameters results in the rotation of the complex number about the origin in the complex plane, with the same frequency. This complex number array is input to the *pwelch* function, which provides the PSD across a range of frequencies.

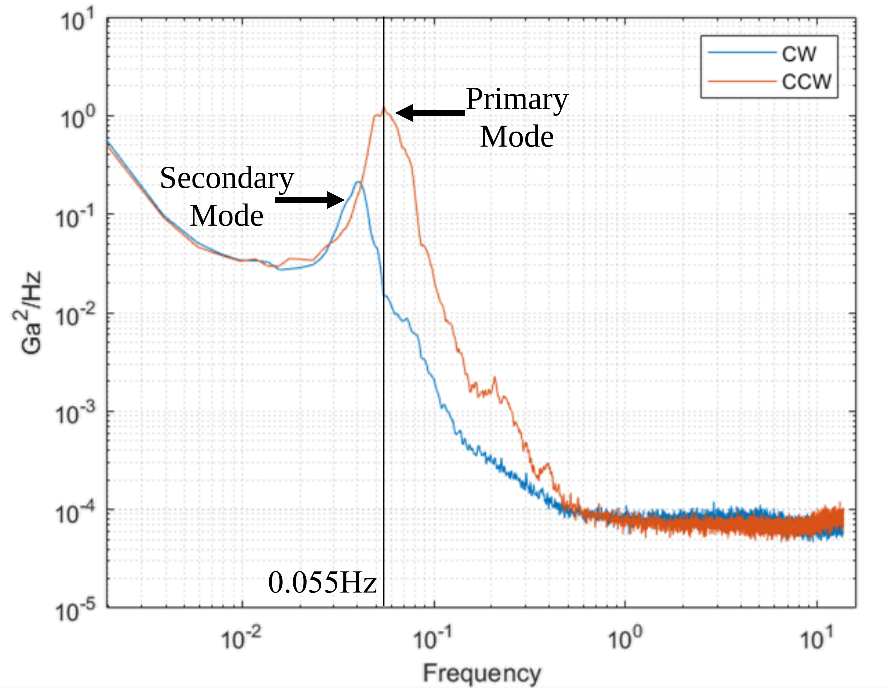
A salient feature of this analysis is the selection of the real and imaginary parts for the complex number. Assigning the X-axis of the compass to the real part and the Y-axis to the imaginary part will give the PSD of clockwise rotation of the fish and conversely, assigning the Y-axis of the compass to the real part and X-axis to the imaginary part will give the PSD of counter-clockwise rotation of the fish.

The main purpose of the rotary spectra analysis is to illustrate the relation between the configuration of the control surfaces on the tail-fins of the fish and the instability. Hence the analysis is performed separately for each configuration. Figure 17(a) and 17(b) are graphs of the PSD of clockwise and counter-clockwise modes for a range of frequencies, for the two control surface configurations. The graph indicates that irrespective of the direction of tilt of the control surface, two modes of instability occur, with one being more dominant than the other, and the dominant mode having a higher frequency as well. Figure 17(a) corresponds to the control surface configuration from Figure 14, which was used to construct the filter, hence it too contains a peak for the primary mode at 0.05Hz. Figure 17(b) however, corresponds to a reversed control surface configuration, and contains peak for the primary mode at 0.055Hz. Since the manipulation of the control surfaces is done somewhat approximately by hand over the course of the experiment, the frequencies are not perfectly mirrored. Nevertheless, the frequencies are similar enough to classify the modes of rotation.

In Figure 17(a), the curve representing the primary (clockwise) mode of rotation in blue, is seen to have another, lower power peak at a higher frequency (around 0.07Hz). It is not clear what phenomena this peak represents, but it could possibly be a third mode of rotation, which is in the same direction as the primary mode but of a higher frequency. The peak associated with this possible third mode of rotation, is not seen in the rotary spectral plots for the reversed control surface configuration (Figure 17(b)). This could indicate a sensitivity of this phenomena to the degree of tilt of the control surfaces.



17(a)



17(b)

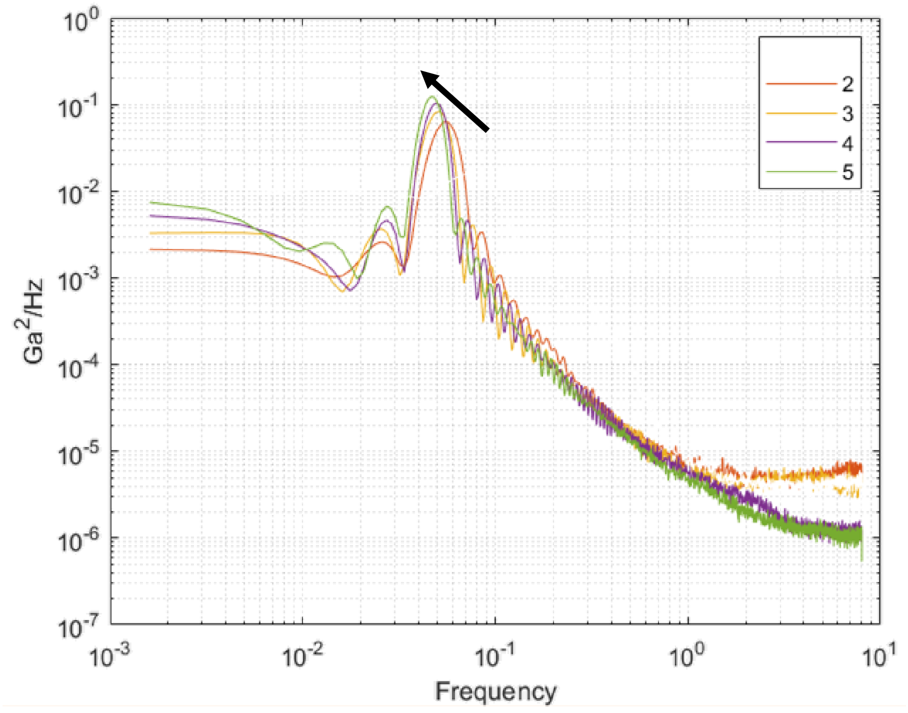
Figure 17 – (a) Rotary spectra for clockwise and anti-clockwise rotation and (b) Rotary spectra with control surfaces reversed

1.7.2 - Depth Based Analysis

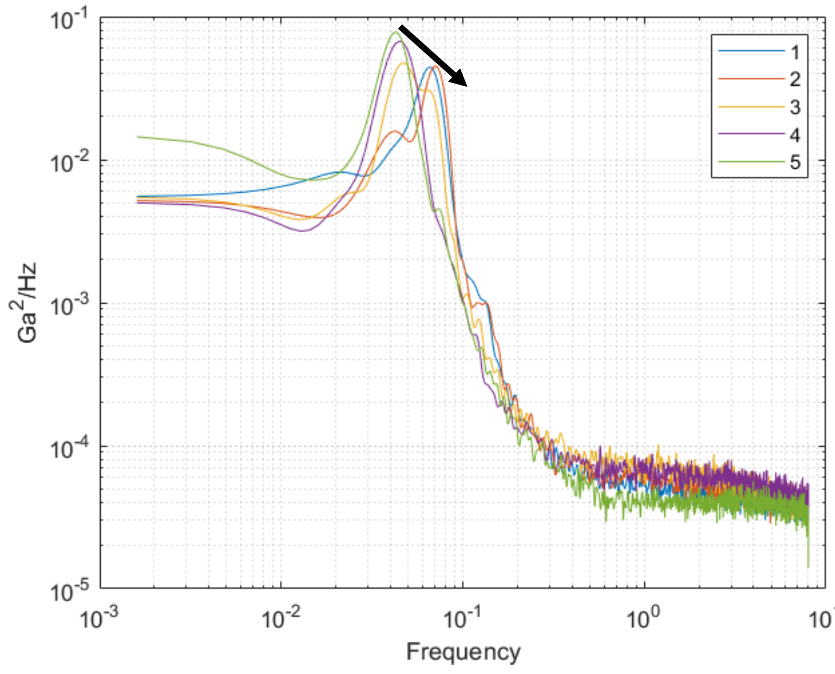
In order to understand the variations in behavior of the fish with depth, data from the IMU was divided into specifiable number of sections, for each vertical profile. Then, a spectral analysis was carried out for each section, and comparing the PSD shows how the intensity of rotation as well as the frequency varies across different depths, and also across a down-cast and an up-cast. The PSD estimates were obtained once again by using Matlab's *pwelch* function. Figure 18(a) and 18(b) shows PSD plots for down-casts and up-casts when the number of sections has been set at five. In both plots, the PSD for the shallowest section is labelled 1 and for the deepest section, is labelled 5.

Looking at the PSD of the down-cast, in Figure 18(a), certain inferences can be drawn. The shallowest section has been omitted from the plot as it contains samples at which the fish is at the surface of the water, and just starting to descend, where there is no rotation occurring. The remaining four sections of the down-cast, however, have distinct peaks. The arrow indicates the direction of motion, from shallower sections to deeper sections of the down-cast. It can be seen that the frequency of rotation decreases with depth, the fish has a longer period of rotation. Also, the power associated with the rotation marginally increases with depth.

The PSD of the up-cast, in Figure 18(b), reveals a lot of information. The deepest section of the plot, represented by the label 5, has one distinct peak, and at a lower frequency. However, as we move to shallower sections such as the middle section, labelled 3, two peaks appear, one at a lower frequency and one at a higher frequency. And even shallower than that, the peak associated with the lower frequency disappears and only the higher frequency peak remains. The phenomena of snapping from the lower frequency secondary mode of rotation to the higher frequency primary mode of rotation is clearly elicited from the PSD of the up-cast. The arrow indicates the direction of motion, from deeper sections to shallower sections of the up-cast



18(a)



18(b)

Figure 18 – (a) PSD of rotation at varying depth sections on down-cast and (b) PSD of rotation at varying depth sections on up-cast

1.8 - Flight Optimization Strategy

From the compass and the PSD plots, it is inferred that the undesirable flight instability of the fish is mainly associated with the clockwise mode of rotation. Although the sporadic occurrence of the counter-clockwise mode on the up-cast slightly mitigates the problem by temporarily unwinding the tow cable, it is too ephemeral in nature, and does not occur on every up-cast. Hence it cannot be relied on to undo the twisting occurring from the primary mode. In order to mitigate the rate of rotation, the effect that the fins and the control surfaces have on the fish need to be manipulated. This is complicated by the fact that control over the direction of rotation is required, as a minimal amount of rotation needs to be induced to ensure stable and predictable flight. However, the rate of rotation induced needs to be curtailed.

Current control surface configuration has the effect of inducing a moment on the fish, which results in rotations. As noted earlier from Figure 10 that the angle with the horizontal is approximately 60° throughout the downcast, this induced moment has a two components. One is a moment about a horizontal axis, and the second a moment about a vertical axis, as illustrated in Figure 19.

