SLIDER CRANK WAVE ENERGY CONVERTER PERFORMANCE ANALYSIS WITH ADAPTIVE AUTOREGRESSING FILTERING

A thesis presented to the faculty of the Graduate School of
Western Carolina University in partial fulfillment of the
requirements for the degree of Master of Science in Technology

By

Md Rakib Hasan Khan

Director: Dr. Bora Karayaka
Associate Professor
School of Engineering and Technology

Committee Members: Dr. Yanjun Yan, School of Engineering and Technology
Dr. Peter Tay, School of Engineering and Technology
Western Carolina University

April 2019

ACKNOWLEDGEMENTS

The contemporary, but vast course structure along with world-class research facilities and renowned faculty members at WCU not only increased my enthusiasm towards research but also enlightened my knowledge of my interest. It's my immense pleasure that at WCU I am working with Dr. H. Bora Karayaka, whose invaluable experience and guidance have helped me to evaluate the research problems comprehensively. I would like to thank my thesis advisor and committee members Dr. Yanjun Yan and Dr. Peter Tay for their help and valuable guidelines in my thesis and conference papers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

SLIDER CRANK WAVE ENERGY CONVERTER PERFORMANCE ANALYSIS WITH ADAPTIVE AUTOREGRESSING FILTERING

Md Rakib Hasan Khan, M.S.T.

Western Carolina University (April 2019)

Director: Dr. Bora Karayaka

This study investigates a performance analysis of wave excitation force prediction to extract wave power for a slider crank power take-off system (PTOS) based on auto regressive (AR) filters. To efficiently convert wave energy into electricity, the prediction of wave excitation forces into near future to keep the generator and the wave excitation force in sync is important for maximum energy extraction. The study shows a prediction methodology of half period and zero crossings in the practical scenario of irregular ocean waves. The prediction has been tested for different wave periods and with different filter orders in noisy and noiseless environment. The prediction results have been used in the PTOS simulation to analyze the energy extraction. It has been shown that the prediction accuracy in the wave half period between the truth data and the predicted data drives the WEC energy extraction efficiency. The amplitude of the wave force is not used and hence the prediction deviation in the wave force amplitude does not affect the PTOS energy extraction. Further analysis shows that the optimum energy can be extracted at 15^{th} order filter with moderate prediction horizon length.

CHAPTER 1: INTRODUCTION

1.1 Key Terms

Autoregressive Filter (AR): An autoregressive filter is a model where the current value of a variable depends upon only the values that this variable took in previous periods plus an error term.

Wave Energy Converter (WEC): A wave energy converter is a device that converts the kinetic and potential energy associated with a moving wave into useful mechanical or electrical energy.

Zero Crossing: When the wave excitation force changes its transition either positive to negative or negative to positives, we called the transition point as a zero crossing.

Half Cycle Duration: The difference between two zero crossings is called the half cycle duration.

1.2 Problem Statement

At present, our electricity production is highly dependent on conventional energy sources includes oil, gas and coal. These conventional sources are usually fossil fuels. Fossil fuels are non-renewable, that is, they draw on finite resources that will eventually dwindle, becoming too expensive or too environmentally damaging to retrieve. In contrast, the many types of renewable energy resources-such as wave, wind and solar energy are constantly replenished and will never run out [1]. Therefore, renewable energy is now a matter of interest to produce electricity.

Ocean wave energy is an emerging field in renewable energy research. In the ocean, energy exists in various form and waves are one of the largest marine resources as well as the most widely accessible [1]. Compared with other renewable energy sources like wind energy, ocean wave energy has a higher power density. Wave energy contains roughly 1000 times the kinetic energy of wind, allowing much smaller and less conspicuous devices to produce the same

amount of power in a fraction of the space. Unlike wind and solar power, power from ocean waves continues to be produced around the clock, whereas wind velocity tends to die in the morning and at night, and solar is only available during the day in areas with relatively little cloud cover. So, Wave power has been considered as one of the most promising renewable energy sources. Wave energy converter (WEC) is a device, which captures the power of waves and transforms it to electricity. The slider crank WEC converts the heave motion of ocean waves into rotational motion. In order to run the slider crank PTOS at relatively high efficiency, the generator needs to be synchronized with the wave excitation forces.

Nowadays real-time control has become popular to maximize the energy extraction of WEC. In order to implement the real-time control, a prediction of wave excitation force is needed. In this control strategy, the control law has to be determined at every time step based on the prediction of wave excitation force in the near future. A semi-submerged spherical buoy is assumed for this research. The prediction algorithms are carried out in the MATLAB^{TM} environment. Then the offline prediction results are used to the simulation model in the SIMULINK^{TM} environment to analyze the energy extraction with different wave periods and filter orders.

The outline of this thesis is organized as follows: Chapter 2 describes the literature review, research on related topics is analyzed to choose the best technique for wave excitation force prediction. Chapter 3 describes the methodology and control methods are used to develop the prediction algorithm. Chapter 4 verifies the effectiveness of the methodology and control methods through simulations. Finally, conclusions and future works are described in Chapter 5.

CHAPTER 2: LITERATURE REVIEW

2.1 Current Techniques

At present, a number of different wave energy converter concepts are being investigated by companies and academic research groups around the world. Among the challenges faced, wave resource prediction is an important barrier to the use of ocean wave energy arising from its highly variable nature. Although many working designs have been developed and tested through modelling and wave tank-tests, only a few concepts have progressed to sea testing. Research has been done on DDR-PTOS such as rack-and-pinion mechanisms and traction tires/wheels [3], but research on the slider-crank PTOS is rare.

Various models have been developed for wave prediction, however they are hardly applied for WEC real-time control. In [4], the fast Fourier transformation has been utilized to predict the random sea waves, whereas in [5], the wave prediction model was developed based on the grey model. But both of this model depends on a large number of historical data and this slow down the prediction process, which is a big concern of real time application. In [6], the first order-one variable grey model GM (1,1) is used to predict the wave forces over the receding horizon. In [7], the autoregressive moving average model is used to predict the wave elevations. In [8], the pseudo-spectral control method is used to optimize the power capture of an oscillating surge WEC.

Fusco et al. showed for low frequency wave prediction, the autoregressive (AR) model is a relatively simple and accurate method [9]. In this study, wave elevation was predicted to give satisfactory results. The initial analysis on the slider-crank PTOS under regular wave condition validated the suboptimal nature of the control strategy [10]. However, because ideal sinusoidal

wave conditions rarely exist in real oceans, it is very important to find a prediction methodology for the system under irregular wave conditions.

Further investigations were also conducted with the slider-crank PTOS under irregular wave conditions, but with the assumption of known future half period information [11] - [12], which is infeasible in practical applications.

