

# Pre-processing of Experimental Signals for Condition Monitoring (CM) purposes.

Aiden D. Devine

Student Number: 1618454

Cardiff University

**Total number of words = 6157**

**Total number of pages = 28**

## Abstract:

The Cardiff Marine Energy Research Group (CMERG) have collected a large array of data from laboratory testing of 1/20<sup>th</sup> scale model Tidal Stream Turbines (TST's). This report details the development of a 'TST dataset pre-processing tool' designed in MATLAB capable of resampling angular encoder signals allowing encoder position and torque measurement data to share a common sample rate. The report also investigates methods of denoising torque data to remove high frequency noise and the development of the tool's user interface. The Pre-processing methods employed are intended to allow further condition monitoring analysis to be conducted using the processed data.

I hereby declare:

**that** except where reference has clearly been made to work by others, all the work presented in this report is my own work;

**that** it has not previously been submitted for assessment; and

**that** I have not knowingly allowed any of it to be copied by another student.

I understand that deceiving or attempting to deceive examiners by passing off the work of another as my own is plagiarism. I also understand that plagiarising the work of another or knowingly allowing another student to plagiarise from my work is against the University regulations and that doing so will result in loss of marks and possible disciplinary proceedings against me.

Signed: A. Devine..... (electronic)

Date: 06/05/21.....

## Contents

1. Introduction .....	1
2. Previous work .....	2
3. The brief .....	3
4. Theory .....	4
4.1 Nyquist Shannon sampling theorem .....	4
4.2 Resampling .....	4
4.3 Moving Average Filter.....	4
4.4 Linear interpolation and Extrapolation.....	4
4.5 Polyfit.....	5
5. Enabling Technologies .....	5
6. Development.....	6
6.1 Process of developing Code .....	6
6.2 Process of developing User Interface.....	10
6.3 Data used for Testing.....	12
7. Discussion.....	12
8. Conclusion .....	16
9. References.....	16
10. Appendices .....	16
Appendix 1- Nomenclature.....	16
Appendix 2- Code and Data .....	17
Datasets.....	17
Code .....	18
Record of project meetings .....	24

## 1. Introduction

With global temperatures likely to reach 1.5°C between 2030 and 2052 (Allen et al. 2018) many governments around the world have acted by setting goals to reduce greenhouse gas production, this reduction will require the improvement of existing sources of renewable energy as well as the introduction of new untapped energy sources, this involves developing methods of harvesting energy from the earth's natural resources e.g., the Sun, wind, and water. Some sources of renewable energy are more reliable than others, the sun may not always shine on solar panels or the wind may not always blow sufficiently for wind turbines, the use of water to generate energy was conceived thousands of years ago and has always been a reliable source of power. While most water reliant techniques are based on the flow of rivers there are also early examples of harnessing the power of the tide. As fluctuations in tide levels are induced by the orbit of the moon its rhythms can be predicted very precisely. Early example of Tidal power can be seen in the restored medieval tidal mills surrounding the U.K.'s coast, these sites would capture water in lagoons at high tide and use the stored water to mill flour during low tides (TidalPower 2013). In a similar era, windmills were developed to harness the power of gusts using sails which could be considered as precursors to the wind turbine blades that are being developed today.

The progress of wind turbines has erupted over the last century, with a myriad of new techniques being developed to optimize aerodynamics and efficiency. As turbines began to be more widely built and operated in more remote and treacherous environments the need for methods of condition monitoring became apparent. Unlike conventional fossil fuel power stations operating around a centralized turbine, renewable energy production often requires generators to be spread across a wide area, this presents a different logistical challenge in terms of condition monitoring as it must be performed remotely, in the case of wind turbines CM instrumentation is built into the nacelles of turbines. The combination of the two technologies of Wind and tidal power have resulted in the emerging field of Horizontal Axis Tidal Stream Turbines (TSTs) these devices are intended to be attached to the seabed in coastal areas specifically selected for their strong and reliable tidal currents, deep sea currents will drive the blades of turbines much like the wind turbines which predate them, these devices hope to produce clean reliable energy all year round. The pay-off for this, seemingly perfect, untapped energy source is the treacherous environment TSTs will be operating within. The complex loading environment that must be considered involves the effects of surface waves, turbulent flows, collisions with underwater debris and angled flows to name a few. To effectively monitor and maintain a farm of underwater turbines, a network

of sensors must be used to build a picture of the condition of the machinery which is mostly invisible and inaccessible to its caretakers on dry land.

Tidal energy technology has yet to be proven with regard to long term operational availability and reliability. It is accepted that the harsh marine environments and problems with accessibility for maintenance may exasperate availability and reliability problems. Minimising uncertainty surrounding the operation and maintenance of such devices will thus be crucial in improving investor confidence and achieving economically viable power extraction. (Allmark et al., 2015)

TST condition monitoring has become an increasingly relevant field in recent years, with CMERG focusing its work on the testing of scale model turbines in a variety of settings and recording a large repository of data from CFD simulations and real-world Tow and Flume tank testing. This project will focus on designing a menu selectable Pre-Processing tool to allow more efficient processing of CMERG's datasets so that further analysis can be more easily performed.

## 2. Previous work

Due to the specific nature of the project brief, combined with the infancy of the field of TST research, there is a limited field of previous work that is directly relevant to the project. There was however several years of research previously completed by CMERG. In 2015 Allmark et al conducted research on the use of time-frequency analysis for blade fault detection and diagnosis, Tests were performed using a parametric model in flume tank set-up at Liverpool University, experiments using a drive-shaft test rig were also conducted, by applying loads predicted by the CFD modelling of the turbine. It was later concluded that the use of time-frequency analysis was not a suitable method of rotor fault detection, with Allmark reporting that: For the time-frequency examples reported it was observed that the spectrogram plots were generally unsuccessful in detecting the offset blade fault. (Allmark et al., 2015) , however the research conducted had left more questions to be answered. Its primary function (the parametric turbine model), when initially deployed, was the validation of simulation study results. Although the measured data was not ideal for time-frequency analysis it was sufficient to instigate such investigations. (Allmark et al., 2015).

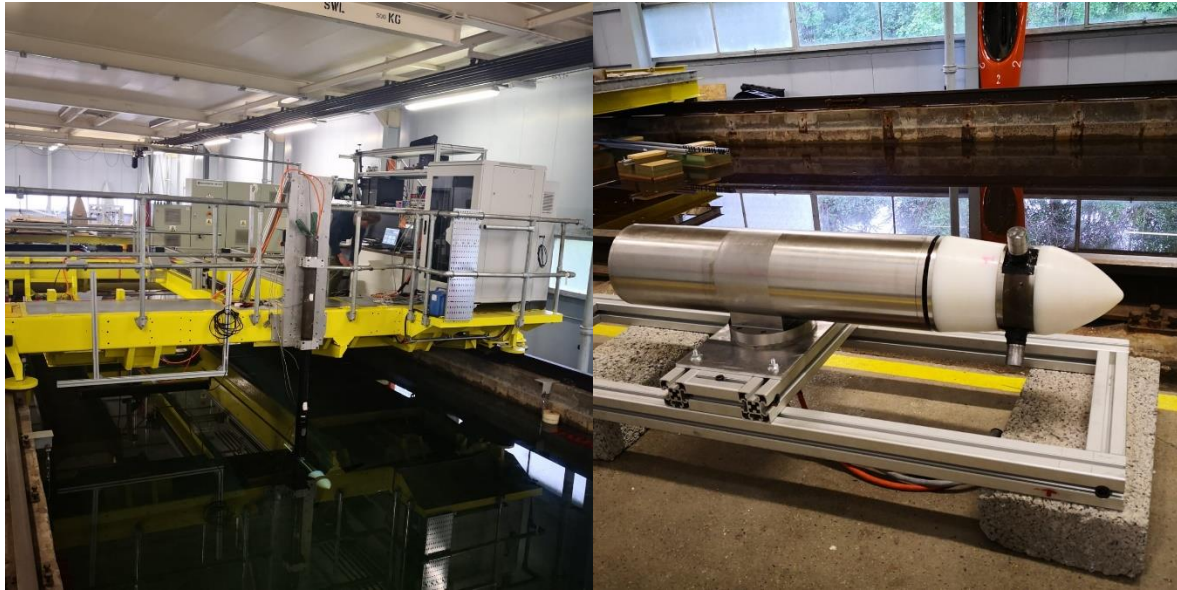


FIGURE 1. A) ACRE ROAD TOW TANK. B) CMERG'S MODEL TST.

The 2015 paper led to further research conducted by CMERG resulting in a second-generation turbine produced in 2016, in 2020 Allmark et al. published a subsequent paper detailing the development of CMERG's third generation of scale model TSTs (see figure 1b), these new models boasted a higher sample rate, additional torque and acceleration sensors and an angular encoder. A bespoke rotor torque and thrust transducer was created by Applied measurements Ltd. The transducer used was an adapted DBBSS/TSF Torque and Axial Force Sensor. The transducer was fastened between the front drive shaft and the turbine rotor upstream of any bearings or seals to measure rotor loads prior to any drive shaft losses. The encoder selected, and used for position feedback, was an optical encoder, the model utilised was the Heidenhain ENC113 encoder with Endat 2.2 interfacing. The encoder is of 13 bit type with a quoted system accuracy of  $\pm 20$  s of arc. (Allmark et al. 2020) The development and testing that followed resulted in a large body of experimental data recorded using the third-generation turbines, including the Tow tank testing conducted at the Acre Road facility in Glasgow (figure 1a).