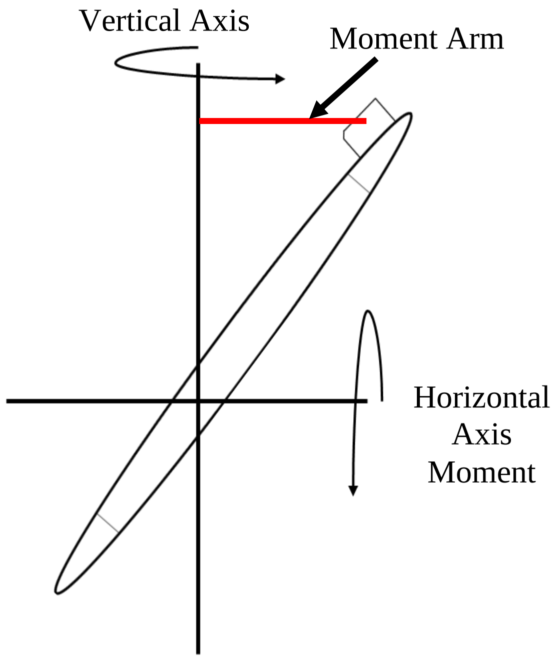


Figure 19 – Sketch illustrating moments acting on the fish, moment arm for the vertical moment

As the fish travels through the water, the horizontal axis moment tries to rotate the fish about the horizontal axis, but from the accelerometer data, it can be seen that there is not such roll occurring as the X-axis of the accelerometer is approximately zero throughout. This is likely due to the fins providing a horizontal stabilizing moment, to counter-act the moment from the control surfaces. The stabilizing moment arises from the fact that the fins must be parallel to the flow under steady state, or they will experience a force in either direction. The vertical axis moment from the control surfaces, on the other hand, has no stabilizing or opposing moment, except for the inertia of the fish itself. This results in the observed excessive rotation about the vertical axis.

A first order mitigation step in reducing the vertical moment, would be to decrease the length of the moment arm. The moment arm for the vertical moment is shown in red in Figure 19. A decrease in the length of the moment arm can be achieved by steepening the descent of the fish, increasing in the angle of descent. The effect of varying the angle of descent on the vertical and horizontal moments of the fish is studied in the following chapter by performing a computational fluid dynamic analysis.

Chapter 2 - Computational Fluid Dynamics Analysis

The objective of this section is to understand the influence of the control surfaces and fins on the flight of the fish by performing a first order CFD analysis to obtain the forces and moments generated for varying angles of descent. The CFD analysis is performed using the toolkit OpenFOAM, for which a brief introduction is provided.

2.1 - OpenFOAM (OF) Overview:

OpenFOAM is an object-oriented open-source CFD software package built on the C++ programming language. As a result, a basic knowledge of C++ is a pre-requisite. OpenFOAM is a complete tool using different libraries to solve complex fluid dynamics problems [12]. The overall structure of OF is displayed in Figure 20. The user has tools available for pre-processing and solving, but no specific post-processing software. For this purpose, a post-processing software called ParaView is most commonly used. It is adopted in this work, for visualization.

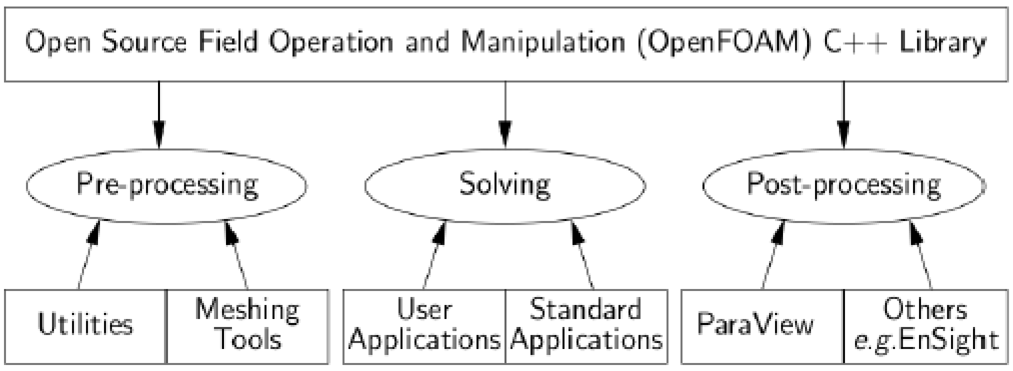


Figure 20 – Structure of OF CFD toolkit [12]

There are two types of entities in OF, the first one is applications, which are executables such as solvers or utilities. Solvers, as the name suggests, solve equations of continuum mechanics, and utilities are used for manipulating data. Each solver represents a specific continuum mechanics problem, with specific implementations of the governing equations. The source code files for each solver in OF are intentionally made accessible, so one can make an informed decision about which solver to use upon studying its capabilities, and make any modifications to the solver, should it be required.

The second type of entity are called cases. Solvers contain physics, but cases on the other hand encompass the specific aspects of each problem such as properties, computational schemes and geometries. Cases in OF are made up of files arranged in a specific folder structure. There are physical property files for velocity, pressure, and turbulence fields, as well as for necessary constants and settings. There are also files that control different aspects of simulations, such as meshing and solving, commonly called dictionaries. Different dictionaries may be required when different OF tools are being used. These files are organized in a specific way, in three directories:

- Constant Directory: This folder contains the physical constants needed for the simulation, as well as a description of the mesh in the folder *polyMesh*.
- System Directory: This folder contains parameters for solver settings and for controlling the simulation. The main file to access these controls is *controlDict*, and it contains all temporal setting for the simulation, among many others. Mathematical controls for solving can be found in the *fvSchemes* and *fvSolutions* files.
- Time Directory: This folder contains initial and boundary settings for the simulation. To be specific, the folder with the initial-time as its name, which in most cases is *0*, contains the initial and boundary settings. Different fields such as velocity and pressure are stored in separate files. Results are written by OF to directories named by the time-step.

Figure 21 depicts a typical case structure containing all files that were used for a simulation in this work.

```
fish
├── 0
│   ├── k
│   ├── nut
│   ├── omega
│   ├── p
│   └── U
├── constant
│   ├── extendedFeatureEdgeMesh
│   ├── polyMesh
│   ├── triSurface
│   ├── transportProperties
│   └── turbulenceProperties
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── cuttingPlane
    ├── decomposeParDict
    ├── forces
    ├── fvSchemes
    ├── fvSolution
    ├── meshQualityDict
    ├── snappyHexMeshDict
    └── surfaceFeaturesDict
```

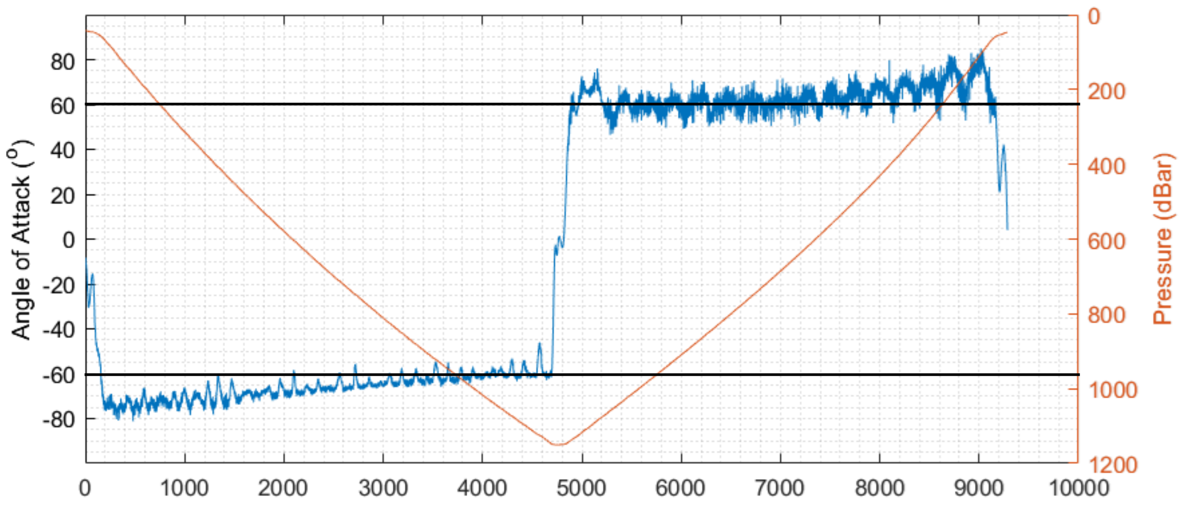
Figure 21 – Case setup example

From a user point of view, one of the advantages of using OpenFOAM are the tutorials that come along with the installation. The developers have set up cases for several physical problems, with executable files to run them and interpret their results. As a result of the variety of tutorials available, the most common approach to developing a case for a problem is to find the tutorial that most closely relates to the required problem, and apply the necessary modifications to it. For the purpose of this work, the tutorial *motorbike* under the *simpleFoam* solver in the *incompressible* case tutorials was adopted and modified accordingly.

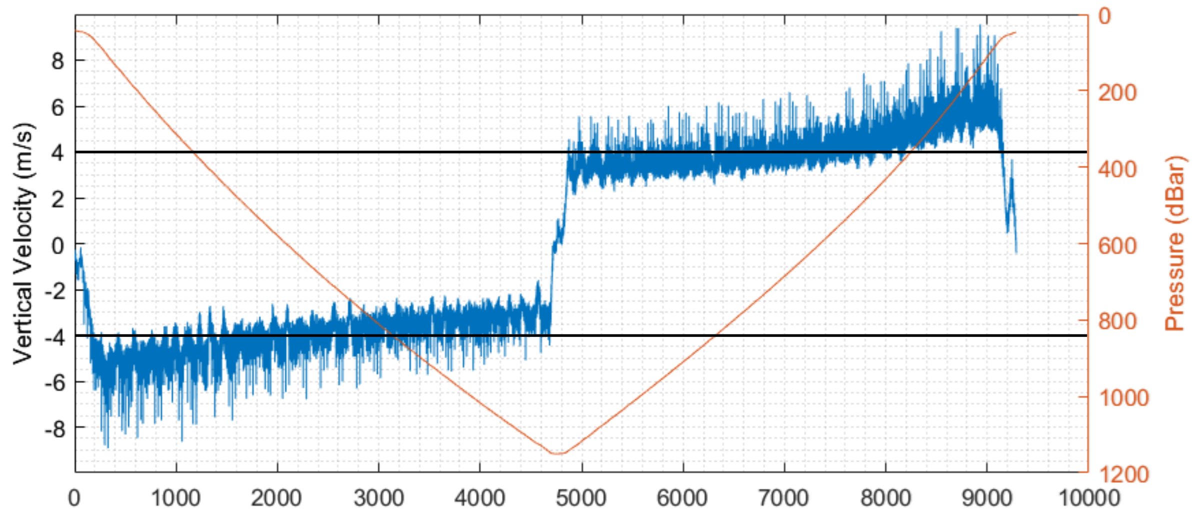
2.2 - Case Setup

The objective of the simulations, as mentioned earlier in this section, is to understand the influence of the control surfaces and fins on the flight of the fish. The forces and moments generated by the control surfaces at different orientations, over multiple angles of descent, are to be studied. For the purpose of simplicity, only the down-cast conditions are simulated, as the up-cast contains an external force on the fish through the winch which varies in time. Since the fish appears to behave the same way on up-casts and down-casts from the data analysis (at least partially), useful results can be drawn from simulating just the down-cast environment. The down-cast of the fish is assumed to be a free-fall through the water. The case is thus structured to be a cuboidal wind tunnel, with the fish contained in it. It is found from the data that the fish approximately has an angle of descent of 60° throughout a downcast, and a vertical velocity of about 4 m/s (supported by Figure 22(a) and 22(b)). Hence, four different cases are developed, with the fish at 60°, 65°, 70° and 75° angles of descent, free-falling at 4 m/s. At each of these angles, three simulations are run; one with the control surfaces in a neutral position, and one each for the two sides the control surfaces can be angled towards. That results in a total of 12 simulations. An evaluation of the forces and moments from these conditions would help in understanding if steepening the descent of the fish would decrease the number of rotations.