2.2 Chosen Technique

The proposed system uses Slider Crank WEC because of its simplicity. Control methodology with predicted wave data has been chosen for performance evaluation purposes. Slider crank WEC has a fixed amplitude motion, which eliminates the need for additional latching control technique to limit buoy motion under extreme wave conditions. In addition, slider crank is a well-accepted mechanical linkage system with a history of more than 2,000 years. This study provides a lean prediction strategy because prediction and control execution runs simultaneously to the slider crank power take-off system for the continuous energy production.

The control methodology with prediction is designed to keep the generator rotating in resonance with the wave excitation force so that energy can be extracted at a relatively high efficiency. In this study, an autoregressive model with a Forward-Backward parameter estimation approach [13] is used for the zero crossings' prediction purpose. The future half period is not constant, it changes its length based on the finding of two zero crossings in each prediction horizon, which makes it suitable for real time control.

CHAPTER 3: METHODOLOGY

3.1 Overall System Model

Driven by the wave excitation force, the buoy's linear motion pushes the generator to start to rotate. The generator that is coupled with the power take-off system needs to maintain continuous rotation in order to generate electricity. During one complete full cycle of rotation, there are instances when the torque is zero, at those moments the generator will pull power from the grid to continue the rotation to complete the cycle. To resonate with the wave, the generator needs to know, when its torque crosses the zero line, the next half-cycle's duration to maintain the resonance [14]. In this study, prediction of future half cycle duration by detecting zero crossings has been analyzed with different AR filter orders. The excitation force is calculated from irregular waves generated through the JONSWAP spectrum [15], and simulations are carried out in the MATLAB/Simulink environment. A sample off – shore implementation of the proposed slider crank WEC system is shown in Fig. 3.1.

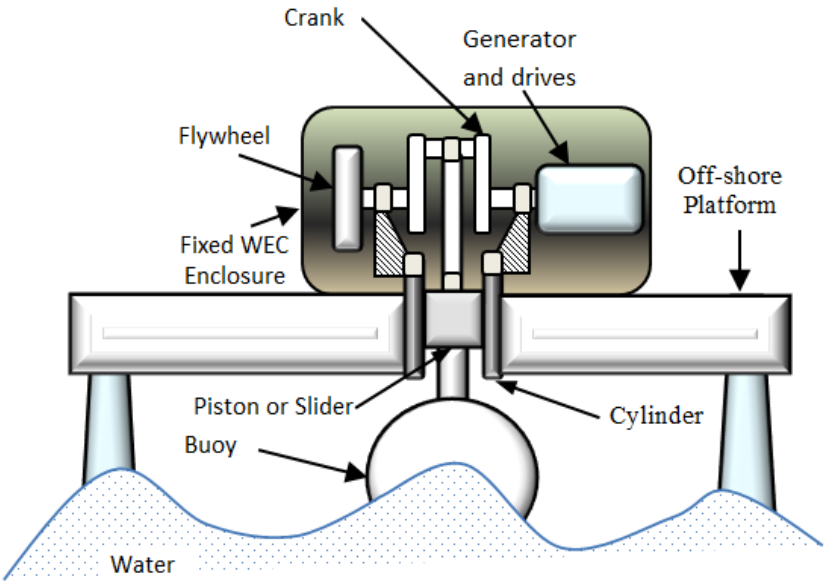

Figure 3.1: Proposed Slider Crank WEC

The basic parts of the slider crank WEC system shown in Fig. 3.1 include a piston or slider, a connecting rod, a crank and a buoy. The piston or slider is firmly affixed to a buoyant wave energy capture device which tracks the relative heave motion of ocean waves. The force that is exerted on the buoy pushes the connecting rod to turn the crank up (or down; depending on the PTOS orientation) and develops the necessary torque that drives the generator to start turning and continue the rotational motion. The control algorithm detects the half period and zero-crossings of wave excitation force and records real time, and then an angle reference is generated. In the meantime, the shaft angle of the generator is detected and compared with the reference. Then the angle control algorithm, which is a simplified version of a PID controller, calculates a speed reference for the motor drive system according to the difference between the shaft angle and its reference. The reference speed makes sure that buoy's velocity is in phase with the excitation force [12].

3.2 Hydrodynamics Model

The Cummins equation [16] is utilized to model buoy and wave interactions. The buoy selected is a sphere and assumed to be half submerged regardless of wave conditions. Equation (1) models the relationship between the buoy motion and hydrodynamic forces.

$$(M + a_{\infty})\ddot{z}(t) + \int_{-\infty}^{t} H_{rad}(t-\tau)\,\dot{z}(\tau)d\tau + S_b z(t) = F_e(t) - F_u(t) \quad (1)$$

where M is the physical mass of the buoy, a_∞ is the buoy-added mass at an infinite wave period, z is the buoy center of the gravity displacement in heave direction, H_{rad} is the radiation impulse response function, S_b is the hydrostatic stiffness, F_e is the wave excitation force, and F_u is the PTOS' reactionary force.

3.3 Wave Excitation Force Generated for Irregular Waves

An irregular wave can be composed of several regular sinusoidal waves with different amplitudes, angular velocities, and phases. In this research, the angular velocity is chosen in the range of 0.08 Hz to 0.22 Hz with an interval of 1.6 mHz. The significant wave height used is 1.4 meter and the peak period of irregular waves varies from 6 seconds to 10 seconds in this study. The amplitudes of the irregular waves were generated with the JONSWAP spectrum shown in Fig. 3.2, which can be expressed as [15]

$$S(f) = \frac{\alpha_j g^2}{(2\pi)^4} f^{-5} \exp\left[-\frac{5}{4}\left(\frac{f_p}{f}\right)^4\right] \gamma^{\Gamma} \tag{2}$$

where α_j is a nondimensional variable that is a function of the wind speed and fetch length, f_p is the peak frequency of the irregular wave, f is the frequency of the wave components, and γ^{Γ} is the peak enhancement factor. A value of 6 is used for γ in this study, and

$$\Gamma = \exp\left[-\left(\frac{\frac{f}{f_p}-1}{\sqrt{2}\sigma}\right)^2\right], \sigma = \begin{cases} 0.07 & f \le f_p \\ 0.09 & f > f_p \end{cases} \tag{3}$$

$$\alpha_j = \frac{H_{m0}^2}{16 \int_0^{\infty} S^*(f) df} \tag{4}$$

In the above equation, H_{m0} is the significant wave height of the irregular wave, and

$$S^*(f) = \frac{g^2}{(2\pi)^4} f^{-5} \exp\left[-\frac{5}{4}\left(\frac{f_p}{f}\right)^4\right] \gamma^{\Gamma} \tag{5}$$

Significant wave heights in the simulations can be chosen according to the equal energy transport theorem .

$$H_{m0} = 2\sqrt{2}A \tag{6}$$

where A is the amplitude of the regular sinusoidal wave with equal energy.