### 3. The brief

The project brief proposed the development of a program to improve the ease and repeatability of pre-processing the datasets of TST testing collected in the CMERG repository. The pre-processing task involves reading in the Torque data collected in the turbines hub sensor array which has a sample rate of 2000Hz and matching the sample rates with the Optical Angular Encoder which is attached to the drive shaft which is sampled at a rate of 40Hz. The dataset can then be Split into individual rotations to allow further analysis to be more easily conducted. The data must also be normalised to bring the values

to an appropriate scale. Another useful features would be filtering the torque signal so that low frequency noise can be removed or reduced. A User Interface should also be designed to further improve the ease and repeatability of the task.

#### 4. Theory

##### 4.1 Nyquist Shannon sampling theorem

The sensors used to capture the datasets used in this project collected discrete data at their respective sample rates, when changing the sample rate of discrete data, it is important to consider the criterion of the Nyquist Shannon sampling theorem. For the impulse-sampled signal to form an accurate representation of the original signal, the sampling rate must be at least twice the highest frequency in the spectrum of the original signal (Alkin 2014). It was ensured that the conditions of this theorem were met throughout the processing of the data.

##### 4.2 Resampling

MATLAB's `resample` function allows data to be changed from one sample rate to another, the sample rate of the data is modified by multiplying the current sample rate of the data by the ratio provided by the user, interpolation is used to find the values of datapoints based on the existing points. The Function also applies a Low Pass Antialiasing filter.

##### 4.3 Moving Average Filter

Experimental data is often recorded with some level of high frequency noise present, to remove noise from experimental datasets some form of filter can be employed, A moving-average filter is a common method used for smoothing noisy data. MATLAB's `filter` function can be used to compute averages along a vector of data. A moving-average filter slides a window of length `windowSize` along the data, computing averages of the data contained in each window. The following difference equation defines a moving-average filter of a vector  $x$ :

$$y(n) = \frac{1}{\text{windowSize}} (x(n) + x(n-1) + \dots + x(n - (\text{windowSize} - 1))).$$

(MathWorks 2021)

##### 4.4 Linear interpolation and Extrapolation

MATLAB's `'interp1'` function allows values to be interpolated and extrapolated from a set of datapoints using several different methods of interpolation. The functions default method being linear interpolation, this method generates a linear equation to fit the data, this equation can then be used to relate values from the opposite axis of a plot. Cubic

interpolation can also be used, working in a similar way but instead fitting a cubic equation to the data. 'pchip' or Shape-preserving piecewise cubic interpolation builds on the cubic method by fitting many cubic equations across the dataset for more accurate representation of endpoints. The interpolated value at a query point is based on a shape-preserving piecewise cubic interpolation of the values at neighboring grid points (MathWorks 2021). 'pchip' may work better than linear and spline interpolation in some scenarios as it prevents 'over-shoots' and overestimates when extrapolating values. The effect of using different interpolation methods was investigated and has been discussed later in the report.

#### 4.5 Polyfit

The 'polyfit' function is MATLAB's polynomial curve fitting tool, it can be used to fit an equation of a chosen degree to a set of data, which can be useful as an alternative to filtering data.

### 5. Enabling Technologies

The crucial technology to enable this project was the development of the scale model TST which has been mentioned in the previous work section, as well as the sensors mounted to the turbine allowed precise measurements to be taken without obstructing the function of the model. Torque measurements are taken using a torque transducer attached to the drive shaft, the transducer was fastened between the front drive shaft and the turbine rotor upstream of any bearings or seals to measure rotor loads prior to any drive shaft losses. The transducer, which operates using strain gauge techniques, makes measurements by detecting the small changes in length between two points when the shaft undergoes torsion, this change in length results in a change in resistance when a current is running between the two points.

The angular measurements were taken using a Rotary Optical Encoder. Rotary encoders are used to measure the rotational motion of a shaft. Figure 2 shows the fundamental components of a rotary encoder, which consists of a light-emitting diode (LED), a disk, and a light detector on the opposite side of the disk. The disk, which is mounted on the rotating shaft, has patterns of opaque and transparent sectors coded into the disk. As the disk rotates, the opaque segments block the light and, where the glass is clear, light is allowed to pass. This generates square-wave pulses, which can then be interpreted into position or motion. (Encoder Measurements: How-To Guide - National Instruments. 2021)



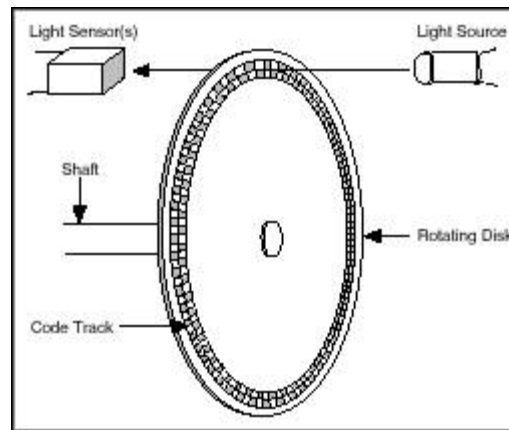


FIGURE 2. OPTICAL ENCODER COMPONENTS

## 6. Development

### 6.1 Process of developing Code

The first challenge in the development of the program was importing the data into vectors, each test run produced two files of readings, the Motor data file contains the values of angular encoder position in degrees along with time stamps for each reading, this allowed the data to be imported directly into respective angle and time vectors. There was an issue of some Motor data files having a different column structure, with some files having an extra column, however this was rectified with an 'if' statement to differentiate between the two variations. The angle time data was also recorded in milli-seconds, values were changed to seconds by dividing each time value by 1000 as it was read from the file. The Rotor data file containing the torque data recorded the timestamps in a date and time format which MATLAB could not recognize, therefore a time vector was created using the 'linspace' function with a vector the same length as the torque vector, equally spaced for each according to the sample rate. The code for importing data were written into two functions named 'importAngleData' and 'importTorqueData'.

To separate the dataset into individual rotations a method was devised using the 'islocalmin' and 'islocalmax' functions in conjunction to return a binary value for each index position of every local minimum and maximum values in the angle vector, the index positions with a value of 1 were stored in a two vectors 'spl' containing local minimums and 'sph' containing the local maximums relating to the first and last sample points for each rotation. In figure 3 angle was plotted against time in blue with the local maximums and minimums plotted in red.

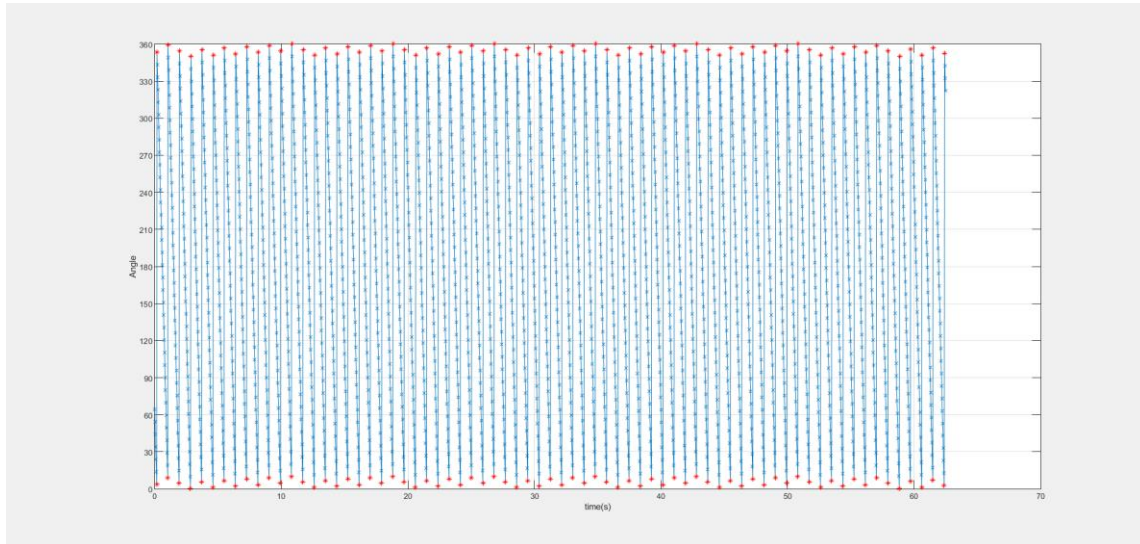


FIGURE 3. ANGLE ENCODER POSITION VS TIME

In the case of most datasets the raw torque signal displayed large amounts of high frequency noise. To improve clarity, it was decided that a moving-average filter should be employed, MATLAB's 'filter' function was used, a 1-Dimensional digital filter, after some experimentation a window size of 100 was selected as it provided the most clarity without being unrepresentative of the raw data. In figure 4 both the raw torque data (blue) and filtered data (red) have been plotted for comparison.