An important point to note here, is that for the simulation, we have assumed that the fish free-falls in a straight line, and it rotates about this same line. In reality, it could be possible that the fish follows a helical path on the way down, in which case the rotations are as a result of the fish revolving about a center line as it descends. The fish could even be undergoing a combination of straight-line free-fall and helical motion, or each of these types of descent could relate to the primary and secondary modes discovered earlier. However, the purpose of the simulations is to perform a force assessment, which is most straightforward with the first assumption.



22(a)



22(b)

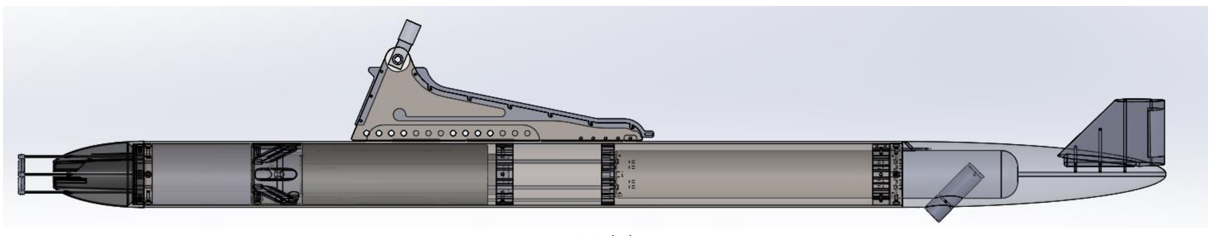
Figure 22 – (a) Angle of attack of the fish and (b) Vertical velocity of the fish

It is should also be noted here that the simulations will be carried out from the fish's inertial frame of reference. The fish will be stationary within the wind tunnel, and flow will be forced at 4 m/s over it. Also, until this chapter, the free-fall is assumed to be vertical, and angles pertaining to the fish such as angle of attack have been with respect to the horizontal. However, for the simulation, the flow will be in the horizontal direction, from left to right, and so the angles of attack of the fish are with respect to a vertical axis. Figures in the following section depict this configuration for the simulations.

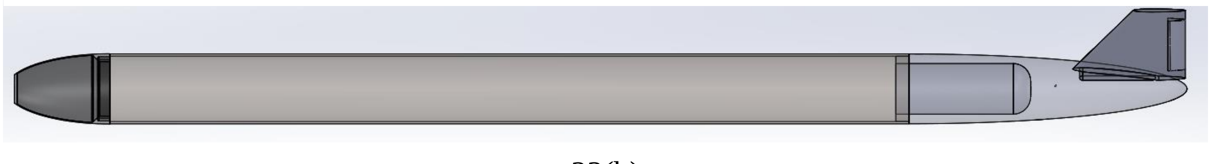
2.2.1 - Geometry and Meshing

2.2.1.1 - CAD Pre-Processing

The CFD model of the fish in OpenFOAM was developed from the existing SolidWorks assembly files that were created during the design of the fish. For the purpose of a first order analysis, several simplifications are made to the fish geometry, keeping only the bare minimum components that affect the flight stability of the fish. Components such as the dorsal fin arrangement, altimeter port, conductivity sensor channels etc. are removed, by simply deleting the parts from the SolidWorks assembly. The original SolidWorks model of the fish is shown in Figure 23(a), and the simplified model is shown in Figure 23(b). Three versions of the simplified model are created, with the control surfaces in a neutral position (called *CSflat*), angled towards one direction (towards the left, while looking at the tail of the fish from behind, called *CSminus10*) and angled towards the opposite direction (towards the right, while looking at the tail of the fish from behind, called *CSplus10*). The control surface positions are depicted in Figure 24(a), 24(b), and 24(c) respectively.

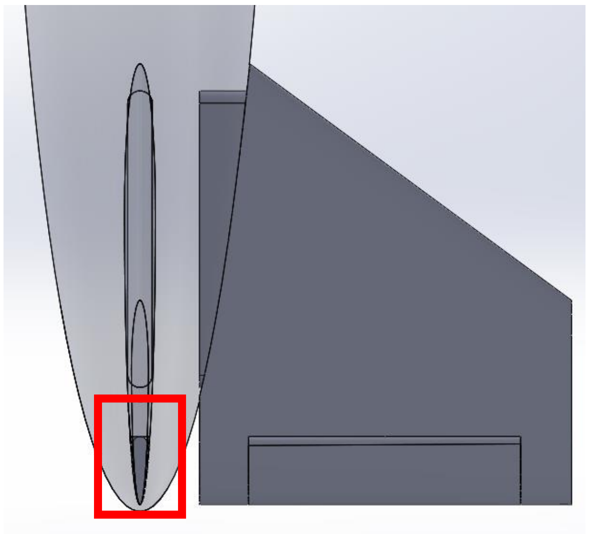


23(a)

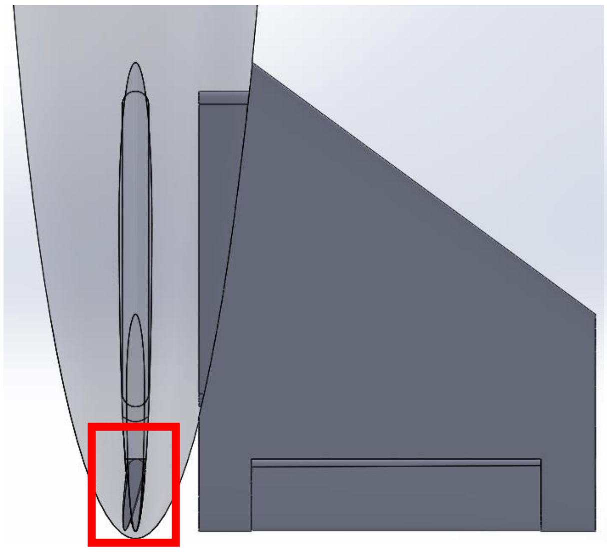


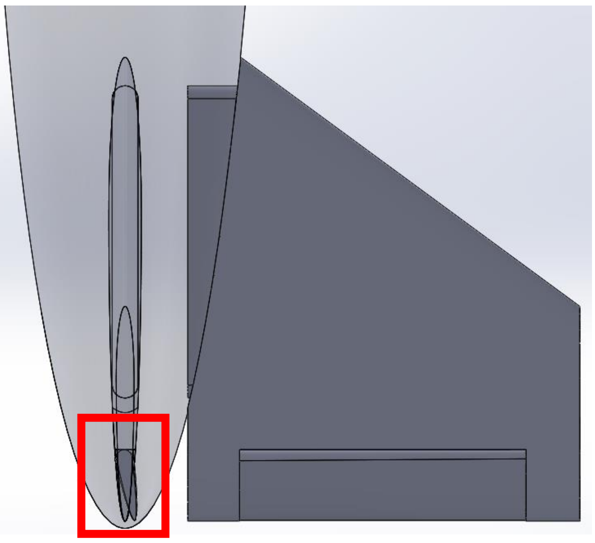
23(b)

Figure 23 – (a) Full fish assembly and (b) simplified fish assembly



24(a)

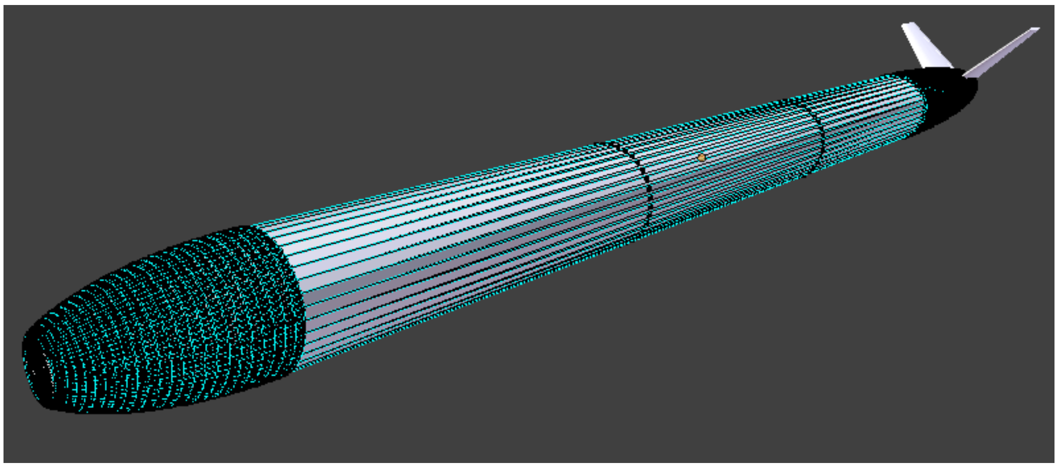




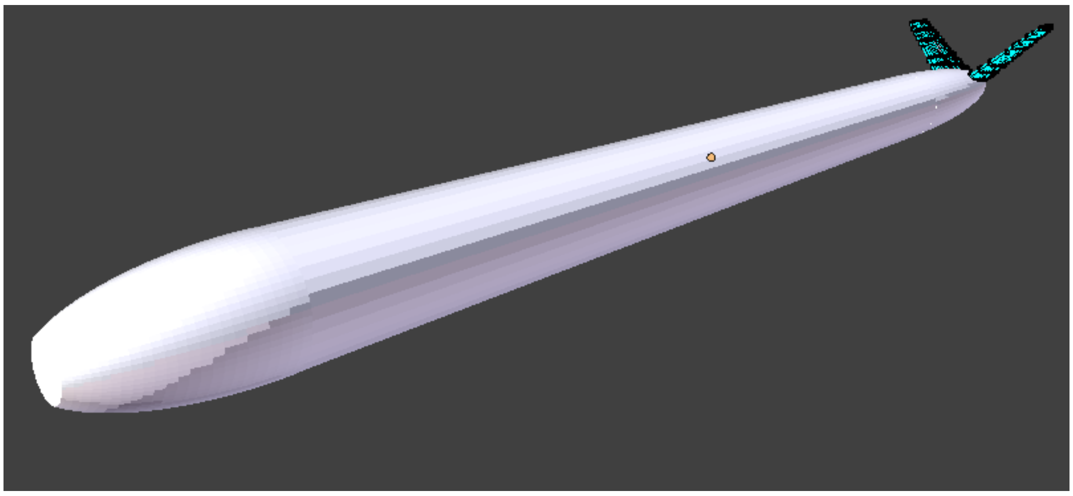
24(b) 24(c)