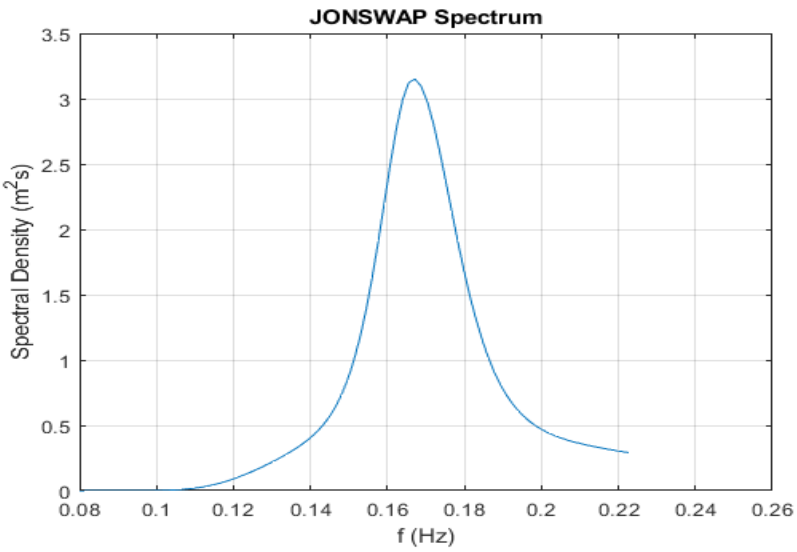

Figure 3.2: The JONSWAP spectrum used for irregular wave

The amplitude of each component of the irregular wave can thus be expressed as in [17].

$$A_i = \sqrt{2S(f_i)\Delta f} \tag{7}$$

where f_i represents each regular wave component and Δf is the separation between two frequency components

The phase of each component of the irregular wave is randomly generated from 0 to 2π, and it is denoted as φ_i . Thus, the irregular wave elevation can be expressed as the summation of all the wave components

$$z_w = \sum_{i=1}^{M} A_i \cdot \sin(\omega_i t + \varphi_i) \tag{8}$$

where M is the total number of wave components, ω_i is the frequency of wave component in radian/second and φ_i is the phase angle for each wave component in radian.

The wave excitation force due to the incident wave is calculated as

$$F_e = |\kappa|\rho g \pi a^2 z_w \angle\varphi_\kappa \tag{9}$$

where z_w is the water surface elevation, κ is the excitation force coefficient g is the acceleration of gravity, ρ is the density of water, and a is the radius of buoy.

The amplitude, imaginary and real parts are calculated as

$$|\kappa| = \sqrt{\frac{4\varepsilon_r}{3\pi ka}} \tag{10}$$

$$Im(\kappa) = \frac{2\varepsilon_r ka}{3} \tag{11}$$

$$Re(\kappa) = \sqrt{|\kappa|^2 - [Im(\kappa)]^2} \tag{12}$$

where κ is wave number and ε_r is radiation resistance coefficient. Wave excitation force data was generated with a precision of 10 ms/sample.

3.4 Autoregressive Filter Model

This study proposes a half-cycle wave excitation force prediction algorithm based on AR model, using the Forward-Backward parameter estimation approach. An autoregressive model is one where the current value of a variable y, depends only upon its previous values and an error term. Equation (13) shows the AR filter expression.

$$\begin{aligned} y_t &= a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_N y_{t-N} + e_t \\ &= \sum_{i=1}^{N} a_i y_{t-i} + e_t \end{aligned} \tag{13}$$

where y_t is the data series, which is the value of the variable y for which the prediction needs to be done at the time period t. It depends upon the previous value of that series i.e. y_{t-1}, $y_{t-2} \ldots y_{t-N}$. Here, N is the order of the filter, or the number of samples used for prediction, and e_t is the noise or disturbance term.

Deriving the linear prediction model involves determining the coefficients a_1, $a_2 \ldots a_N$ in the equation. Several methods and algorithms exist for calculating the coefficients of the AR model, and many are implemented by MATLAB™ command 'ar' [13], which is what this study has used.

3.5 Training Window Model

In the wave excitation force prediction model, the first five complete cycles of irregular wave excitation force initialized the training window. As the prediction continues the training window also shifts by the last ten zero crossings. Equation (14) shows the shifting of the training data.

$$\boldsymbol{y}_{shift} = [y_{1+bn}\ y_{2+bn} \cdots \cdots \cdots y_{n+bn}] \tag{14}$$

where bn is the data index of the first zero crossing point in each predicted half cycle and n is a constant and defines the length of data array with the first ten zero crossings. After getting the first training window, the autoregressive filter coefficients can be found with the help of training data in (13).

After getting the AR filter coefficients, prediction horizon size is set to be about four zero-crossings in (15) to ensure two zero crossings exist in each prediction window since the waves are irregular.

$$T_{step} = 4\times(n \div d) \tag{15}$$

where d is the number of zero crossings in the first training window and set to 10 in this study. As mentioned before, n is the data length.

3.6 Prediction Model

After determining the prediction horizon and AR filter coefficients, prediction starts. Equation (16) shows the prediction of the data series [13].

$$y_{t+1} = -a_1 y_t - a_2 y_{t-1} - \cdots - a_N y_{t-N-1} \tag{16}$$

where y_t is the final data value of each training window, a_1, $a_2 \ldots a_N$ are the AR filter coefficients, N is the filter order. After getting the predicted data, the prediction algorithm detects the zero crossing and half cycle duration.

3.7 Zero Crossing and Half Period Prediction

To maintain the resonance between the slider-crank and the generator, a prediction of future half period of the wave excitation force is needed. The prediction algorithm flow chart is shown in Fig. 3.3. The future half period is identified by two zero crossings detection mechanism. At first, the irregular wave data generated by the irregular wave force calculator is loaded. Then the prediction horizon is chosen, following the procedure presented in Equation (15).

Equation (17) shows the data length that needs to be predicted

$$y_{length} = l - n \tag{17}$$

where l is the length of total actual data and n is the length of initial training data that encompasses ten zero crossings.

The prediction algorithm is initialized by finding the AR filter coefficients from (13). The algorithm is repeated inside a while loop until the end of the sample data series to be predicted (see Fig. 3.3). The AR model estimation is constantly adapted using the data from the last zero-crossing point. The AR filter coefficients are used in (16) to predict data in each prediction horizon by using a for loop and the number of repetitions of that is the prediction horizon length. The zero crossing is determined when the value of the predicted data makes a transition from being positive to negative or negative to positive. If the number of the zero crossings is fewer than two, prediction horizon T_{step} is extended as a safety precaution to cover two zero crossings. However, this rarely happens.