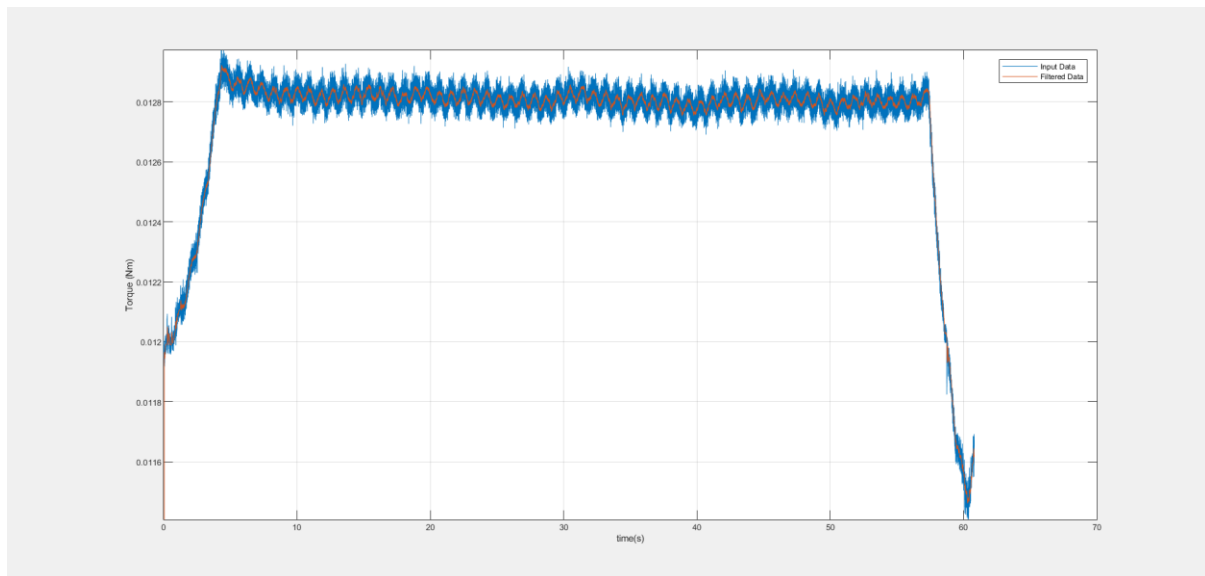


FIGURE 4. DRIVE SHAFT TORQUE VS TIME (BLUE: RAW DATA, RED: FILTERED DATA)

In the initial stages of development attempts were made to utilise MATLAB's 'Resample' function, during experimentation with this function an attempt was made to lower the torque signal to the sample rate of the angle signal, however due to the different sample rates not being directly compatible it was only possible to match the two rates by resampling both

signals to a common lower sample rate which had to be predetermined by the user, as this method of resampling had to be performed manually it was inappropriate for use in the final program as it did not simplify the task as the brief required. The resample function also only returned values with unpredictable time stamps when processing the torque signal, which did not match up to the timestamps of the angular encoder data. Using the resample function would also only provide values within the start and end points that angles were sampled, so the range of angle values could not be extended past the start and end samples of each rotation, to rectify these issues the 'interp1' function was then investigated.

It was decided that it would be more useful to up sample the angle signal to the higher sample rate of the torque signal, this would later allow more precise filtering of the torque function to made using other methods of denoising the data.

Using the 'interp1' function allowed the sample rate of the angular signal to not only be Up sampled but also allowed values to be extrapolated outside of the confines of the sampled range, by extrapolation values of angles could be calculated down to 0 degrees and up to 359 degrees (values were only extrapolated to 359 not 360 as the encoder positions 0 degrees and 360 degrees are equivalent) for each individual rotation in the dataset.

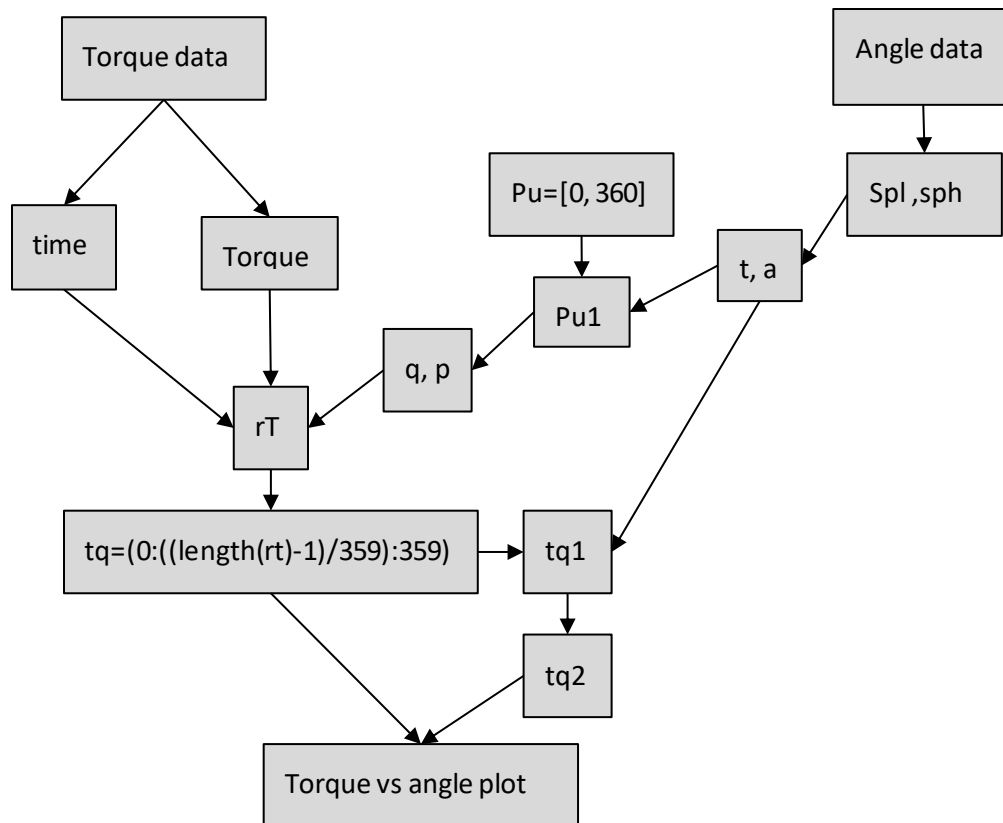


FIGURE 5. DATA PROCESSING FLOW CHART

For accurate resampling of the entire dataset, each rotation must be resampled individually. Using a 'For loop' to the following process was iterated between each local maximum and local minimum angle value stored in the 'spl' and 'sph' index vectors. First the 'interp1' function was used to extrapolate time values for the angles 360 and 0, these would act as the start and end times of the rotation currently being resampled. The start and end times were then used to find index value of start and end points in torque vector the 'find' function to search for the same time value in the torque signal. The equivalent start and end times were then used as limits of torque vector for given rotation. The sample points were then generated by creating a 'linspace' vector for angle values between 0 and 359. The sample point spacing of the vector was given by the equation 1:

$$\text{Sample Point Spacing} = \frac{(\text{Length}(rT)-1)}{359} \quad (1)$$

Where 'rT' is the torque vector of the rotation currently being resampled, thus giving the angle vector 'tq' sampled at the same rate as the torque signal, between the extended range of angles form 0-359 degrees.

Using 'tq' the 'interp1' function was used to return time values for each angle sample point. Then the 'interp1' was used to interpolate torque values for each new time value just calculated. The new torque vector was plotted against the new time values (see figure 6), however the torque vector had to be flipped before plotted as the angle encoder was operating in reverse order, flipping the vector made the plot more readable. A similar process was used to plot torque against time but instead resampled to single degree intervals, this was achieved using a simpler 'linspace' vector with only 359 sample points.

Figure 5 has been included to demonstrate the flow of data and use of variables through the program, the full version of the MATLAB code described has been made available in appendix 2.

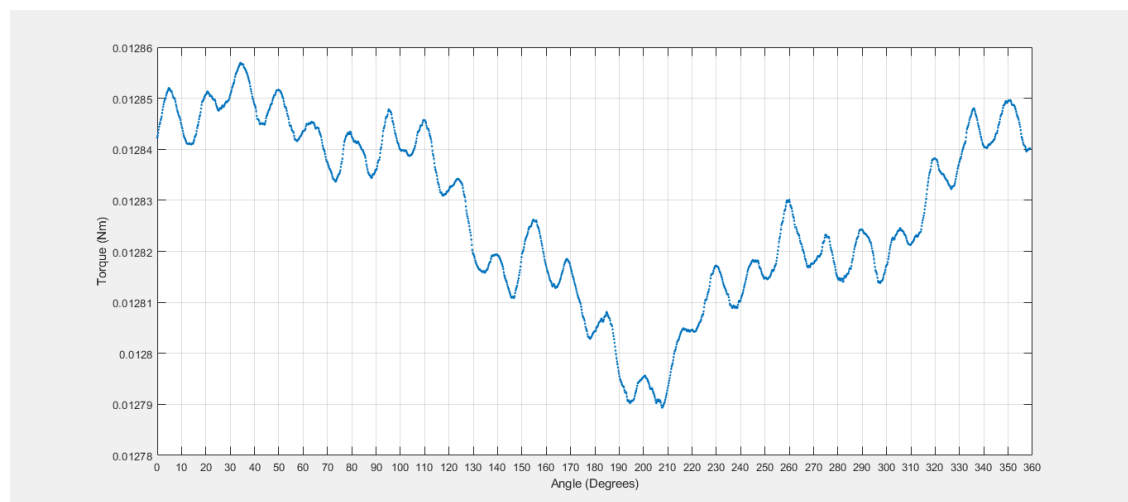


FIGURE 6. EXAMPLE OF ANGLE VS TIME PLOT

## 6.2 Process of developing User Interface

During development of the code, filenames had to be entered manually which was a time-consuming step when testing with different datasets. To improve ease of use a user interface was developed for the code that had been developed (figure 7). At a minimum, a suitable user interface would require a way for the user to import files and plot the relevant processed data. This was achieved using a MATLAB's 'uicontrol' command to generate a menu selectable interface, using a combination of editable text boxes to allow file paths to be manually entered, pushbuttons to trigger the 'uigetfile' function so that files can be selected from the user's document library and toggle buttons allow the user to load and plot the data by calling input and plot functions.