Figure 24 – (a) Neutral control surfaces ‘CSflat’, (b) Left-biased control surfaces ‘CSminus10’ and (c) Right-biased control surfaces ‘CSplus10’

The final models are exported to an STL file, and imported into Blender, a free modeling software. The purpose of this step is to properly position the fish with respect to the coordinate system origin, which is simpler in Blender than in SolidWorks, and to define different regions within the fish. Positioning of the fish at different angles of descent is done at this stage. Definition of multiple regions enables individual post-processing operations for each region, which is vital for this analysis. The fish was divided into two regions, the body and the fins (both fins grouped in one region), as shown in Figure 25(a) and 25(b). This model is saved as an OBJ file, as this is found to be the most convenient file format to input multiple-region-geometries to OF.



25(a)



25(b)

Figure 25 – (a) Region containing body of fish in Blender and (b) Region containing both fins of fish in Blender

2.2.1.2 - Mesh Generation using snappyHexMesh

To carry out a wind-tunnel type simulation, an external mesh that envelopes the fish is to be generated. The mesh needs to be limited by some definite walls in all three dimensions. This is a fairly complex mesh structure, and thus, the OF native mesh generation tool *snappyHexMesh* has been used.

A summary of the steps involved in using *snappyHexMesh* are listed below.

- Bounding/background mesh generated using *blockMesh*
- Load geometry in STL or OBJ format
- Create a castellated mesh
- Snap the mesh to the surface
- Add layers between the surface and mesh

The background mesh is generated using the *blockMesh* utility in OF and controlled by the dictionary file *blockMeshDict*. An example blockMeshDict is attached in Appendix A.1. The boundaries of the simulation, which would be the walls of the wind tunnel, are chosen to be at a distance where the flow can be considered free-stream. The walls are also defined such that there is additional space downstream from the fish, so any wake phenomena may be properly recorded by the simulation. The background mesh is made moderately fine, but not too fine. This is because further refinement around the fish will be carried out by *snappyHexMesh* and having a very fine mesh at great distances from the fish is inefficient.

The *snappyHexMesh* is generated and controlled by the dictionary file *snappyHexMeshDict*. There are three main functions called upon by *snappyHexMeshDict*; *autoRefineDriver* which is responsible for the refining and cutting of the mesh, *autoSnapDriver* which snaps the mesh onto the surface, and *autoLayerDriver* which creates the boundary layers [12]. The castellation of the mesh, snapping to the surface, and addition of layers are all activated in the dictionary file, among several other controls. This is an iterative process, and *snappyHexMeshDict* needs to be tuned to achieve the desired mesh refinement. An example *snappyHexMeshDict* is attached in Appendix A.2 and is a comprehensive list of all available controls. The most important controls are listed in Table 2 to give an overview.

Table 2 – snappyHexMeshDict controls [12]

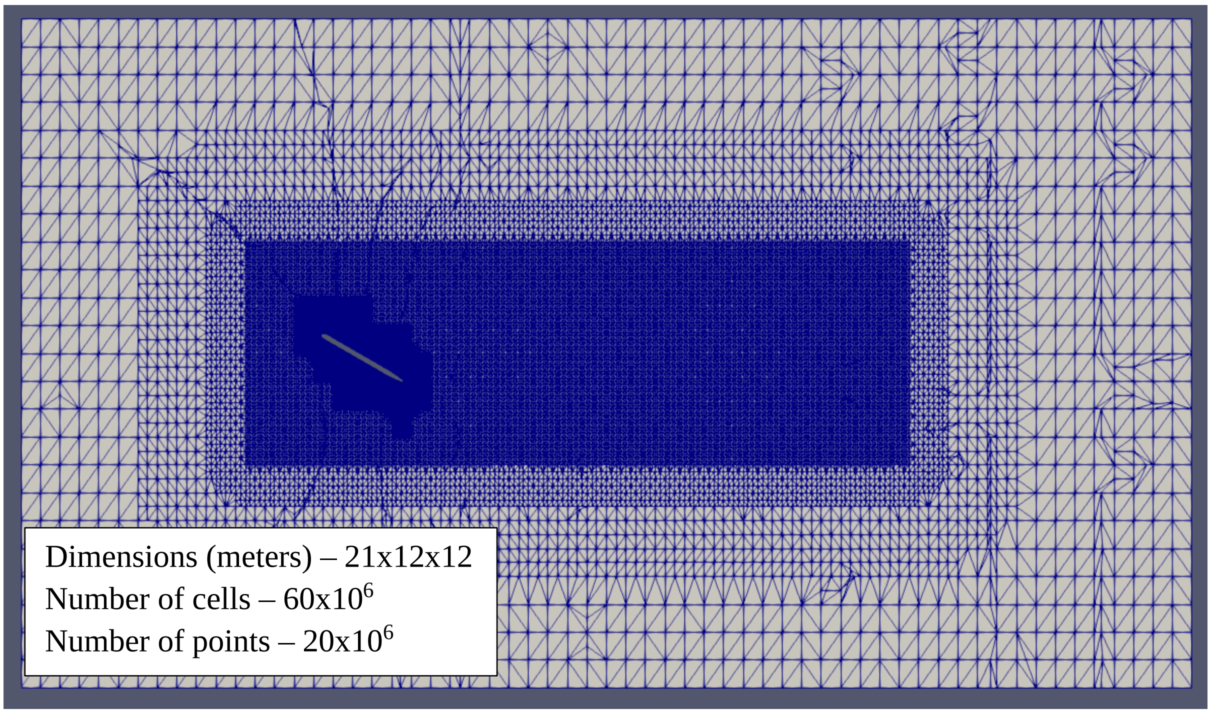
Main Controls	
castellatedMesh	Create the castellated mesh
snap	Perform surface snapping
addLayers	Add surface layers
castellatedMeshControls	
locationInMesh	Location vector inside the region to be meshed
maxGlobalCells	Total cell limit
features	Refinement level for cells intersected by its edges
refinementSurfaces	Refinement level for cells intersected by its surfaces
refinementRegions	Refinement level for cells in relation to a surface
snapControls	
nSmoothPatch	Number of patch smoothing iterations
nSolveIter	Number of mesh displacement iterations
nRelaxIter	Number of snapping relaxation iterations
addLayersControls	
layers	Number of layers per final patch
finalLayerThickness	Desired final added layer thickness
minThickness	Minimum thickness of cell layer
nSmoothNormals	Number of smoothing iterations of interior mesh

First, the geometry, the regions within the geometry, and their desired patch names are specified. At this stage, refinement shapes such as boxes or spheres may be defined at different points in the background mesh where further refinement may be required. In this case, a refinement box around the fish is defined so as to have a fine, high-quality mesh around the surface, and downstream of the fish.

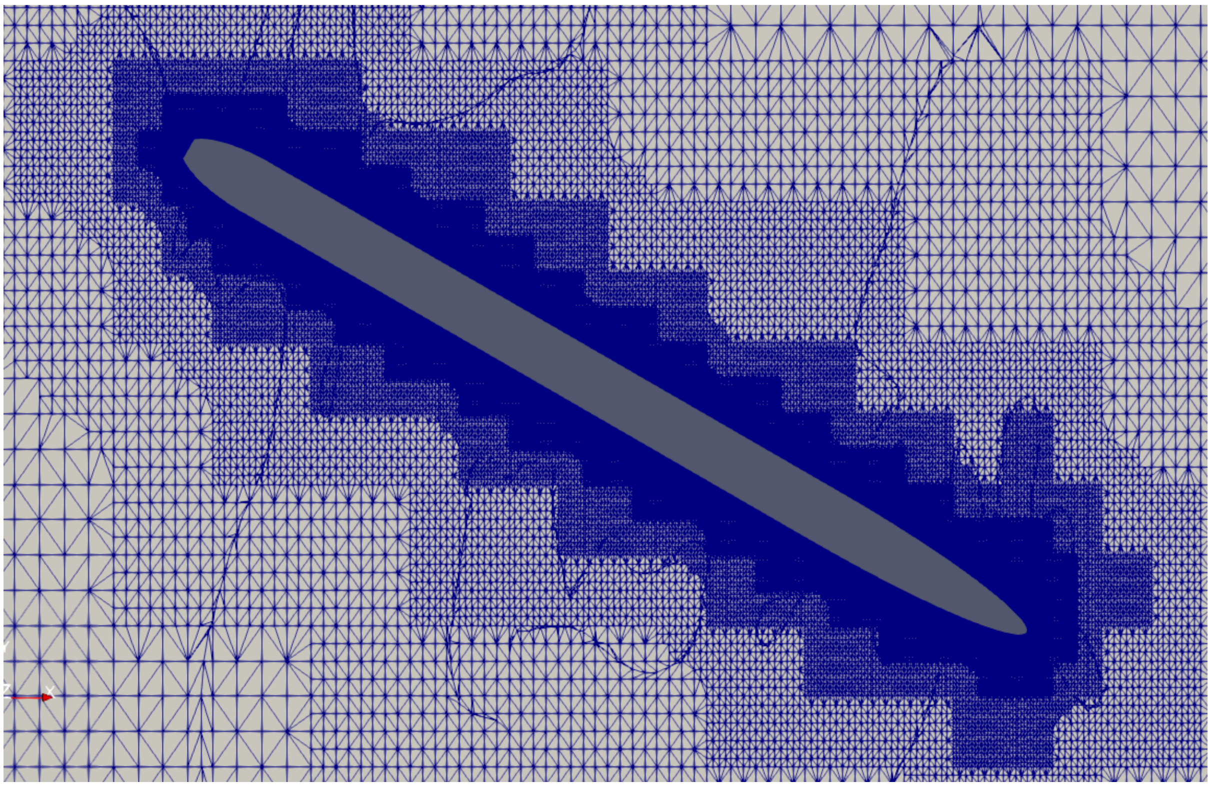
The next stage of meshing is the castellation of the mesh. Here, the cells of the background mesh that completely lie within the geometry are removed and cells that intersect the surface of the geometry are refined. The location within the boundaries where the mesh should be generated must be specified. Several features such as maximum total number of cells, number of surface refinement cells and number of cells between levels can be controlled. Two refinements are carried out at this stage, a surface level refinement, by providing an eMesh file which is obtained using a *surfaceFeatures* utility in OF, and refinement associated with the refinement regions that have been defined in the geometry section. This further refinement can be targeted either inside or outside the regions defined (inside, for this work).