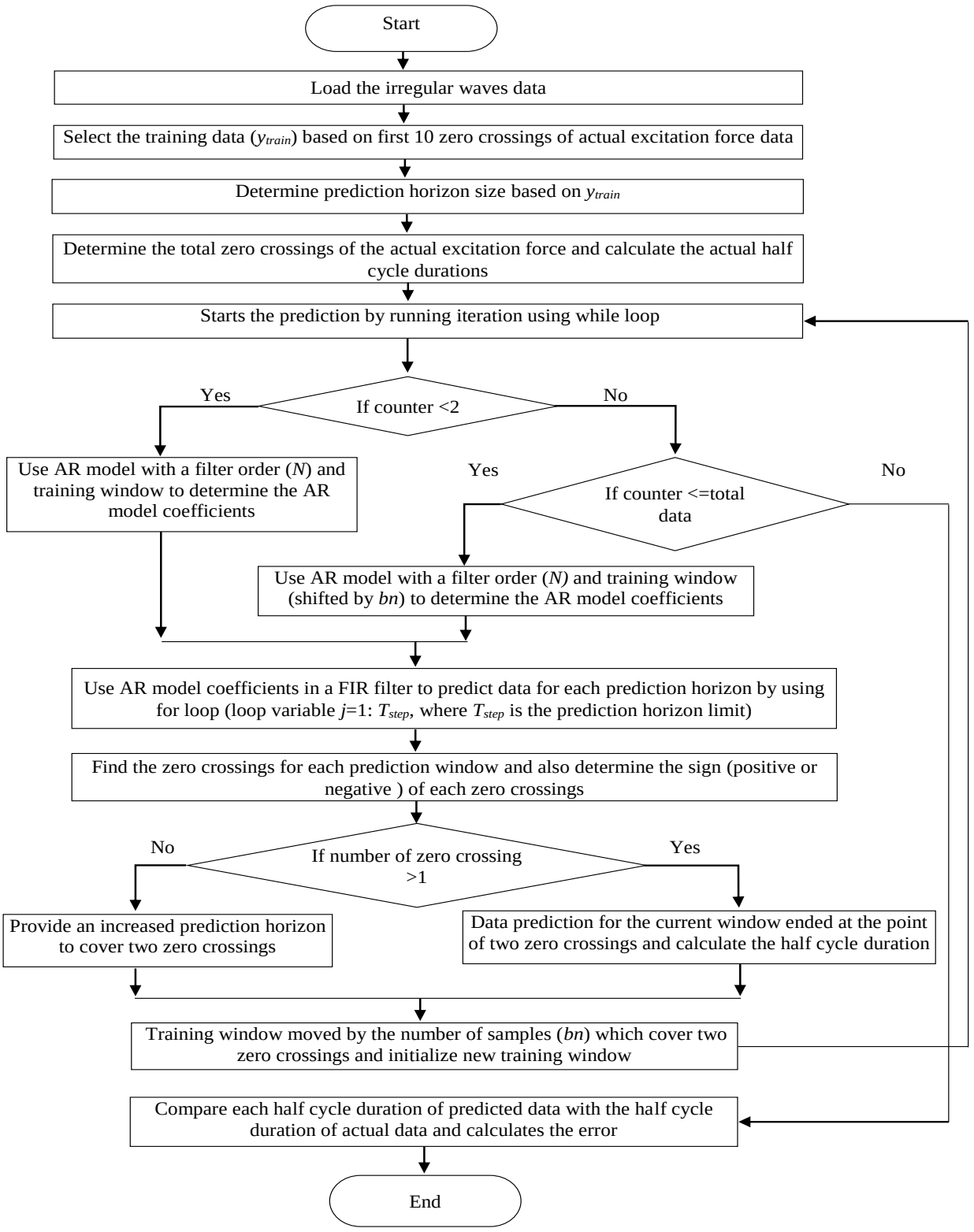

Figure 3.3: Flow chart of wave excitation force prediction algorithm

Data prediction for the current window ends as soon as two zero crossings are determined. After that, new training and prediction cycle start. It should be noted that the algorithm predicts the same zero crossing twice, one with the larger prediction length and the next with the smaller prediction length. For the second prediction of the same zero crossing, the training window will be the same only the starting point of that prediction advances for better accuracy. After that, the half cycle duration has been calculated and recorded. Equation (18) shows the half cycle duration calculation.

$$T_p = t_{zc2} - t_{zc1} \tag{18}$$

where t_{zc1} and t_{zc2} are the time of first and second zero crossing occurrence time of each prediction horizon. This process of zero crossing and half cycle detection is continued until the end of the actual data series. When the prediction is finished, the difference between the square of the predicted half cycles and the actual half cycles duration is calculated and its sum is the metric to evaluate the sum of squares for error (SSE). Equation (19) shows the SSE, T_{error} between the actual and predicted half cycles duration.

$$T_{error} = \sum_{i=1}^{K} (T_{ai} - T_{pi})^2 \tag{19}$$

where T_a is the array, which contains the half cycles durations of the actual data series and T_p is the array, which contains the half cycles durations of the predicted data series. K is the number of observations.

The normalized SSE can be found by dividing the SSE by the number of observations. Equation (20) shows the normalized sum square error.

$$T_K = T_{error} \div K \tag{20}$$

T_K is later used to assess the quality of prediction process for specific filter and time series.

The wave excitation force prediction is used to identify the zero-crossing points to predict the next half-wave period, and hence the amplitude of the wave excitation force prediction is not critical. Although, there are some minor errors in predicted waveform's peaks and valleys, these errors won't impact our analysis since this study focuses only the zero crossings and half cycle durations.

3.8 Test Procedure

First the prediction control algorithm is successfully tested and validated with regular wave forces. Then the control algorithm is tested with irregular wave forces for practical purposes. It should be noted that the irregular wave forces are generated with the help of JONSWAP spectrum. The irregular wave force calculator was already mentioned in section 3.3. After getting the satisfactory prediction results, the next step is to extract the energy. For this reason, simulation has been done to show how much energy can be extracted in comparison to the actual wave data. To compare between the actual and predicted results, the actual wave data needs to be used in the simulation first. After getting power extraction simulation result from the actual data then the prediction data has been used in the simulation model. Finally, the two output results were compared. The wave data has been analyzed with five frequently occurs wave period in ocean, i.e. 6,7,8,9 and 10 seconds wave period. The filter order has been tested for 7, 10 & 15 orders. For each wave period, 5 different wave samples were generated and tested again with 3 different filter orders, in total 75 different cases were analyzed in this study.