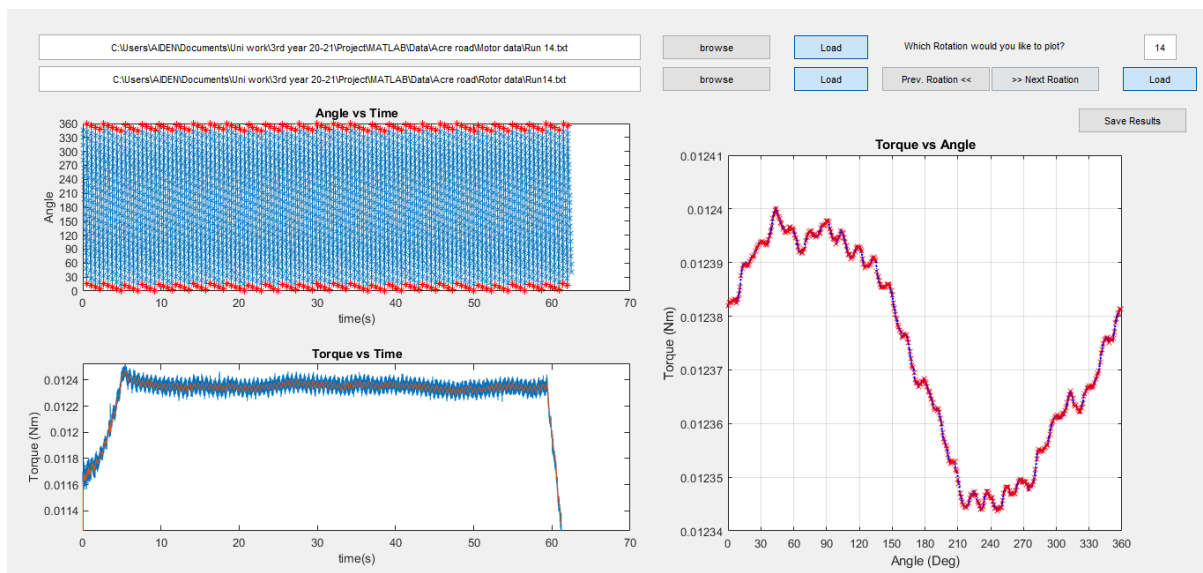


FIGURE 7. EXAMPLE OF USER INTERFACE LAYOUT

Further functionality was later added by giving the user the ability to move between individual rotation plots with the use of an editable text field allowing a specific rotation number to be entered or by moving between plots using previous and next rotation buttons. The option to output the resampled and filtered data to an output file was also added, this will create a single output file containing the torque, angular encoder position, and time value for sampled at the sample rate of the torque signal. This would allow the new data to be saved for later use so that further analysis could be performed without any further editing. For clarity, the flow chart in figure 8 has been included, detailing the intended path the user should take to interact with the program.

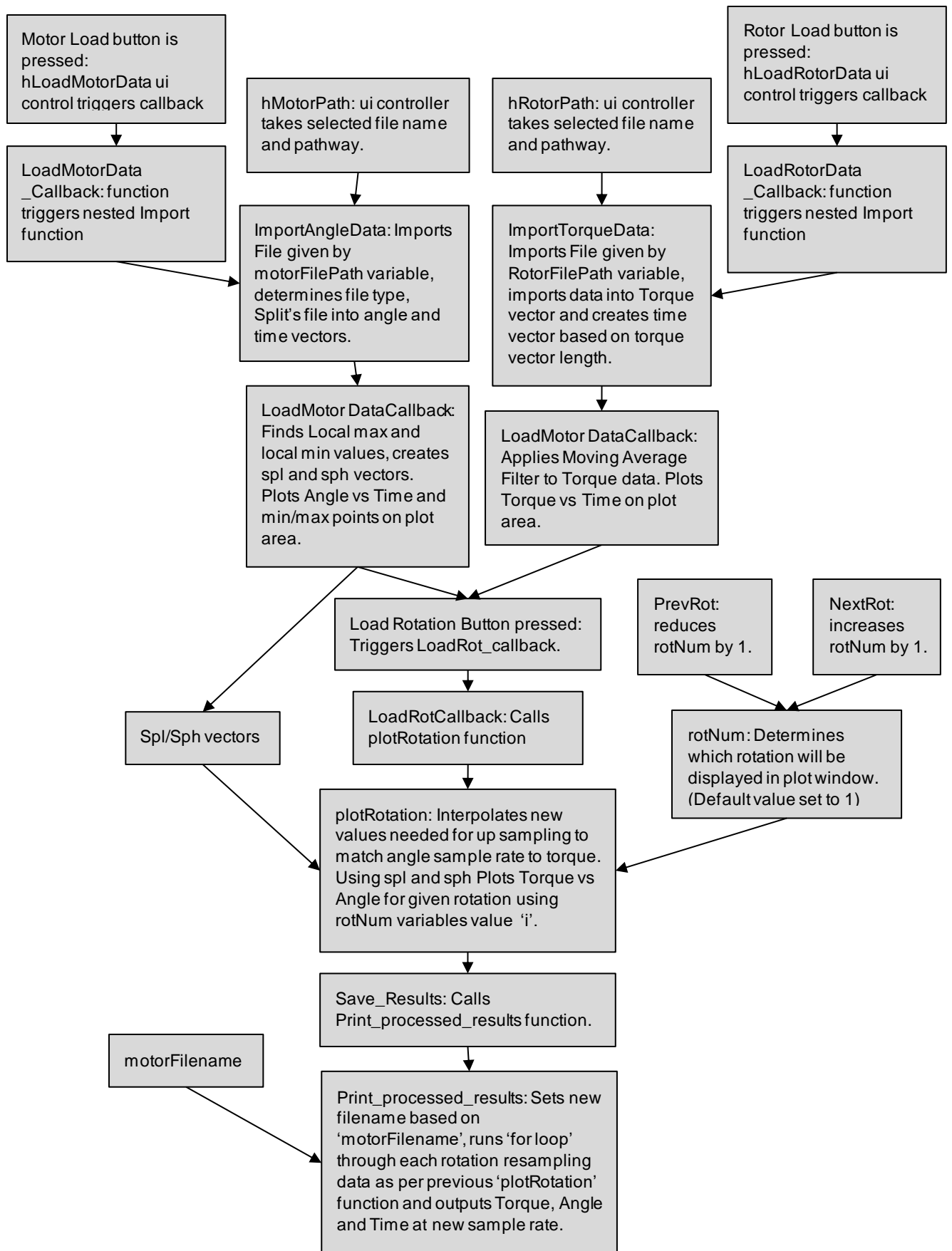


FIGURE 8- USER INTERFACE FLOW CHART

### 6.3 Data used for Testing.

The data the software was designed to analyze was primarily the repository of TST testing datasets collected by the CMERG over the several previous years, this data included tests from several sites: Glasgow university's tow tank at their Acre Road facility; Consiglio Nazionale delle Ricerche Institute of Marine Engineering (CNR- INM) wave-tow tank; Institut Francais de Recherche pour l'exploitation de la mer (IFREMER) re-circulating flume; and the Kelvin Hydrodynamic Laboratory (KHL) tow tank. These datasets were recorded using CMERG's three 0.9m diameter lab scale Horizontal Axis Tidal Turbines, these turbines were designed to investigate the effects of different loading environments. The software was tested primarily using a range of datasets recorded at the Acre Road test site. A selection of 20 datasets were used in testing, which were recorded when the turbine was operating at its most efficient tip speed ratio with no rotor imbalances or flow speed changes present. Using such data allowed the program to be tested with relatively normal signals with predictable output so that any issues with the plots could be more easily recognized instead of any anomalous data causing confusion during testing. Example datasets have been included in appendix 2.

## 7. Discussion

After processing some of the datasets, patterns were noticed in the data, once the rotor had reached its average speed for the dataset, the torque signal changed periodically with angle, this effect has been demonstrated in figure 9 by plotting rotations 7 to 64 of test run 10, for this plot a larger window size of 500 points was used when filtering the data to display the oscillation more clearly. As seen below torque fluctuates cyclically over every rotation, a pattern that is consistent with CFD models developed by (Allmark 2016) which were based on the turbine design which was later used to collect the data used in this project. The from the parallels between the CFD models and the plots produced by the Pre-Processing Tool, it can be deduced that the processing methods utilised have worked effectively.