Once the castellation has been completed, the mesh needs to be snapped to the surface of the geometry, or else the faces of the cells adjacent to the surface will be considered as the surface of the geometry itself, distorting the fish. Several controls such as number of snapping iterations, number of iterations for features and number of patch smoothing iterations are available. Following the surface snapping of the mesh, additional layers can be added. However, it is recommended to generate a stable mesh before carrying out layer addition operations, as they tend to be fairly complex. For this work, one additional layer is added. Similar to the castellation and snapping steps, several controls are available such as number of layers, layer thicknesses, number of smoothing normal and number of iterations.

Apart from the above, certain miscellaneous controls are available in the *meshQualityControls* section, which are tuned accordingly to obtain a high-quality mesh. The final mesh is shown in Figure 26(a) and 26(b). Figure 27(a) shows the names of the patches which have been defined, which will be useful for the following sections where initial and boundary conditions have been defined, and Figure 27(b) shows the coordinate axes with respect to the orientation of the fish.

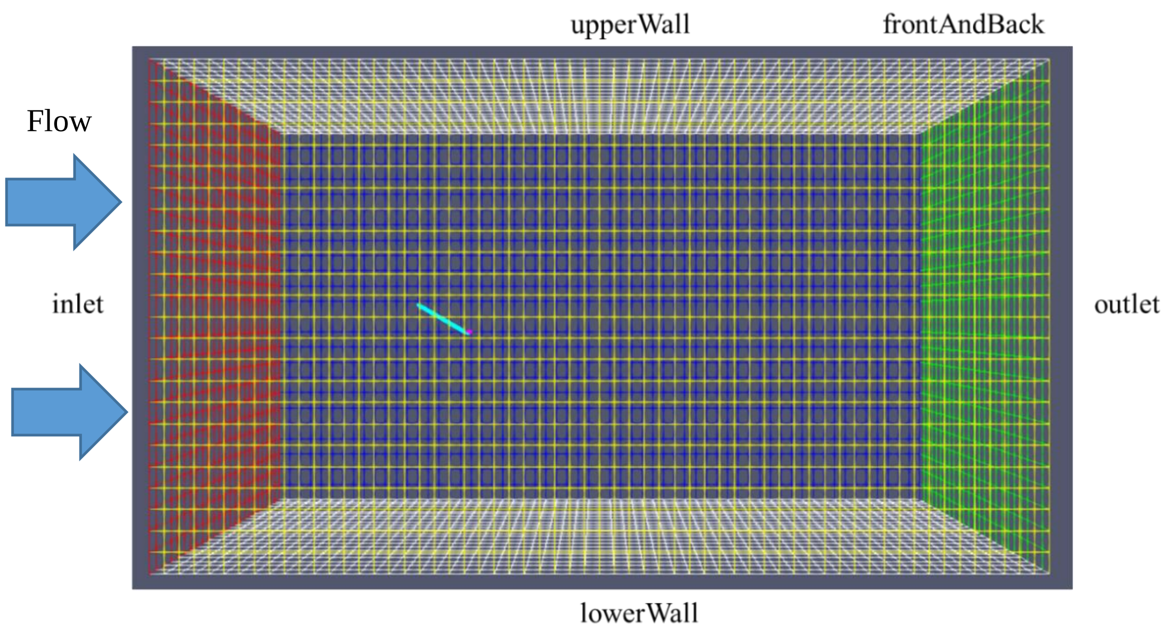

Dimensions (meters) – 21x12x12
Number of cells – 60x10^6
Number of points – 20x10^6

26(a)

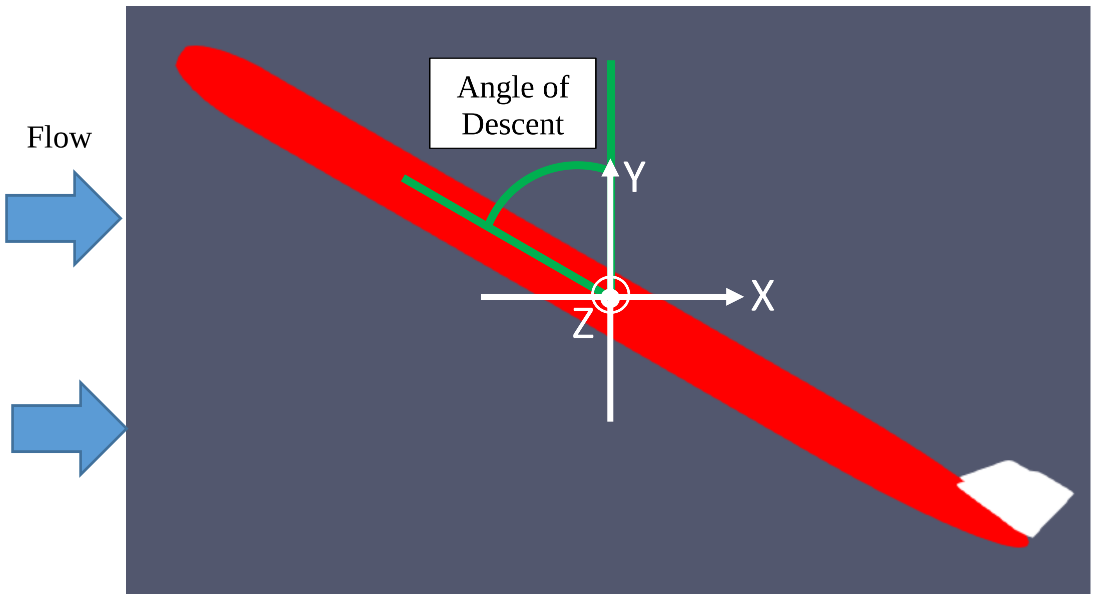


26(b)

Figure 26 – (a) Slice of entire bounding box mesh and (b) Zoomed in image of mesh around the fish



27(a)



27(b)

Figure 27 – (a) Bounding box with fish inside, along with patch-names and (b) Coordinate axis with respect to the fish

2.2.2 - Boundary and initial conditions

Boundary and initial conditions in OF, are provided in the *0* directory of the case folder (since our simulation starts at *t=0*). The parameters such as velocity, pressure, etc. are referred to as *fields* and each field has its own file which depicts the values at each of the patches in the mesh. Field files for velocity and pressure are attached in Appendix A.3. Conditions at these patches are enforced by specifying the type of patch, in relation to that field. For example, for the field velocity, the patches containing the surface of the fish would be of type *noSlip*, and the inlet, top and bottom, and front and back patches of the bounding box would be of type *freestream*, with another line specifying the value of the free-stream velocity. In this manner, initial and boundary conditions are applied on every patch for every field. An overview of the initial and boundary conditions for pressure and velocity can be found in Table 3.

Table 3 – Initial and boundary conditions for pressure, velocity

Boundary		**Pressure**	**p**	**Velocity**	**U**
Patch Group	**Patch Name**	**Type**	**Value**	**Type**	**Value**
inlets	inlet	freestreamPressure	0	freestreamVelocity	(4 0 0)
	lowerWall	freestreamPressure	0	freestreamVelocity	(4 0 0)
	upperWall	freestreamPressure	0	freestreamVelocity	(4 0 0)
	frontAndBack	freestreamPressure	0	freestreamVelocity	(4 0 0)
outlet	outlet	freestreamPressure	0	freestreamVelocity	(4 0 0)
fish	body	zeroGradient	-	noSlip	(0 0 0)
	fins	zeroGradient	-	noSlip	(0 0 0)

2.2.3 - Fluid Properties

The fluid properties are found in the *constant* folder, in a file named *transportProperties*. The transport model was selected to be Newtonian, which means that the kinematic viscosity is constant. It is specified under the keyword *nu* in this file and was taken to be equal to $1.05 \times 10^{-6} m^2/s$, which is a value obtained for seawater at a temperature of 20°C and a salinity of 35 g/kg [13]. Since the flow is assumed to incompressible, no input for density is required. However, a density value is required later on during post-processing force evaluation.

2.2.4 - Turbulence Model

An evaluation of the flow using a velocity of *4 m/s*, length scale of *1.7 m* (nose to tail length of the fish) and a viscosity of *1.05 x* $10^{-6} m^2/s$, yields a Reynolds number of approximately *6.5 x* 10^{6}. There is no available pertinent literature for an exact critical Reynolds number of external aerodynamic flow, as it is heavily geometry dependent, but a number of the order of 10^6 warrants turbulence modeling in the simulations.

Reynolds Averaged Navier Stokes (RANS) turbulence models offer an economic approach for computing complex turbulent flows. Prominent examples are the $k - \epsilon$ or $k - \omega$ models, that simplify the solution by adding an Eddy-Viscosity to compute the Reynolds Stresses. For this analysis, a $k - \omega$ SST (Shear Stress Transport) model is chosen, as it is better at predicting adverse pressure gradient boundary layer flows and separation compared to the $k - \epsilon$ model, and is also better designed to avoid the freestream sensitivity of the $k - \omega$ model. Furthermore, the SST model has been calibrated to accurately compute flow separation from smooth surfaces. For this reason, it one the most widely used model for aerodynamic flows [14].

The turbulence controls are accessed using the *turbulenceProperties* file located in the *constant* directory. First, the simulation is specified to be RAS (or RANS) and then the $k - \omega$ SST model is chosen by using the keyword *kOmegaSST*. Finally, the turbulence option is turned on. The pre-requisite to turbulence modeling, is that the solver chosen for the simulation permits computations involving turbulence, which is true for this case.

2.2.5 - ***simpleFoam*** Solver

The solver to be used for this analysis is selected based on the assumptions made on the fluid flow for the simulation.

The assumptions and their justification are listed as follow:

- Incompressible – The flow-field is assumed to be incompressible, as the Mach number of seawater flowing at 4 m/s is 0.0026. The flow would have to be in the near-sonic region to consider compressibility effects.
- Steady State – Since the desired output of the simulation is an evaluation of the forces and moments acting on the fish, a steady state analysis is sufficient to obtain these results. Suppressing the transient terms of the governing equations vastly decreases the computational effort involved in performing the simulation.
- The solver also needs to be able to model turbulence, given the moderately high Reynolds number of the flow (6.5 million).