It should be noted, in order to compare the predicted data with the actual data, the actual waveform also be truncated to make the same length with the predicted data based on the first 10 zero crossings, as after the first 10 zero crossings prediction starts.

CHAPTER 4: RESULTS

4.1 Prediction Results with Regular Waves

A regular sinusoidal ocean wave is adopted as the excitation source and it has the following form:

$$z_w = A \cdot \sin(\omega t + \varphi) \tag{21}$$

where A is the amplitude of wave, ω is wave angular velocity and φ is the initial angle of the wave.

In this study, five different shift angles have been chosen for the regular waves, i.e. 0°, 72°, 144°, 216°, 288°. Then for each wave periods, 5 samples of data with different shift angles have been formed. It has to be noted that, the regular wave data length is 500 seconds. The wave excitation force due to incident regular wave is calculated as

$$F_e = \kappa \rho g \pi a^2 z_w \tag{22}$$

where κ is the excitation force coefficient, whose amplitude is calculated as

$$|\kappa| = \sqrt{\frac{4\varepsilon_r}{3\pi ka}} \tag{23}$$

The generation of wave excitation force for 6 seconds wave period with the shift angel of 0° is shown in Fig. 4.1. In the figure the blue line shows the true wave excitation forces. The horizontal axis shows the time duration. The total time duration is 500 seconds, where in the figure it shows the duration from 30 seconds to 80 seconds. The vertical axis shows the amplitude of the waves with respect to time.

After generating the excitation force the prediction algorithms discussed in chapter 3 was employed. It generates the predicted waveform shown in Fig 4.2, where the red lines indicate the predicted data for the same time duration 30 seconds to 80 seconds within the total time 500

seconds. The horizontal and vertical axis are the same as discussed in Fig 4.1. This prediction runs for 6 seconds wave periods and filter order 15.

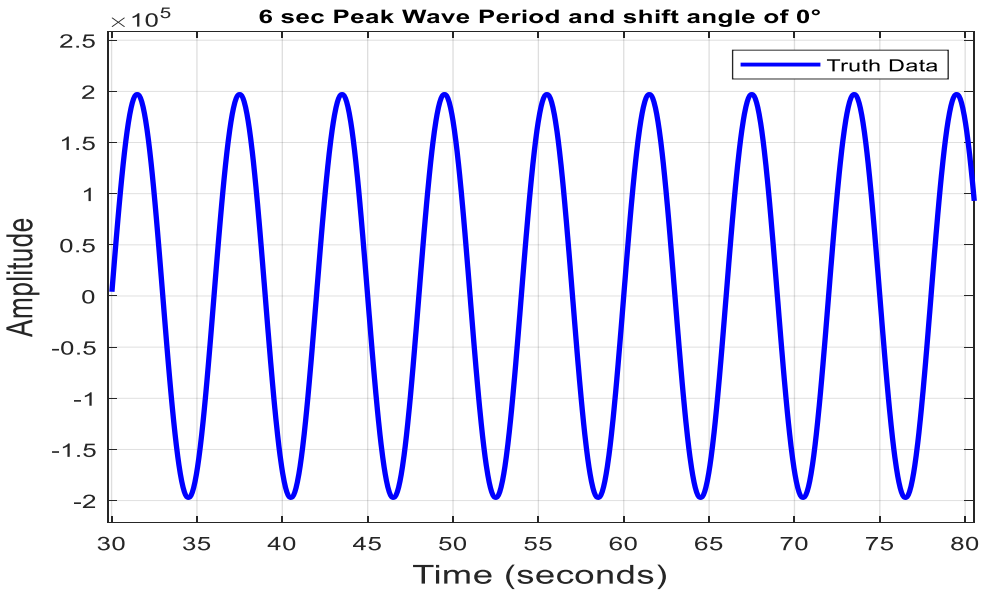

Figure 4.1: Wave excitation force for regular waves

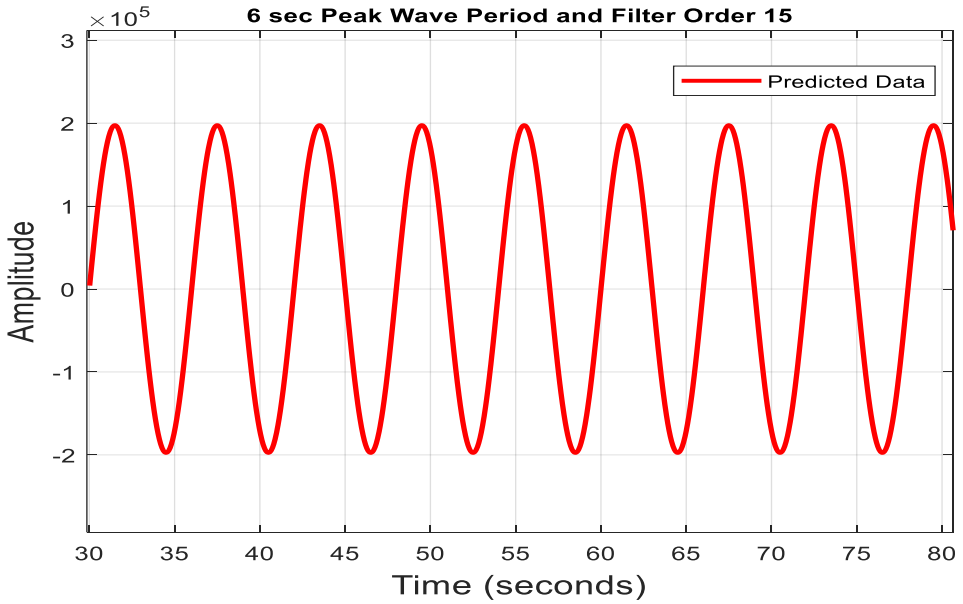

Figure 4.2: Predicted wave excitation force for regular waves

In Fig 4.3, the plot shows the predicted data vs actual data. The blue waveforms show the actual or truth data and the red waveforms shows the predicted data. From the figure, it can be shown that for 6 seconds wave period and filter order 15, the predicted data perfectly overlaps with the actual data. This indicates the accuracy of prediction. In order to distinguish both waveforms, the truth data linewidth is increased, and the predicted data linewidth is decreased as it is shown in legend section of the figure. As discussed in chapter 3, this prediction is based on autoregressive (AR) model, which is a model that can predict the future data based on previous data.

In Fig 4.4, the prediction error histogram is shown, which clearly shows the accuracy of the prediction.