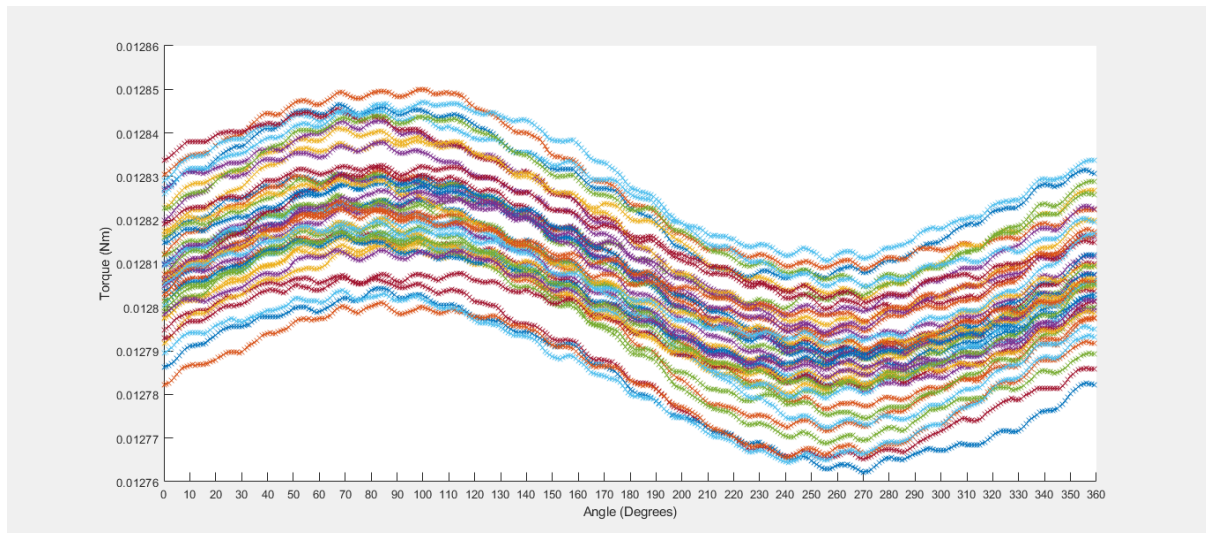


FIGURE 9. ANGLE VS TIME PLOTS OF ROTATIONS 17 TO 64 OF DATASET 10 (EACH ROTATION HAS BEEN PLOTTED IN A DIFFERENT COLOUR)

There was some discussion of it was preferable to up-sample or down-sample the data. If an attempt to raise the rate was made there were insufficient datapoints and plotting the data was impossible due to both vectors being different lengths. Successful attempts of using resample were only possible when resampling the data to a predetermined sample rate, where the number of sample points in both datasets were multiples of each other so that the vector lengths would match when plotting. This was due to the function only being able to compute the resample ratio if it consists of integer values. The interpolation method devised late was much more proficient as it allowed a full range of angle values to be calculated as well as allowing up-sampling of the angular encoder rate and down-sampling of both sensor rates. It can also be said that up-sampling removes the possibility of data loss when reducing points. The method devised also prevented aliasing as angular encoder position changed at an almost linear rate throughout the datasets.



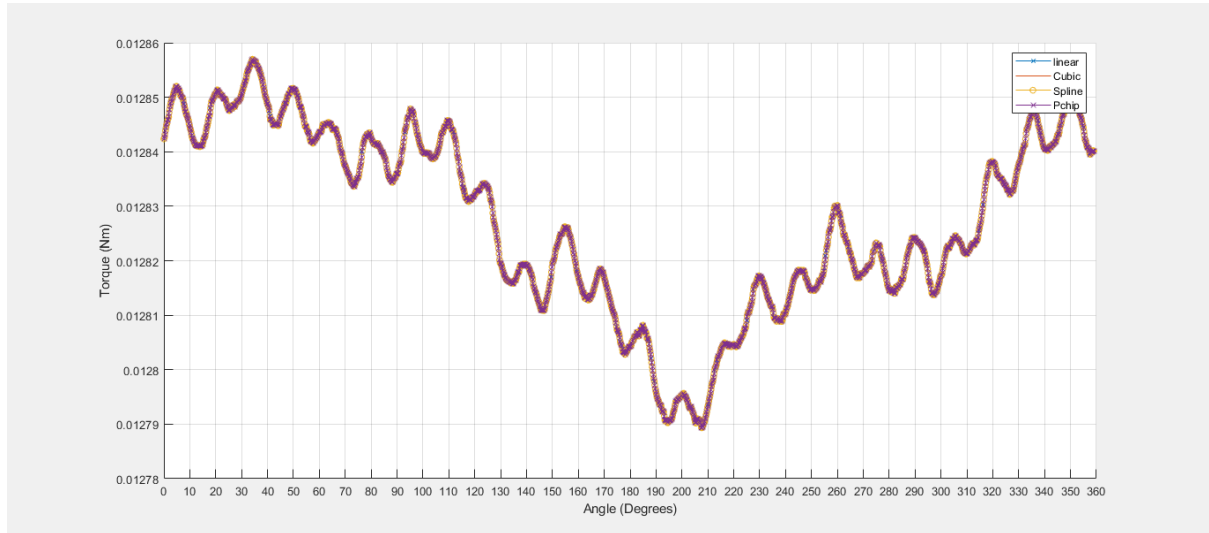


FIGURE 10. COMPARISON OF DIFFERENT INTERPOLATION TYPES.

To assess the effectiveness of using different methods of interpolation the resulting data of the linear, cubic, spline and pchip interpolation were plotted together in figure 10. The points all follow the same path showing that using a more complex interpolation type in the program would have negligible effect on the resulting plots, therefore the linear method of interpolation was selected to reduce the runtime and complexity of the software.

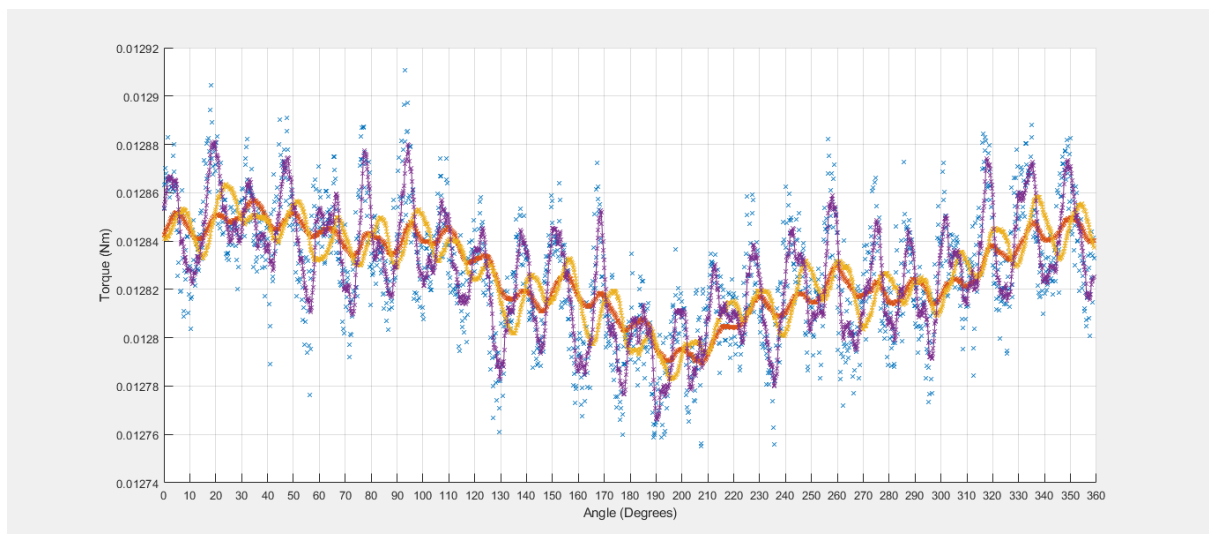


FIGURE 11. COMPARISON OF MOVING AVERAGE FILTER WINDOW SIZE.

The effect of different filter window sizes was also experimented with, a window size of 100 was selected to be used as the default window size for the moving average filter as it provided the most clarity without being unrepresentative of the raw data. As show in figure 11 the most extreme outliers of the data have been removed but the shape of the plot has not been fundamentally altered.

An alternative to using a moving-average filter would be to use the 'polyfit' function to plot a smoother curve, by fitting a polynomial to the data, low frequency noise will be removed, 'polyfit' was initially investigated using high order polynomials however a similar effect could be achieved in future research by fitting a sine wave to the data. A further addition to the software would be the giving the user the option to select which method of filtering is used to process the dataset, this could be achieved using a dropdown menu allowing the filter type to be selected (e.g.: moving-average, polyfit, unfiltered data) as well as the ability to specify any parametric values, such as the degree of polynomial or the window size of the moving-average filter. It may also be useful to subtract the mean values from the plots thus scaling the plots to the zero line so that the change in torque is shown. In figure 12 the effect of applying a moving-average filter has been compared to poly-fitting a 5<sup>th</sup> order degree polynomial to the same set of data, this graph illustrates the efficacy of poly-fitting when removing low frequency noise from signals.