For these reasons, the incompressible, steady state solver *simpleFoam* is selected for this simulation, and the tutorial case *motorBike,* which uses simpleFoam is adopted to build the case directory. SIMPLE, stands for Semi-implicit Method for Pressure-Linked Equations. It follows the following iterative process [15]:

1. Guess a pressure field.
2. Solve a discretized momentum equation for the intermediate velocity field.
3. Compute the mass fluxes at the cells faces.
4. Solve a pressure correction equation defined from continuity equation using under-relaxation.
5. Correct the mass fluxes at the cell faces.
6. Correct the velocities on the basis of the new pressure field.
7. Update the boundary conditions and the pressure field.
8. Repeat until convergence.

Since *simpleFoam* is a steady state solver, the time steps used are integration time steps, not actual time steps. That is, they have no relation to the actual passing of time, but rather the number of times that *simpleFoam* has gone through its iterative process to achieve the solution. Hence the time-step for *simpleFoam*, and indeed all steady state solvers, is set to 1.

2.2.6 - Simulation Controls

Time, write, and post-processing controls for the simulation are accessed through the *controlDict* file in the *system* directory. Controls such as the start-time, end-time, time-steps, write-steps, etc. can be controlled through the parameters in this file. A *controlDict* file from this work is attached in Appendix A.4. For this work, each of the simulations were run for 500 time-steps, and solutions written every 25 time-steps. Since it is a steady state simulation, the only time-step of interest is the final one, as it contains a solution closest to the actual steady state solution. The convergence criterion enforced is that both velocity and pressure must be below 10^{-6}. The solutions did not converge in the 500 iterations that were computed.

2.2.7 - Post-processing tools

The desired results from these simulations are the force and moments generated by the fins and control surfaces. Here, we use the OF post-processing tools *forces* and *forceCoeffs* instead of ParaView to compute the forces and moments. *forces* computes pressure and viscous forces, and pressure and viscous moments about the X, Y, and Z-axes, and *forceCoeffs* computes the moment coefficient, lift coefficient, and drag coefficient about specified axes, which pass through the fish itself. These tools require an input patch to calculate the forces or force-coefficients on, which are specified in the file *forces* in the *system* directory. Since we require a force analysis over the entire (simplified) fish, the patch-group *fish* is provided as input. The post-processing tools then perform computations for every time-step and write results to a *postProcessing* directory. A *forces* file is attached in the Appendix A.5.

2.2.8 – Simulation Details

The simulations are carried out on remotely on an Intel Xeon computer belonging to the CFD lab of the Mechanical and Aerospace Engineering department of the University of California San Diego. The meshing process for each geometry takes four hours, and a solution of 500 iterations takes 24 hours to complete. One simulation, pertaining to one control surface configuration at one angle of descent, requires 30GB of memory, and the entire project requires close to 500GB of memory.

2.3 - Simulation Results – Force Assessment

The pertinent results from the simulations are the pressure forces and moments in the X, Y, and Z direction for each of the 3 control surface configurations across the 4 angles of attack. The results are given in Table 4. Figure 28 depicts the coordinate axes with respect to the fish in order to better interpret the force and moment results.

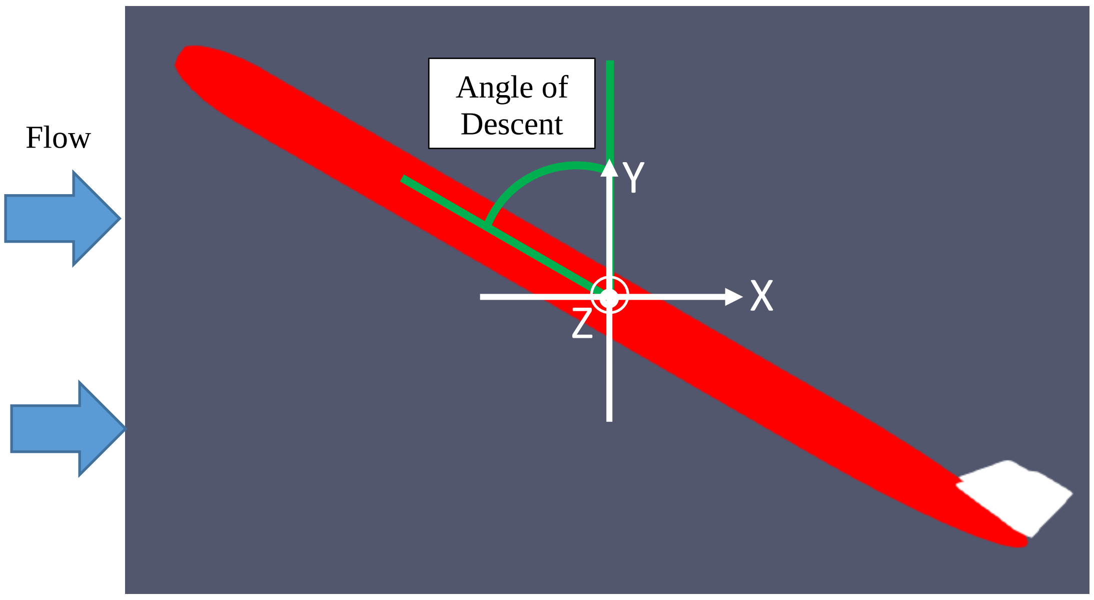


Figure 28 – Coordinate axis with respect to the fish

Table 4 – Simulation force and moments results

Angle of Descent (°)	Control Surface Config	Pressure Force (N)			Pressure Moment (Nm)		
		X	Y	Z	X	Y	Z
60	CSflat	167.26	292.44	1.48	-0.41	-0.55	99.49
65	CSflat	131.67	287.45	0.40	-0.14	0.05	96.85
70	CSflat	88.59	260.47	0.68	-0.15	-0.52	84.44
75	CSflat	51.78	212.12	-0.26	0.03	-0.07	60.94
60	CSminus10	174.69	308.41	-31.83	**7.40**	**18.85**	109.91
65	CSminus10	132.80	288.77	-21.61	**5.54**	**17.42**	95.74
70	CSminus10	90.82	262.32	-19.04	**4.41**	**18.72**	84.14
75	CSminus10	53.62	215.44	-20.25	**2.62**	**18.66**	62.34
60	CSplus10	178.11	308.87	27.17	**-10.25**	**-24.87**	105.46
65	CSplus10	133.26	290.36	18.64	**-4.58**	**-14.39**	97.51
70	CSplus10	94.97	267.48	18.90	**-3.90**	**-16.94**	84.80
75	CSplus10	53.38	215.21	18.13	**-2.36**	**-16.93**	62.13

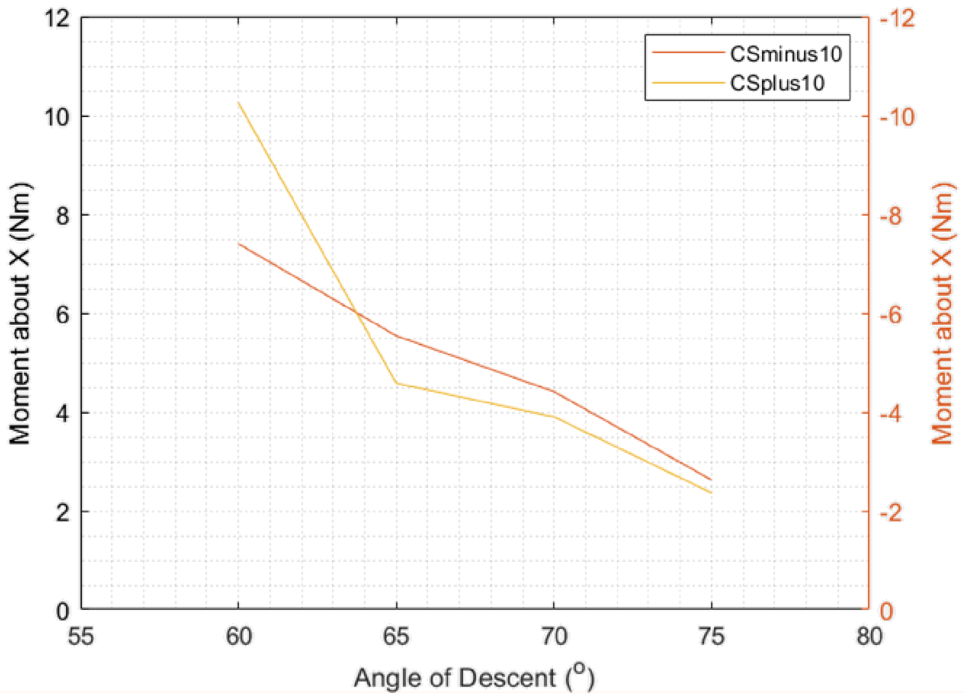


Figure 29 – Decrease in rotating moment with increase in angle of attack

The moment associated with the (excessive) rotation of the fish, is the moment about the X axis (marked in red for CSminus10 and yellow for CSplus10 on Table 4 and on Figure 29). This moment corresponds to the vertical moment that is discussed in the optimization strategy section at the end of the data analysis chapter (Section 1.8). It can be seen from the results that increasing the angle of descent, or steepening the descent of the fish, decreases the vertical moment. The moment for the CSminus10 configuration at an angle of 60° is 7.40 Nm and at an angle of 75° is 2.62 Nm. The same behavior can be observed in the CSplus10 configuration, where the magnitude of the moment decreases from 10.25 Nm to 2.36 Nm (sign is negative to indicate rotation in opposite direction). From Figure 29, the relation between the rotating moment and the angle of descent appears to be linear. This change in vertical moment would directly result in a decrease in the intensity of rotation of the fish on a down-cast. If we assume that the magnitude of the moment at 60° maps directly to the rotation rate of 0.05Hz found from the data analysis, then the moment at 75° will result in a reduction in the rotation rate of the fish by a factor of 3-4. Another inference drawn is that the horizontal moment (marked in blue for CSminus10 and in green for CSplus10 on Table 4), remains relatively constant across all angles of attack, which is likely due to the stabilizing effect of the fins opposing the moment caused by the control surfaces in steady state.

From these results, it is clear than an increase in the angle of descent is favorable in mitigating the intensity of rotation occurring in the fish. In order to achieve this, there needs to be a greater lift force generated on the tail of the fish than is currently being produced. To that end, two design changes involving the fins are investigated in the following section.