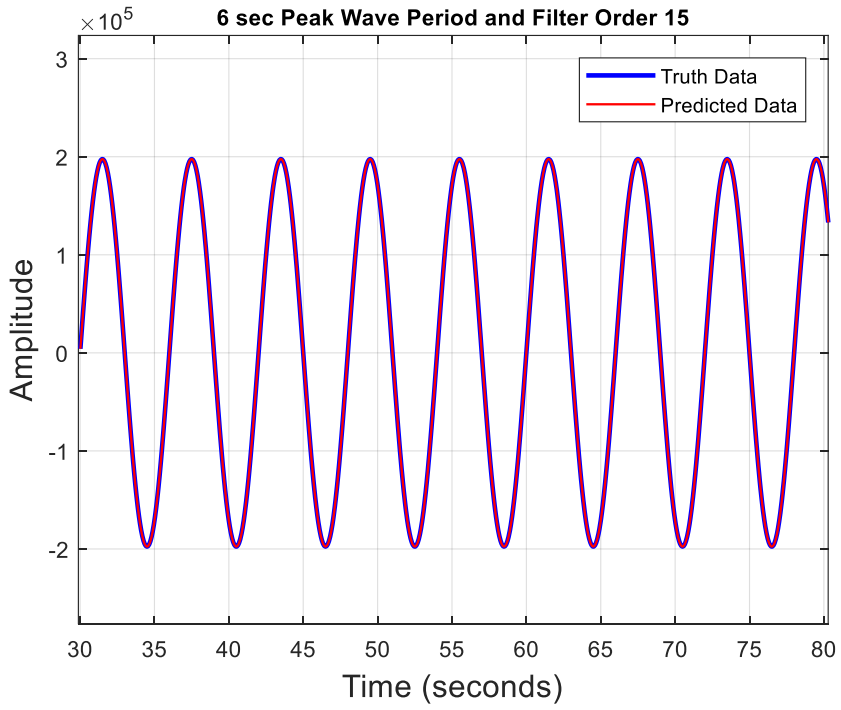

Figure 4.3: Predicted data vs actual data for regular waves

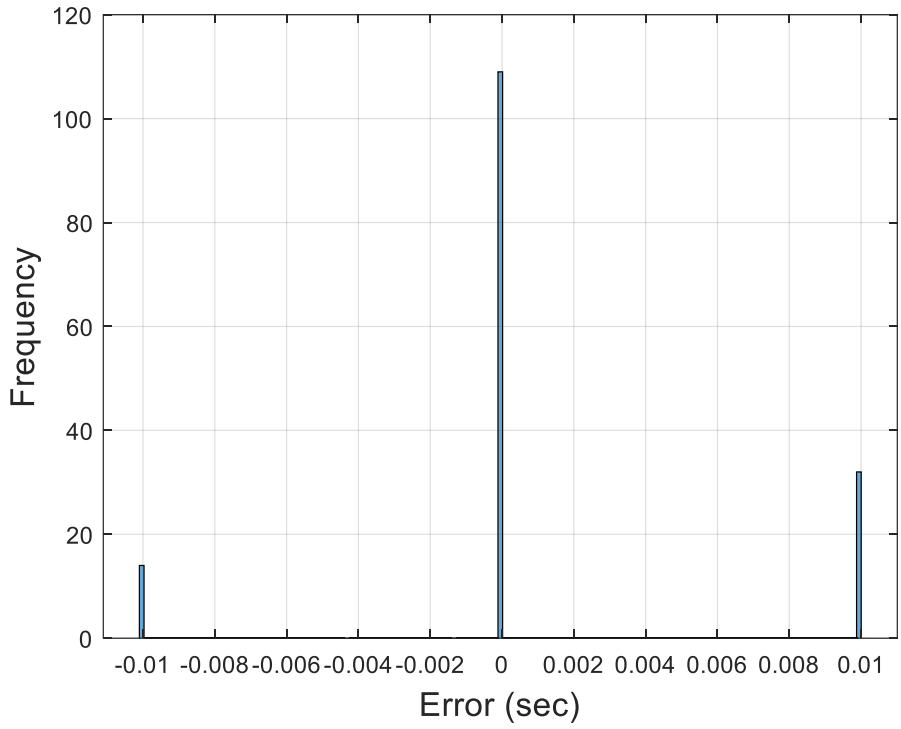

Figure 4.4: Prediction error histogram for regular waves

From the prediction error histogram in Fig 4.4, it has been shown that most of the error occurs within ±10 ms. This indicates the validity of prediction algorithms for regular waves. It should be noted that this prediction histogram error plot is for 6 seconds wave periods and filter order 15.

To validate the system under different wave conditions and study the influence of period on energy extraction, prediction process is carried out with regular waves of five different periods (6, 7, 8, 9, and & 10 sec) for a data length of 500 seconds. For each wave period these cases were studied and three different filter orders, i.e. 7, 10 & 15 were utilized. The prediction results are shown in Table 4.1.

Table 4.1: Half cycle duration prediction results for regular waves

Wave Period (Sec)	Filter Order (N)	Average Normalized SSE (ms) of Half Cycle Duration Prediction	Average Zero Crossings Number	
			Real	Predicted
6	7	0.052	156	156
	10	0.050		156
	15	0.049		156
7	7	0.065	132	132
	10	0.072		132
	15	0.075		132
8	7	0.067	120	120
	10	0.069		120
	15	0.067		120
9	7	0.073	106	106
	10	0.061		106
	15	0.069		106
10	7	0.07	99	99
	10	0.07		99
	15	0.08		99

From the table, it can be observed that, for 6 second wave period, filter order 15 gives the best prediction in terms of half cycle duration error, which is 0.049 ms. However, for 7 second, filter order 7 gives the best prediction with error of 0.065 ms. Again for 8 second wave period, filter order 7 and 15 gives the best accuracy with an error of 0.067 ms, whereas for 9 second, filter order 10 gives the best prediction with 0.061 ms error and for 10 second, filter order 7 and 10 gives the best prediction with 0.07 ms error. However, the average normalized sum square error based on filter order for overall 6,7,8,9 & 10 second together is really close, i.e. 0.07 ms for filter order 7, 0.06 ms for filter order 10 and 15. In addition, with all the filter orders, the number of zero crossings in the actual and predicted data series is identical. Or, the prediction gives almost the same replica of the actual wave curve.

4.2 Power Extraction Simulation Results with Regular Waves

In this simulation, only the control algorithm is modified, the PTOS model and the hydrodynamic model is still the same as in this research's early work [12]. The simulation model is the same for both regular and irregular waves. In simulations of this study, parameters in Table 4.2 and Table 4.3 are adopted. Table 4.2 shows the mechanical parameters used in this simulation and Table 4.3 shows the generator's parameters used in this simulation.