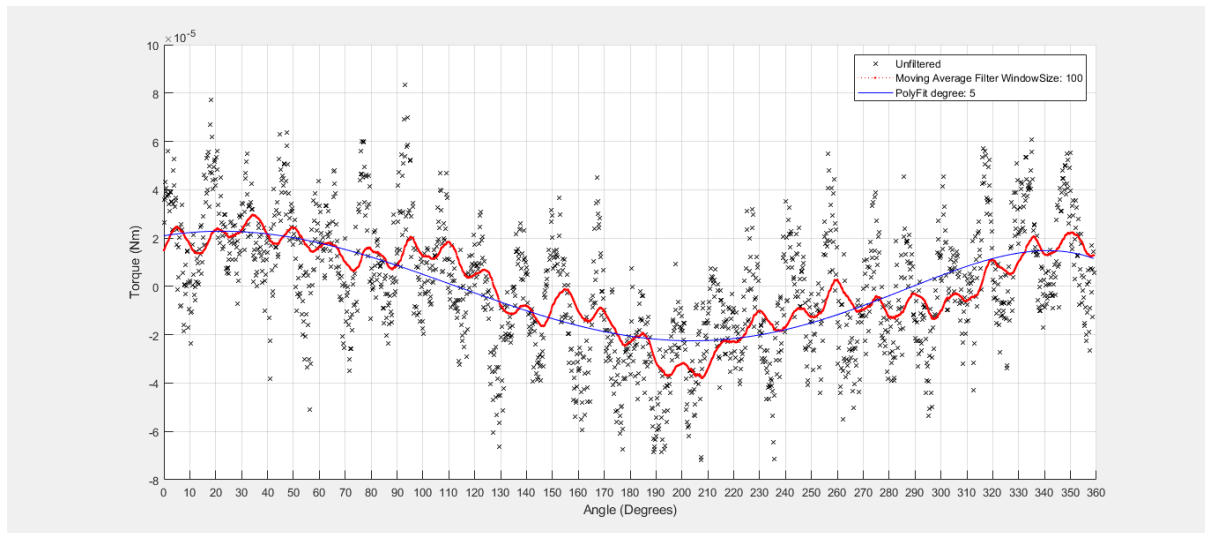


FIGURE 12. COMPARISON OF FILTERING METHODS

An additional idea for future development would be by some method summing the angular values of each rotation but adding one value to the previous rotations value so that angle is tallied continuously throughout the dataset. This would result in a dataset of torque measurements alongside time and values of angular displacement so that once the user has saved the dataset can be plotted in its entirety without the requiring the method of splitting rotations detailed earlier in this paper. Furthermore, modification of the data import functions may allow entire folders of datasets to be imported and processed at once to further increase the efficiency of the tool that has been developed.

## 8. Conclusion

The resulting program demonstrated its proficiency in processing TST datasets. A user can now apply a filter to the data, resample both torque and angular encoder signals to a common sample rate, view a plot of torque vs angle of any given rotation in the dataset, and save the processed data to a conveniently named output file for later use. This has made the pre-processing task considerably more repeatable and significantly reduced the time needed to process the data. There is however room for further work and improvements to be made, to summarise the ideas discussed above: the addition of more data filtering options, improvements of to the format of the output files, and the ability to import and process multiple datasets at once would all be useful additions to be made in the future.

## 9. References

- 1-D digital filter - MATLAB filter - MathWorks United Kingdom. [no date]. Available at: <https://uk.mathworks.com/help/matlab/ref/filter.html> [Accessed: 21 April 2021].
- Alkin, O. 2014. *Signals and Systems: A MATLAB® Integrated Approach*. Available at: <https://learning.oreilly.com/library/view/signals-and-systems/9781466598546/cover.xhtml> [Accessed: 11 November 2020].
- Allen, M. et al. 2018. *IPCC: Drafting Authors: Summary for Policymakers SPM*. Available at: [https://www.ipcc.ch/site/assets/uploads/sites/2/2019/05/SR15\\_SPM\\_version\\_report\\_LR.pdf](https://www.ipcc.ch/site/assets/uploads/sites/2/2019/05/SR15_SPM_version_report_LR.pdf) [Accessed: 5 May 2021].
- Allmark, M. 2016. Condition monitoring and fault diagnosis of tidal stream turbines subjected to rotor imbalance faults. Available at: <http://orca.cf.ac.uk/98633/> [Accessed: 11 November 2020].
- Allmark, M. et al. 2020. The development, design and characterisation of a scale model Horizontal Axis Tidal Turbine for dynamic load quantification. *Renewable Energy* 156, pp. 913–930. Available at: <https://doi.org/10.1016/j.renene.2020.04.060>.
- National Instruments 2017. Encoder Measurements: How-To Guide - National Instruments. Available at: <https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000x1riCAA&l=en-US> [Accessed: 1 May 2021].
- TidalPower 2013. History of Tidal Power. Available at: <http://www.tidalelectric.com/history-of-tidal-power> [Accessed: 5 May 2021].

## 10. Appendices

### Appendix 1- Nomenclature

TABLE 1- BASE UNITS

Quantity	Name of base unit	Symbol
length	meter	M
mass	kilogram	Kg
time	second	S
electric current	ampere	A
thermodynamic temperature	kelvin	K

## Project report

amount of substance	mole	mol
luminous intensity	candela	Cd

## Appendix 2- Code and Data

### Datasets


The datasets used can be accessed via Microsoft One Drive upon request of Dr. M. Almark.

```
Run 10 - Notepad
File Edit Format View Help
k(ms);Axis_1 [1] Turbine_4_0(P-0-0043.0.0 Torque-generating current, actual value;Axis_1 [1] Turbine_4_0(S-0-0380.0.0 DC bus voltage;Axis_1 [1] Turbine_4_0(P-0-0067.0.0
0;-0.122;735.4;0.197;-0.171;-27.4;14.8;-67.5344;64.3051;
25;-0.247;735.7;-0.021;-0.293;-7.6;5.6;-67.7454;54.1761;
50;-0.22;734.7;-0.327;0.097;-9.4;-15.2;-67.5809;44.0496;
75;-0.287;737.1;0.207;0.205;-6.9;9.5;-67.42;33.9324;
100;-0.186;735;0.227;-0.229;-19.8;5.2;-67.1732;23.8012;
125;-0.041;735.7;-0.068;0.019;-13.9;8.2;-67.5881;13.6557;
150;-0.076;735;-0.107;0.102;-16.5;-5.7;-67.6203;3.5402;
175;-0.169;737.2;0.241;-0.078;-30.2;3.5;-67.5738;353.4386;
200;-0.144;735.6;0.112;-0.214;-23.5;-4.9;-67.3449;343.301;
225;-0.001;736.1;0.002;-0.102;-29.8;-8.8;-67.6954;333.1705;
250;-0.014;734.7;-0.002;0.019;-22.8;-1.5;-67.5201;323.0368;
275;-0.06;737.2;0.097;-0.034;-16.7;-1.5;-67.5595;312.9117;
300;-0.077;734.6;-0.102;-0.004;-17;1.8;-67.6596;302.7963;
325;-0.085;735.8;-0.141;0.092;-24.9;2.2;-67.2555;292.6738;
350;-0.124;735;0.122;0.053;-26.8;0;-67.4522;282.5422;
375;-0.134;735.6;0.036;-0.185;-17.5;2;-67.6954;272.42;
400;-0.042;734.9;-0.073;0.078;-24.5;-4;-67.4236;262.2939;
425;-0.044;735.8;0.112;-0.107;-29.1;11.8;-67.3771;252.1583;
450;-0.104;735.3;-0.151;0.068;-13.8;7.6;-68.078;242.0245;
475;-0.152;735.8;-0.112;-0.107;-9.5;3.5;-67.1911;231.9078;
500;-0.097;735;-0.083;0.151;-8;-1.6;-67.2591;221.7833;
525;-0.109;736.3;0.061;0.102;-6.8;-8.5;-67.4844;211.6626;
550;-0.137;733.8;0.102;-0.135;-18.3;-4;-67.327;201.5396;
575;-0.046;735.2;0.036;0.048;-17.2;3;-67.4522;191.4055;
600;-0.048;735.7;-0.026;-0.053;-14.5;0.4;-67.7454;181.2808;
625;-0.081;735.8;0.117;-0.107;-17.4;9.3;-67.3485;171.1619;
650;-0.008;733.8;0.046;-0.019;-17.4;-3.6;-67.5058;161.0287;
675;-0.08;735.8;-0.117;0.068;-2.3;-4.8;-67.7633;150.9068;
700;-0.039;736.1;0.083;-0.029;-20.9;4.1;-67.7562;140.7855;
725;-0.046;735.2;-0.046;0.063;-18.8;2.2;-67.5702;130.6517;
750;-0.003;734.4;0.021;-0.043;-4.4;-2.9;-67.4057;120.5159;
775;-0.033;736.2;0.012;0.043;-21.4;4.5;-67.3521;110.4113;
800;-0.125;735.2;-0.146;-0.058;-24.6;10.4;-67.5488;100.2766;
825;-0.072;735.5;0.136;0.024;-25;-19.9;-67.5058;90.1452;
850;-0.006;734.8;0.026;0.083;-28;15.3;-67.1446;80.0332;
```

FIGURE 13. EXAMPLE MOTOR DATA (RUN 10)