2.4 - Incremental Design Changes

The objective of this section is to extract more lift from the fins without any major changes to the parts or to the assembly. This would minimize the implementation cost and time. Two design changes are proposed:

- Rotation of each fin about their own central axis by 10° – referred to as *CSmod1* (depicted in Figure 30)
- Rotation of each fin about the coupling line with the tail-cone by 10° – referred to as *CSmod2* (depicted in Figure 31)

Implementation of these design changes is straightforward, theoretically, as the same tail-cone and fins can be used. The only difference would be a change in the assembly of the fin to the tail-cone.

The changes are implemented in the original SolidWorks file of the fish and the following procedure of CAD pre-processing, meshing, simulation, and post-processing remain the same. The modified fishes are only simulated at an angle of descent of 60°, with the control surfaces in a neutral position.

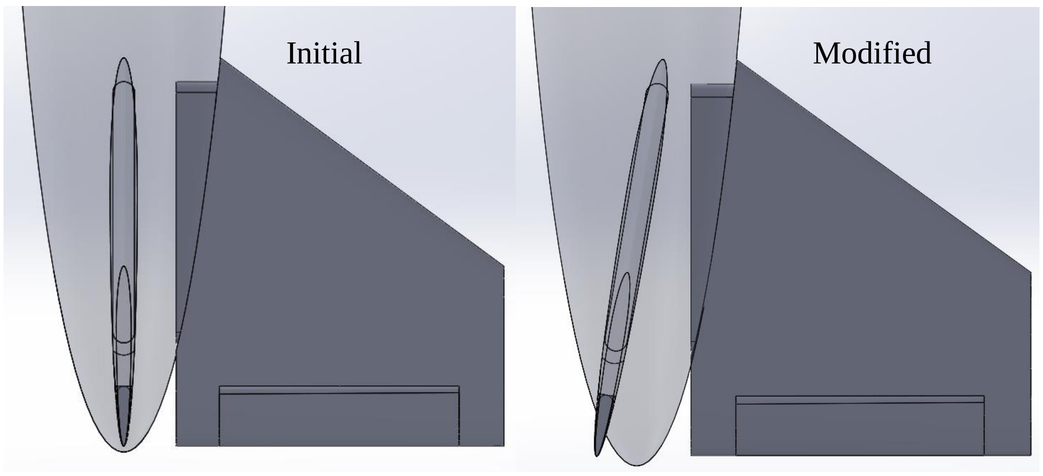
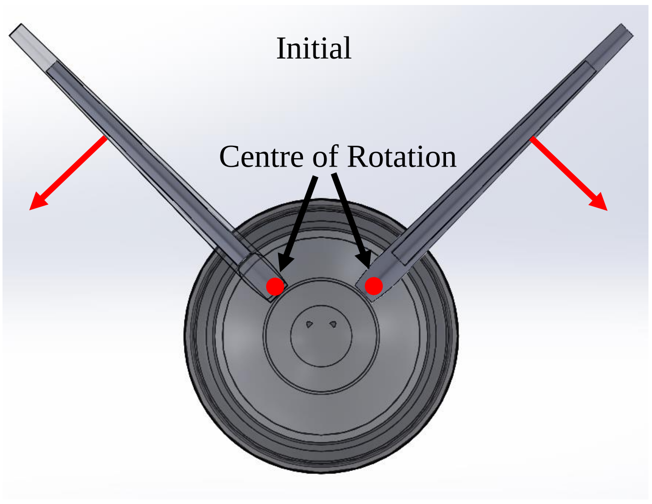
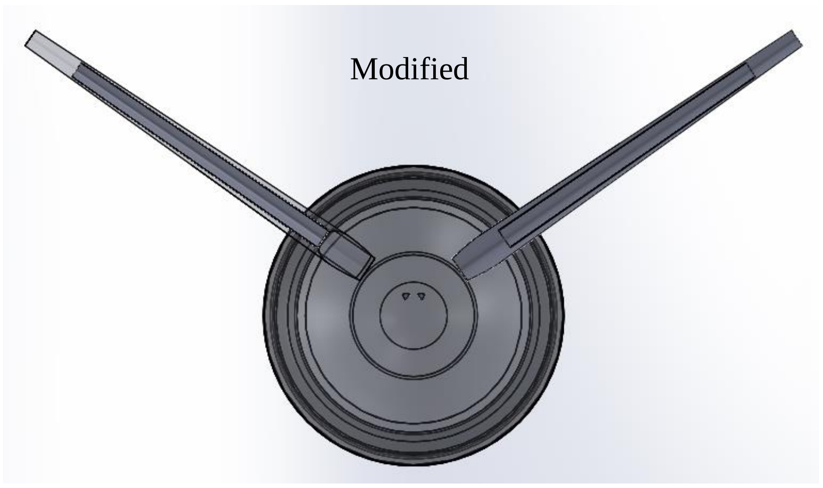


Figure 30 – *CSmod1* design change to the fins, rotation about fin's vertical axis



31(a)



31(b)

Figure 31 – (a) Depiction of fin rotation about coupling point and (b) *CSmod2* design change to the fins

2.5 - Design Change Simulation Results – Force Assessment

The results from the simulations of the design changes are compared with the base case simulation, which is neutral control surfaces at an angle of descent of 60°. The force and moment results are given in Table 5, and the force-coefficient results are given in Table 6.

Table 5 – Design change simulation force and moment results

Angle of Descent (°)	Control Surface Config	Pressure Force (N)			Pressure Moment(Nm)			ΔZ mom
		X	Y	Z	X	Y	Z	
60	CSflat	167.26	292.44	1.48	-0.41	-0.55	**99.49**	**-**
60	CSmod1	198.05	267.32	0.94	-0.68	-1.35	**100.99**	**1%**
60	CSmod2	178.43	301.49	-4.04	1.55	3.70	**118.90**	**20%**

Table 6 – Design change simulation force coefficient results

Angle of Descent (°)	Control Surface Configuration	Moment Coefficient C_m	Drag Coefficient C_d	Lift Coefficient C_l
60	CSflat	**0.91**	**0.32**	**5.39**
60	CSmod1	0.92	0.92	5.29
60	CSmod2	**1.09**	**0.39**	**5.60**

The moment associated with the angle of descent of the fish is the pressure moment about the Z axis (highlighted in green), which is also referred to as the pitching moment. It is found that the first design change (CSmod1), rotation of the fin about its central axis, does not appreciably change the pitching moment when compared to the original fish (CSflat). The second design change (CSmod2) however, appears to have close to 20% greater pitching moment, when compared to the original fish, which would increase the angle of descent of the fish. This result can also be seen from the force coefficient results, as the moment coefficient (also called pitching coefficient) of the modified fish is 1.09 and the original fish is 0.91

(highlighted in blue). Also, the modified fish has a greater lift coefficient (5.60 vs 5.39, highlighted in purple). However, the second design change does have an increased drag coefficient (0.39 vs 0.32, highlighted in red). This is expected as it is generating more lift. This increased drag should be compensated by the fact that the angle of descent will increase, making the fish more parallel to the flow and thus reducing drag forces. Overall, it can be seen that the second design change produces favorable results towards the goal of increasing the angle of descent of the fish.

Conclusion

The aim of this thesis has been to understand and quantify the rotations occurring in the deployment of the FastCTD profiling body, and initiate the first steps in addressing them. The study has been carried out using onboard data analysis to learn about the nature of the rotations occurring, and come up with a preliminary hypothesis for mitigation, which was the increase the angle of descent of the fish. This hypothesis has been put to the test by performing a computational fluid dynamics analysis using the software OpenFOAM, and investigating the forces and moments that affect the fish, and their relation to the angle of descent. Following this force assessment, two straightforward design changes are proposed, and simulated. The findings of the analysis are as follows:

- The rotations in the fish that lead to twisting of the cable occur only about a vertical axis, no rolling occurs.
- A spin counter has been developed based on a rotated coordinate system of the IMU. This utility will be added to the real-time control software of the FastCTD to monitor the rate of rotation of the fish and the status of the tow cable.
- Two modes of rotation occur, a dominant primary mode, responsible for the excessive twisting of the tow cable, and a sporadically occurring secondary mode.
- The hypothesis of increasing the angle of descent is validated by the force assessment from the simulations. Results indicate a possible 4x decrease in the rate of rotations occurring, with an increase in the angle of descent from 60° to 75°.
- The second proposed design change, involving rotation of the fins about the coupling line, generates 20% greater pitching moment than the current fin configuration, at 60° angle of descent.
- Increased pitching moment would lead directly to an increase in angle of descent, and in turn a decreased the rate of rotation.

The validity of the results obtained in this work will need to be implemented in a future deployment of the FastCTD in order to confirm their effectiveness. However, it should be considered that several simplifications have been made in order to perform a first order analysis of the rotations occurring. The simplifications made are listed below:

- The effect of the tow cable tension on the flight of the fish has been ignored.
- Straight-path free-fall has been assumed for the force assessment.
- Several parts removed from the CAD model of the fish.
- The effect of winch and cable forces on the up-cast has not been considered for the simulation.
- The effect of the moment produced by the center of mass about the tow point ignored.

Future work may focus on collecting tow point force data from experiments, performing a more sophisticated simulation involving mechanisms for external forces that vary in time and consideration of mass and center of gravity for force assessments. A combination of lab-based studies and field testing of results could iteratively lead to a perfected design of the FastCTD.

References

[1] Science Learning Hub - Pokapū Akoranga Pūtaiao. [2010]. *Ocean Density*. Retrieved from https://www.sciencelearn.org.nz/resources/687-ocean-density

[2] Subrata Chakrabarti. [2005]. *Handbook of Offshore Engineering*. Volume 1

[3] St. Laurent, L., M.H. Alford, and T. Paluszkiewicz. [2012]. *An introduction to the special issue on internal waves*. Oceanography 25(2):15–19, http://dx.doi.org/10.5670/oceanog.2012.37

[4] S. A. Thorpe. [2007]. *An Introduction to Ocean Turbulence*. Cambridge University Press.

[5] Shroyer E.L., Nash J.D., Waterhouse A.F., Moum J.N. [2018]. *Measuring Ocean Turbulence*. In: Venkatesan R., Tandon A., D'Asaro E., Atmanand M. *Observing the Oceans in Real Time*. Springer Oceanography. Springer, Cham

[6] J.S.M. Coleman, K.T. Law. [2015]. *Reference Module in Earth Systems and Environmental Sciences*. Retrieved from https://www.sciencedirect.com/topics/earth-and-planetary-sciences/meteorological-phenomenon

[7] Seabird Scientific. [2018]. *AUV/ROV/Glider Sensors*. CTD Brochure retrieved from https://www.seabird.com/moving-platform/sbe-49-fastcat-ctd-sensor/family-downloads?productCategoryId=54627473793

[8] Yost Labs. [2017]. *3-Space Sensor Embedded – User Manual*. Retrieved from https://yostlabs.com/wp/wp-content/uploads/pdf/3-Space-Sensor-Users-Manual-Embedded-1.pdf

[9] Wikepedia. [n.d.]. *Rodrigues' Rotation Formula*. Retrieved from https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula

[10] Natural Resources Canada. [n.d.]. *Magnetic Field Calculator*. Retrieved from https://geomag.nrcan.gc.ca/calc/mfcal-r-en.php?date=2019-05-08&latitude=40&latitude_direction=-1&longitude=160&longitude_direction=1

[11] MathWorks Documentation. [n.d.]. *Welch's power spectral density estimate*. Retrieved from https://www.mathworks.com/help/signal/ref/pwelch.html

[12] OpenFOAM Foundation. [2018]. *OpenFOAM User Guide – Version 6*.