Table 4.2: Mechanical parameters used in simulations

Parameter	Value
Radius of crank (r)	0.5m
Length of connecting rod (l)	1.0m
The distance between the lowest edge of the crank and reference water surface ($d_r - d_{sb}$)	1.0m
Density of water (ρ)	1020kg/m2
Gravitational acceleration (g)	9.81N/kg
Viscous force coefficient (R_v)	10kg/s
Friction force coefficient (R_f)	0
The total mass of slider and connecting rod ($m_{cr} - m_p$)	10kg

Table 4.3: Generator's parameters used in simulations

Parameter	Value
Nominal Speed	1184rpm
Nominal Power	149.2kW
Nominal Voltage	440V
Viscous Friction Coefficient	0.32N/(m/s)
Armature Resistance	0.076Ω
Armature Inductance	0.00157H
Field Resistance	310 Ω
Field Inductance	232.25H
Mutual Inductance	3.320H

In order to get a better picture of simulation results, the simulation was run for 5 different wave periods (i.e. 6,7,8,9 & 10 seconds) and 5 different cases for each wave period. For each simulation case 3 different filter orders (i.e. 7, 10 & 15) were tested. In total 75 simulations have been conducted to validate the prediction algorithm. Fig. 5 highlights various blocks of the proposed WEC system with the control features. In the simulation model, the control algorithm is highlighted in blue, the hydrodynamics part is highlighted in yellow and the PTOS model is highlighted in green.

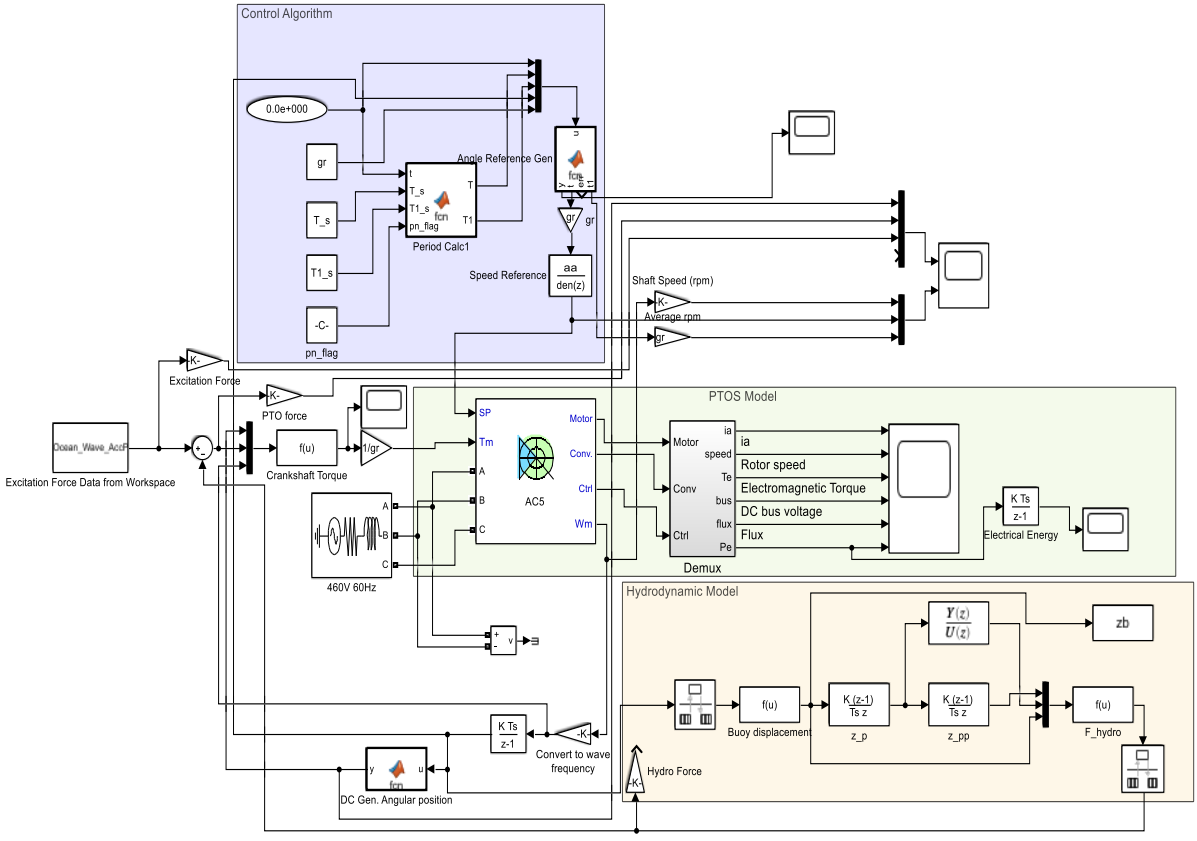

Figure 4.5: Simulation model for the WEC system.

The offline prediction results are provided to the control algorithm block, which includes the half period duration array T_s, the zero crossing time-stamp array T1_s and the type of zero crossing array pn_flag (positive or negative zero crossing). In simulations, the complete duration of each run is 500 seconds. The data from the first ten zero-crossings is used in training, and the rest of the data is used in prediction. The wave cycle duration in waves varies, and hence the actual duration of the first ten zero-crossings, $T_{training}$, may vary between runs, and correspondingly the actual duration of the prediction phase is $T_{prediction} = 500 - T_{training}$ varies. From the simulation data shown in Table 4.4, it can be observed that, for 6 second wave period the output power is increased overall by 0.008 % with the predicted data, whereas for 7 seconds the output power is increased by 0.003% and for 8 seconds its 0.38%. It should be noted that these results only reflect computational analysis of this WEC system.

Table 4.4: Simulation power extraction results for regular waves

Wave Period (Sec)	Filter Order (N)	Average Truth Output Power (kw)	Average Predicted Output Power (kw)	Wave Period wise Average Predicted Power Change (%)	Filter wise Average Predicted Power Change (%)
6	7	34.110	34.112	0.008	0.006
	10		34.114		0.012
	15		34.112		0.006
7	7	38.574	38.574	0.003	0
	10		38.578		0.01
	15		38.574		0
8	7	39.954	39.960	0.38	0.01
	10		40.104		0.38
	15		40.252		0.75
9	7	39.518	39.518	0.14	0
	10		39.522		0.01
	15		39.680		0.41
10	7	38.344	38.348	0.33	0.01
	10		38.510		0.43
	15		38.532		0.49

For 9 second wave period, the output power is increased overall by 0.14 % with the predicted data and for 10 second it increased by 0.33 %. In terms of filter orders of overall examined wave periods, the filter order of 15 gives the best power generation with an increased power extraction of 0.34%, whereas the filter order 10 gives 0.18% increased power extraction and the filter order 7 gives an increased power extraction of 0.01%. Results from the 75 simulation cases of regular waves validate the system can be able to work under a variety of irregular wave conditions and produce satisfactory amounts of energy.