01/01/1904 12:00:00.000000 AM	0.011981	0.012328	0.021411	0.002328	0.004290	0.021415	0.021417	0.005716
01/01/1904 12:00:00.000500 AM	0.011976	0.012309	0.021411	0.002326	0.004290	0.021415	0.021417	0.005726
01/01/1904 12:00:00.001000 AM	0.011980	0.012339	0.021411	0.002329	0.004293	0.021415	0.021417	0.005726
01/01/1904 12:00:00.001500 AM	0.011975	0.012326	0.021411	0.002336	0.004296	0.021415	0.021417	0.005728
01/01/1904 12:00:00.002000 AM	0.011986	0.012335	0.021411	0.002332	0.004294	0.021415	0.021417	0.005726
01/01/1904 12:00:00.002500 AM	0.011982	0.012349	0.021411	0.002343	0.004300	0.021415	0.021417	0.005734
01/01/1904 12:00:00.003000 AM	0.011987	0.012341	0.021411	0.002344	0.004301	0.021415	0.021417	0.005734
01/01/1904 12:00:00.003500 AM	0.011978	0.012355	0.021411	0.002339	0.004300	0.021415	0.021417	0.005737
01/01/1904 12:00:00.004000 AM	0.011984	0.012324	0.021411	0.002336	0.004302	0.021415	0.021417	0.005741
01/01/1904 12:00:00.004500 AM	0.011984	0.012337	0.021411	0.002333	0.004310	0.021415	0.021417	0.005740
01/01/1904 12:00:00.005000 AM	0.011974	0.012328	0.021411	0.002332	0.004315	0.021415	0.021417	0.005746
01/01/1904 12:00:00.005500 AM	0.011986	0.012320	0.021411	0.002329	0.004313	0.021415	0.021417	0.005751
01/01/1904 12:00:00.006000 AM	0.011978	0.012351	0.021411	0.002319	0.004314	0.021415	0.021417	0.005747
01/01/1904 12:00:00.006500 AM	0.011977	0.012340	0.021411	0.002329	0.004320	0.021415	0.021417	0.005750
01/01/1904 12:00:00.007000 AM	0.011968	0.012334	0.021411	0.002329	0.004321	0.021415	0.021417	0.005764
01/01/1904 12:00:00.007500 AM	0.011963	0.012342	0.021411	0.002328	0.004325	0.021415	0.021417	0.005761
01/01/1904 12:00:00.008000 AM	0.011971	0.012323	0.021411	0.002327	0.004328	0.021415	0.021417	0.005769
01/01/1904 12:00:00.008500 AM	0.011976	0.012337	0.021411	0.002320	0.004324	0.021415	0.021417	0.005772
01/01/1904 12:00:00.009000 AM	0.011970	0.012344	0.021411	0.002319	0.004330	0.021415	0.021417	0.005774
01/01/1904 12:00:00.009500 AM	0.011966	0.012310	0.021411	0.002311	0.004332	0.021415	0.021417	0.005772
01/01/1904 12:00:00.010000 AM	0.011967	0.012309	0.021411	0.002297	0.004332	0.021415	0.021417	0.005772
01/01/1904 12:00:00.010500 AM	0.011965	0.012318	0.021411	0.002275	0.004334	0.021415	0.021417	0.005783
01/01/1904 12:00:00.011000 AM	0.011941	0.012323	0.021411	0.002262	0.004322	0.021415	0.021417	0.005783
01/01/1904 12:00:00.011500 AM	0.011979	0.012329	0.021411	0.002259	0.004323	0.021415	0.021417	0.005784
01/01/1904 12:00:00.012000 AM	0.011976	0.012330	0.021411	0.002255	0.004316	0.021415	0.021417	0.005789
01/01/1904 12:00:00.012500 AM	0.011985	0.012314	0.021411	0.002244	0.004309	0.021415	0.021417	0.005796
01/01/1904 12:00:00.013000 AM	0.011988	0.012313	0.021411	0.002244	0.004298	0.021415	0.021417	0.005801
01/01/1904 12:00:00.013500 AM	0.012004	0.012309	0.021411	0.002241	0.004295	0.021415	0.021417	0.005798
01/01/1904 12:00:00.014000 AM	0.011974	0.012314	0.021411	0.002246	0.004294	0.021415	0.021417	0.005798
01/01/1904 12:00:00.014500 AM	0.011969	0.012327	0.021411	0.002245	0.004277	0.021415	0.021417	0.005796
01/01/1904 12:00:00.015000 AM	0.011971	0.012312	0.021411	0.002246	0.004252	0.021415	0.021417	0.005796
01/01/1904 12:00:00.015500 AM	0.011973	0.012326	0.021411	0.002246	0.004236	0.021415	0.021417	0.005795
01/01/1904 12:00:00.016000 AM	0.011967	0.012311	0.021411	0.002249	0.004211	0.021415	0.021417	0.005797
01/01/1904 12:00:00.016500 AM	0.011983	0.012328	0.021411	0.002248	0.004195	0.021415	0.021417	0.005799
01/01/1904 12:00:00.017000 AM	0.011980	0.012309	0.021411	0.002248	0.004186	0.021415	0.021417	0.006053
01/01/1904 12:00:00.017500 AM	0.011977	0.012324	0.021411	0.002083	0.004182	0.021415	0.021417	0.006389

FIGURE 14. EXAMPLE ROTOR DATA (RUN 10)

 Processed Data Run 10 - Notepad

File	Edit	Format	View	Help
Torque(N)	Angle(deg)	Time(s)		
0.011974	359.000000	0.191456		
0.011974	358.840657	0.191956		
0.011974	358.681314	0.192455		
0.011974	358.521971	0.192955		
0.011974	358.362628	0.193454		
0.011974	358.203285	0.193954		
0.011974	358.043941	0.194454		
0.011974	357.884598	0.194953		
0.011974	357.725255	0.195453		

FIGURE 15. EXAMPLE OUTPUT FILE PRODUCED BY THE PRE-PROCESSING TOOL (RUN 10)

## Code

```
function CMERG_TST_Data_PreProcessing_Tool
clc;
clear;
close all;

set(0,'DefaultFigureWindowStyle','normal');

f = figure('visible','off','unit','pixels','position',[0 0 1100 550]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialise controls %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hMotorPath = uicontrol('Style','edit','String','Input Motor data
FilePath.','Position',[30,500,550,25],'Callback',{@MotorPath_Callback});
hRotorPath = uicontrol('Style','edit','String','Input Rotor data
FilePath.','Position',[30,470,550,25],'Callback',{@RotorPath_Callback});
hbrowseMotorPath =
uicontrol('Style','pushbutton','String','browse','Position',[600,500,100,25],
'Callback',{@browseMotorPath_Callback});
hbrowseRotorPath =
uicontrol('Style','pushbutton','String','browse','Position',[600,470,100,25],
'Callback',{@browseRotorPath_Callback});
hLoadMotorData =
uicontrol('Style','togglebutton','String','Load','Position',[720,500,70,25],
'Callback',{@LoadMotorData_Callback});
hLoadRotorData =
uicontrol('Style','togglebutton','String','Load','Position',[720,470,70,25],
'Callback',{@LoadRotorData_Callback});

hPicRot =
uicontrol('Style','edit','String','1','Position',[1040,500,30,25],'Callback',
{@PicRot_Callback});
hLoadRot =
uicontrol('Style','togglebutton','String','Load','Position',[1020,470,70,25],
'Callback',{@LoadRot_Callback});
```

## Project report

```
hRotLabel = uicontrol('Style','text','String','Which Rotation would you
like to plot?','position',[800,495,190,25]);

hNextRot = uicontrol('Style','pushbutton','String','>> Next
Roation','Position',[900,470,100,25],'Callback',{@NextRot_Callback});
hPrevRot = uicontrol('Style','pushbutton','String','Prev. Roation
<<','Position',[800,470,100,25],'Callback',{@PrevRot_Callback});
hSave = uicontrol('Style','pushbutton','String','Save
Results','Position',[980,430,100,25],'Callback',{@Save_Callback});

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialise Plot Objects %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ha_angle = axes('Units','pixels','position',[70,280,500,160]);
ha_angle.YLim=[0 360];
ha_angle.YTick=[0 30 60 90 120 150 180 210 240 270 300 330 360];
ha_angle.XLabel.String='time(s)';ha_angle.YLabel.String='Angle
(Deg)';ha_angle.Title.String='Angle vs Time';%lables axes

ha_torque = axes('Units','pixels','position',[70,50,500,160]);
ha_torque.XLabel.String='time(s)';ha_torque.YLabel.String='Torque
(Nm)';ha_torque.Title.String='Torque vs Time';%lables axes

ha_angle_torque = axes('Units','pixels','position',[660,50,360,360]);
ha_angle_torque.XLim=[0 360];
ha_angle_torque.XTick=[0 30 60 90 120 150 180 210 240 270 300 330 360];
ha_angle_torque.XLabel.String='Angle
(Deg)';ha_angle_torque.YLabel.String='Torque
(Nm)';ha_angle_torque.Title.String='Torque vs Angle';%lables axes

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Normalise objects %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hMotorPath.Units = 'normalized';
hRotorPath.Units = 'normalized';
hbrowseMotorPath.Units = 'normalized';
hbrowseRotorPath.Units = 'normalized';
hLoadMotorData.Units = 'normalized';
hLoadRotorData.Units = 'normalized';
hPicRot.Units = 'normalized';
hLoadRot.Units = 'normalized';
ha_angle.Units = 'normalized';
ha_torque.Units = 'normalized';
ha_angle_torque.Units = 'normalized';
hRotLabel.Units = 'normalized';
hNextRot.Units= 'normalized';
hPrevRot.Units= 'normalized';
hSave.Units= 'normalized';

f.Name = 'CMERG TST Data Pre-Processing Tool';% Assigns a name to appear in
the window title.
f.NumberTitle = 'off'; %takes the figure number out of the figure title
movegui(f,'center'); % Moves the window to the center of the screen.
f.Visible = 'on'; % Make the UI visible

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialise variables %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
motorFilename = '';
rotorFilename = '';
motorFilePath = '';
rotorFilePath = '';
angleTime =zeros(1,10000);
theta=zeros(1,10000);
Torque=zeros(1,1000000);
```