[13] National Physics Laboratory, [2017]. *Kaye and Laby Tables of Physical and Chemical Constants*. Retrieved from http://www.kayelaby.npl.co.uk/general_physics/2_7/2_7_9.html

[14] ANSYS Inc. [2017]. *ANSYS Fluent Theory Guide Release 17.0*. Retrieved from https://www.sharcnet.ca/Software/Ansys/17.0/en-us/help/flu_th/th-x1-20001.1.html

[15] Passalacqua, A. [2014]. *OpenFOAM guide/the simple algorithm in OpenFOAM*. Retrieved from https://openfoamwiki.net/index.php/OpenFOAM_guide/The_SIMPLE_algorithm_in_OpenFOAM.

Appendix

A.1 - Sample blockMeshDict file:

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
    (-6 -6 -6)
    (15 -6 -6)
    (15 6 -6)
    (-6 6 -6)
    (-6 -6 6)
    (15 -6 6)
    (15 6 6)
    (-6 6 6)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (60 24 24) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    frontAndBack
    {
        type patch;
        faces
        (
            (3 7 6 2)
            (1 5 4 0)
        );
    }
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (2 6 5 1)
        );
    }
    lowerWall
    {
        type patch;
        faces
        (
            (0 3 2 1)
        );
    }
    upperWall
    {
        type patch;
        faces
        (
            (4 5 6 7)
        );
    }
);

// ************************************************************************* //
```

A.2 - Sample snappyHexMeshDict file:

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       true;
```

```
// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    fish
    {
        type triSurfaceMesh;
        file "fish.obj";
       regions
       {
              fish
              {
                     name body;
              }
              fish.001
              {
                     name fins;
              }
       }
    }

    refinementBox
    {
        type searchableBox;
        min (-2 -2 -2);
        max (10 2 2);
    }
};



// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 1500000;

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
    // actually be a lot less.
    maxGlobalCells 20000000;

    // The surface refinement loop might spend lots of iterations refining just a
    // few cells. This setting will cause refinement to stop if <= minimumRefine
    // are selected for refinement. Note: it will at least do one iteration
```

```
// (unless the number of cells to refine is 0)
minRefinementCells 8;

// Allow a certain level of imbalance during refining
// (since balancing is quite expensive)
// Expressed as fraction of perfect balance (= overall number of cells /
// nProcs). 0=balance always.
maxLoadUnbalance 0.10;


// Number of buffer layers between different levels.
// 1 means normal 2:1 refinement restriction, larger means slower
// refinement.
nCellsBetweenLevels 8;



// Explicit feature edge refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

// Specifies a level for any cell intersected by its edges.
// This is a featureEdgeMesh, read from constant/triSurface for now.
features
(
    {
        file "fish.eMesh";
        level 8;
    }
);



// Surface based refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~

// Specifies two levels for every surface. The first is the minimum level,
// every cell intersecting a surface gets refined up to the minimum level.
// The second level is the maximum level. Cells that 'see' multiple
// intersections where the intersections make an
// angle > resolveFeatureAngle get refined up to the maximum level.

refinementSurfaces
{
    fish
    {
        // Surface-wise min and max refinement level
        level (9 10);

        // Optional specification of patch type (default is wall). No
        // constraint types (cyclic, symmetry) etc. are allowed.
    }
}

// Resolve sharp angles
resolveFeatureAngle 30;


// Region-wise refinement
```

```
    // ~~~~~~~~~~~~~~~~~~~~~~

    // Specifies refinement level for cells in relation to a surface. One of
    // three modes
    // - distance. 'levels' specifies per distance to the surface the
    //   wanted refinement level. The distances need to be specified in
    //   descending order.
    // - inside. 'levels' is only one entry and only the level is used. All
    //   cells inside the surface get refined up to the level. The surface
    //   needs to be closed for this to be possible.
    // - outside. Same but cells outside.

    refinementRegions
    {
        refinementBox
        {
            mode inside;
            levels ((1E15 8));
        }
    }


    // Mesh selection
    // ~~~~~~~~~~~~~~

    // After refinement patches get added for all refinementSurfaces and
    // all cells intersecting the surfaces get put into these patches. The
    // section reachable from the locationInMesh is kept.
    // NOTE: This point should never be on a face, always inside a cell, even
    // after refinement.
    locationInMesh (14.0001 5.0001 5.0001);


    // Whether any faceZones (as specified in the refinementSurfaces)
    // are only on the boundary of corresponding cellZones or also allow
    // free-standing zone faces. Not used if there are no faceZones.
    allowFreeStandingZoneFaces true;
}



// Settings for the snapping.
snapControls
{
    //- Number of patch smoothing iterations before finding correspondence
    //  to surface
    nSmoothPatch 3;

    //- Relative distance for points to be attracted by surface feature point
    //  or edge. True distance is this factor times local
    //  maximum edge length.
    tolerance 2.0;

    //- Number of mesh displacement relaxation iterations.
    nSolveIter 70;

    //- Maximum number of snapping relaxation iterations. Should stop
    //  before upon reaching a correct mesh.
```

```
    nRelaxIter 8;

    // Feature snapping

        //- Number of feature edge snapping iterations.
        //  Leave out altogether to disable.
        nFeatureSnapIter 8;

        //- Detect (geometric only) features by sampling the surface
        //  (default=false).
        implicitFeatureSnap true;

        //- Use castellatedMeshControls::features (default = true)
        explicitFeatureSnap true;

        //- Detect points on multiple surfaces (only for explicitFeatureSnap)
        multiRegionFeatureSnap false;
}



// Settings for the layer addition.
addLayersControls
{
    // Are the thickness parameters below relative to the undistorted
    // size of the refined cell outside layer (true) or absolute sizes (false).
    relativeSizes true;

    // Per final patch (so not geometry!) the layer information
    layers
    {
    //    {
            nSurfaceLayers 1;
    //    }
    }

    // Expansion factor for layer mesh
    expansionRatio 1.0;

    // Wanted thickness of final added cell layer. If multiple layers
    // is the thickness of the layer furthest away from the wall.
    // Relative to undistorted size of cell outside layer.
    // See relativeSizes parameter.
    finalLayerThickness 0.3;

    // Minimum thickness of cell layer. If for any reason layer
    // cannot be above minThickness do not add layer.
    // Relative to undistorted size of cell outside layer.
    minThickness 0.1;

    // If points get not extruded do nGrow layers of connected faces that are
    // also not grown. This helps convergence of the layer addition process
    // close to features.
    nGrow 0;

    // Advanced settings

    // When not to extrude surface. 0 is flat surface, 90 is when two faces
    // are perpendicular
    featureAngle 60;

    // At non-patched sides allow mesh to slip if extrusion direction makes
    // angle larger than slipFeatureAngle.
    slipFeatureAngle 30;

    // Maximum number of snapping relaxation iterations. Should stop
    // before upon reaching a correct mesh.
    nRelaxIter 8;

    // Number of smoothing iterations of surface normals
    nSmoothSurfaceNormals 3;

    // Number of smoothing iterations of interior mesh movement direction
    nSmoothNormals 3;

    // Smooth layer thickness over surface patches
    nSmoothThickness 8;

    // Stop layer growth on highly warped cells
    maxFaceThicknessRatio 0.5;

    // Reduce layer growth where ratio thickness to medial
    // distance is large
    maxThicknessToMedialRatio 0.3;

    // Angle used to pick up medial axis points
    // Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in 17x.
    minMedianAxisAngle 90;


    // Create buffer region for new layer terminations
    nBufferCellsNoExtrude 0;


    // Overall max number of layer addition iterations. The mesher will exit
    // if it reaches this number of iterations; possibly with an illegal
    // mesh.
    nLayerIter 60;

    nRelaxedIter 40;
}



// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    #include "meshQualityDict"
}


// Advanced

// Write flags
writeFlags
```

```
(
    scalarLevels
    layerSets
    layerFields     // write volScalarField for layer coverage
);


// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1e-6;


// ************************************************************************* //
```

A.3 – Velocity and Pressure field files

Velocity:

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#include        "include/initialConditions"

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (4 0 0);

boundaryField
{
    //- Set patchGroups for constraint patches
    #includeEtc "caseDicts/setConstraintTypes"

    inlets
    {
        type            freestreamVelocity;
        freestreamValue $internalField;
    }

    outlet
    {
        type            freestreamVelocity;
        freestreamValue $internalField;
    }
    fish
    {
        type            noSlip;
    }
}

// ************************************************************************* //
```

Pressure:

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#include        "include/initialConditions"

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    //- Set patchGroups for constraint patches
    #includeEtc "caseDicts/setConstraintTypes"

    inlets
    {
        type            freestreamPressure;
        freestreamValue $internalField;
    }

    outlet
    {
        type            freestreamPressure;
        freestreamValue $internalField;
    }

    fish
    {
        type            zeroGradient;
    }
}
```

```
// ************************************************************************* //
```

A.4 – controlDict file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         500;

deltaT          1;

writeControl    timeStep;

writeInterval   25;

purgeWrite      0;

writeFormat     binary;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  no;

maxCo           0.2;

functions
```

```
{
    #include "forces"
}

// ************************************************************************* //
```

A.5 – forces and forceCoeffs file

```
 /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (fish);
    rho             rhoInf;      // Indicates incompressible
    rhoInf          1029;           // Redundant for incompressible
    liftDir         (0.5 0.866 0);
    dragDir         (0.866 -0.5 0);
    CofR            (0 0 0);  // Axle midpoint on ground
    pitchAxis       (0 0 1);
    magUInf         4;
    lRef            1.714;        // length scale
    Aref            0.007854;        // Estimated
    /*
    binData
    {
        nBin        20;          // output data into 20 bins
        direction   (1 0 0);     // bin direction
        cumulative  yes;
    }
    */
}

forces
{
    type            forces;
    libs            ("libforces.so");
    writeControl    timeStep;
    timeInterval    1;
    log             yes;
    patches         (fish);
    rho             rhoInf;      // Indicates incompressible
    rhoInf          1029;           // Redundant for incompressible
    CofR            (0 0 0);  // point of rotation for moments
}

// ************************************************************************* //
```