## Project report

```
Tfilter=zeros(1,1000000);
torqueTime=zeros(1,1000000);
rotNum=[1];
spl=zeros(1,1000);
sph=zeros(1,1000);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Callback functions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function MotorPath_Callback(~,~)
motorFilePath = hMotorPath.String;
end

function RotorPath_Callback(~,~)
motorFilePath = hRotorPath.String;
end

function browseMotorPath_Callback(~,~)
[file,path] = uigetfile({'*.txt'}, 'Select Motor Data File');
motorFilePath = [path,file];
hMotorPath.String = [path,file];
motorFilename = file;
end

function browseRotorPath_Callback(~,~)
[file,path] = uigetfile({'*.txt'}, 'Select Rotor Data File');
rotorFilePath = [path,file];
hRotorPath.String = [path,file];
rotorFilename = file;
end

function LoadMotorData_Callback(~,~)

[angleTime, theta] = importAngleData(motorFilePath);
localMins = islocalmin(theta, 'SamplePoints', angleTime);
localMax = islocalmax(theta, 'SamplePoints', angleTime);
plot(ha_angle, angleTime, theta, '-
x', angleTime(localMins), theta(localMins), 'r*', angleTime(localMax), theta(localMax), 'r*'); %plots angle vs time, labels local minimums
ha_angle.XGrid, 'on';
ha_angle.YGrid, 'on';
ha_angle.YLim=[0 360];
ha_angle.YTick=[0 30 60 90 120 150 180 210 240 270 300 330 360];

ha_angle.XLabel.String='time(s)'; ha_angle.YLabel.String='Angle'; ha_angle.Title.String='Angle vs Time'; %labels axes

spl=find(localMins==1);
sph=find(localMax==1);
end

function LoadRotorData_Callback(~,~)
[torqueTime, Torque] = importTorqueData(rotorFilePath);

windowSize = 100;
b = (1/windowSize)*ones(1,windowSize);
a = 1;
Tfilter= filter(b,a,Torque);

plot(ha_torque, torqueTime, Torque, torqueTime, Tfilter);

ha_torque.YLim=[(min(Torque)) (max(Torque))];
```

## Project report

```
    ha_torque.XGrid, 'on';
    ha_torque.YGrid, 'on';
    ha_torque.XLabel.String='time(s)';%lables axes
    ha_torque.YLabel.String='Torque (Nm)';%lables axes
    ha_torque.Title.String='Torque vs Time';
end

function PicRot_Callback(~,~)
    rotNum=str2double(hPicRot.String);
end

function NextRot_Callback(~,~)
    plotRotation(rotNum+1);
    rotNum=(rotNum+1);
    hPicRot.String=rotNum;
end

function PrevRot_Callback(~,~)
    plotRotation(rotNum-1);
    rotNum=(rotNum-1);
    hPicRot.String=rotNum;
end

function LoadRot_Callback(~,~)
    plotRotation(rotNum);
end

function Save_Callback(~,~)
    Print_processed_results;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Functions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [angleTime, theta] = importAngleData(~)
    mdata =importdata(motorFilePath, ';');%Imports data into
variable 'mdata'

    ncol = numel(mdata.colheaders);

    if (ncol == 9)
        %Loading Angle encoder data
        angleTime=(mdata.data(:,1))/1000;
        theta=(mdata.data(:,9));
        %Reads time and angle columns converts to seconds
    elseif(ncol == 8)
        %Loading Angle encoder data
        angleTime=(mdata.data(:,1))/1000;
        theta=(mdata.data(:,7));
        %Reads time and angle columns converts to seconds
    end

end

function [torqueTime, Torque] = importTorqueData(~)
    rdata =importdata(rotorFilePath, '\t');%Imports data into variable
'rdata'

    Torque=rdata.data(:,1).';

    %creates time vector for values of torque in 5ms intervals
```



## Project report

```
    startValue = 0;
    endValue = length(Torque)/2;
    nElements = length(Torque);
    stepSize = (endValue-startValue)/(nElements-1);
    torqueTime = (startValue:stepSize:endValue)/1000;
end

function plotRotation(i)

t=angleTime(sph(i):spl(i+1));
a=theta(sph(i):spl(i+1));

pu=[0,359];
pul=interp1(a,t,pu,'linear','extrap');%time values for start and end of
rotation

    q=find(abs(torqueTime-pul(2)) < 0.00025);
    p=find(abs(torqueTime-pul(1)) < 0.00025);

    rT=Tfilter(q:p);
    length(rT);

    tq=(0:1/((length(rT)-1)/359):359);%Creates stretches angleTime over
every torque value
    tq1=interp1(a,t,tq,'linear','extrap');%time values for every torque
value

    tq2=interp1(torqueTime,Tfilter,tq1,'linear','extrap');

    plot(ha_angle_torque,tq,flip(tq2),'b:.');

    tu=(0:1:359);
    tu1=interp1(a,t,tu,'linear','extrap');%time values for angles 0-359

    tu2=interp1(torqueTime,Tfilter,tu1,'linear','extrap');
    hold on
    plot(ha_angle_torque,tu,flip(tu2),'rx');
    hold off
    grid; xlim([0 360]);
    xticks([0 30 60 90 120 150 180 210 240 270 300 330 360]);
    xlabel('Angle (Deg)');ylabel('Torque (Nm)');

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    ha_angle_torque.XGrid,'on';
    ha_angle_torque.YGrid,'on';
    ha_angle_torque.XLim=[0 360];
    ha_angle_torque.XTick=[0 30 60 90 120 150 180 210 240 270 300 330
360];
    ha_angle_torque.XLabel.String='Angle (Deg)';%lables axes
    ha_angle_torque.YLabel.String='Torque (Nm)';%lables axes
    ha_angle_torque.Title.String='Torque vs Angle';
end

function Print_processed_results(~)

    oldFileName=motorFilename;
    newFileName=append('Processed Data ',oldFileName);
```

## Project report

```
fileID=fopen(newFileName,'w');
fprintf(fileID,'%10s %10s
%10s\n','Torque (Nm) ','Angle (deg) ','Time (s) ');

for i=1:length(spl)-1

t=angleTime(sph(i):spl(i+1));
a=theta(sph(i):spl(i+1));

pu=[0,359];
pul=interp1(a,t,pu,'linear','extrap');%time values for start and end of
rotation

q=find(abs(torqueTime-pul(2)) < 0.00025);
p=find(abs(torqueTime-pul(1)) < 0.00025);

rT=Tfilter(q:p);
length(rT);

tq=(0:1/((length(rT)-1)/359):359);%Creates stretches angleTime over
every torque value
tq1=interp1(a,t,tq,'linear','extrap');%time values for every torque
value

tq2=interp1(torqueTime,Tfilter,tq1,'linear','extrap');

A = [flip(tq2); flip(tq); flip(tq1)];
fprintf(fileID,'%10.6f %10.6f %10.6f\n',A);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

fclose(fileID);

end
end
```

## Project report

### Record of project meetings

CARDIFF SCHOOL OF ENGINEERING - Mechanical

**NAME:** Aiden Devine

**STUDENT NUMBER:** 1618454

**SUPERVISOR:** Prof. Roger Grosvenor

**TEACHING DISCIPLINE:** Mechanical Engineering

<b>Date of meeting On-line or F2F?</b>	<b>Student's summary of progress since last meeting – to be completed before the next meeting.</b>	<b>Supervisor's assessment of progress</b>	<b>Actions by next meeting</b>
15/10/20 Online meeting	First meeting	n/a	Read up on fast Fourier transforms, look at CMERG research website
23/10/20 Online meeting	Built an understanding of FFTs	n/a	-Start writing project brief -Find similar research
29/10/20 Online meeting	Drafted a project proposal for feedback from supervisor	n/a	-Review feedback for brief -Watch lectures From CM module
5/11/20 Online meeting CANCELLED due to internet issues	n/a	n/a	n/a
12/11/20 Online meeting CANCELLED due supervisors schedule	n/a	n/a	n/a
19/11/20 Online meeting	Watched supervisor's lectures, reading on wind turbine CM, reading on MATLAB functions.	n/a	Find more papers
26/11/20 Online meeting		n/a	Review more literature
03/12/20 Online meeting	Preparing presentation	Making good progress	Prepare presentation
10/12/20 Online meeting	Completed presentation, summarised current work.	n/a	Provided datasets for testing in MATLAB

Christmas Break and Autumn Exam Period			
05/02/2021	Reviewed presentation feedback	n/a	Experiment with data
11/02/21	Imported data into MATLAB	n/a	Separate data
18/02/21 Cancelled	n/a	n/a	n/a
25/02/21	Used islocalmins to separate rotations.	Good use of local min/max MATLAB function	Attempt to split data into rotations
4/03/21	Split rotations and Attempted to use resample to some success	n/a	-investigate resample.
11/03/21	more attempts with resample	n/a	Continue investigating resample
18/03/21	-resample function ruled out	Good attempt but investigate other methods of resampling	-investigate alternatives to resample.
25/03/21	-Gave feedback on code ideas and discussed new ideas to work on over break	Strong progress	-investigate Interpolation -Develop User interface
Easter Break (3 weeks)			
22/04/21	- Successfully used Interpolation to resample angle data -strong progress with report	Good work	-finish report
29/04/21	n/a	n/a	n/a
6/05/21	-Proof reading and feedback	Well written report, add references to introductions, figure numbers and spell check	-Complete report