4.3 Prediction Results with Irregular Waves

In this prediction algorithm, adaptive training and prediction run sequentially, so the filter order needs to balance between accuracy and time delay. Filter order determines how many past values need to be used to predict the future value. In this study 7th, 10th and 15th order filters were examined with variety of irregular waves. The prediction process is carried out with irregular waves of five different peak periods (6, 7, 8, 9, and & 10 sec) for a data length of 500 seconds. The significant wave height kept the same at 1.414 m, which is chosen according to the equal energy transport theorem. All the randomly generated waves are derived from the JONSWAP spectrum. In total, there are 75 different trials tested using the prediction and simulation model. The generation of wave excitation force for 8 seconds wave period is shown in Fig. 4.6. In the figure the blue line shows the truth wave excitation forces. The horizontal axis shows the time duration. The total time duration is 500 seconds, the figure shows the duration from 100 seconds to 150 seconds. The vertical axis shows the amplitude of the waves with respect to time. Then the prediction algorithm runs and generates the predicted waveform as shown in Fig. 4.7. In this figure wave period is 8 seconds and filter order is 15.

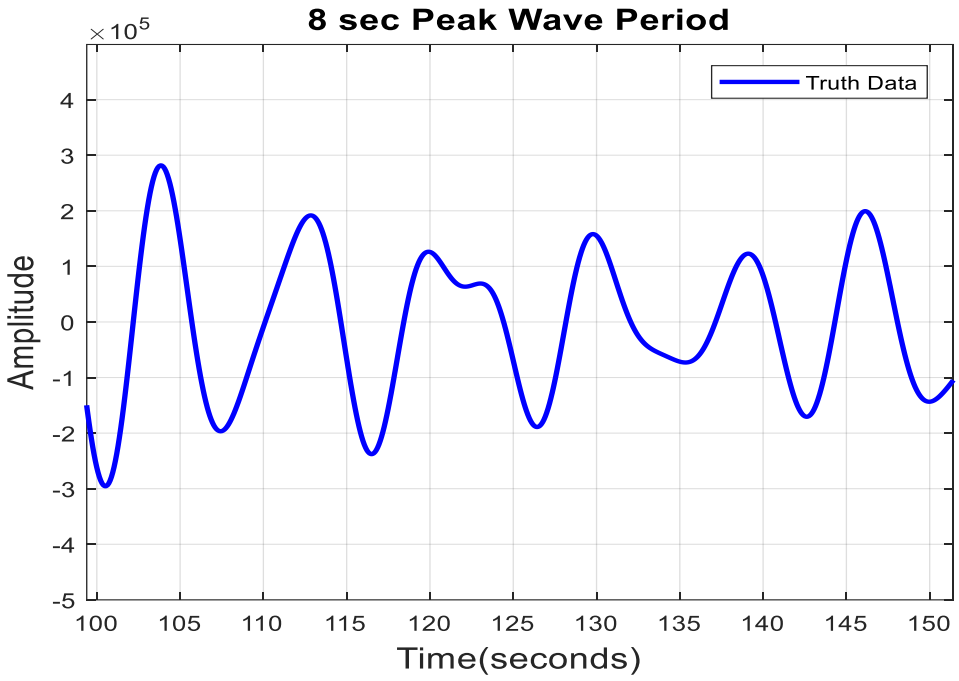

Figure 4.6: Wave excitation force for irregular waves

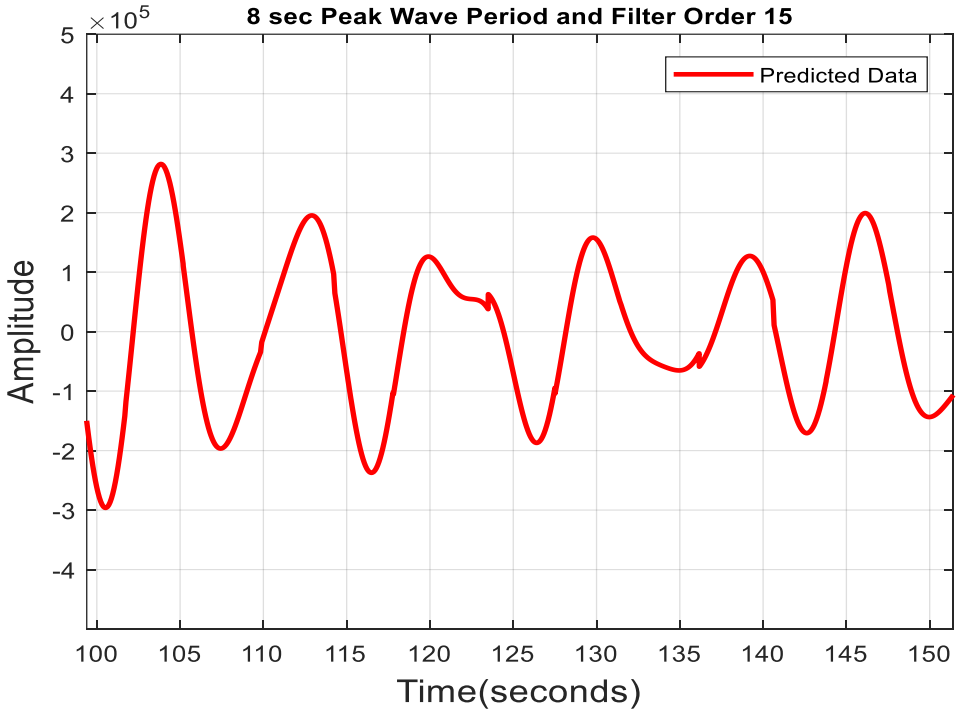

Figure 4.7: Predicted wave excitation force for irregular waves

In Fig 4.8, the plot shows the predicted data vs truth data. The blue waveform shows the actual or truth data and the red waveform shows the predicted data. From the figure, it can be shown that for 8 seconds wave period and filter order 15, the predicted data closely matches the actual data. This indicates the accuracy of prediction. Minor excursions in the peak and valleys of the predicted wave curve comparing with the actual curve can be observed, but as mentioned earlier the amplitude of the future wave is not used in the PTO system. The small spikes in predicted (red) curve represent the reinitializations of a new training/prediction cycles for better accuracy. It is important to note that in every half cycle prediction process, there is a training followed by a prediction. This prediction is later fine-tuned or corrected by making another prediction as real time approaches the first zero crossing in this cycle.

Fig. 4.9 shows the prediction error histogram for 8 second sample wave period with filter order 15. From the prediction error histogram, it has been shown that most of the error occurs within ±200 ms window.

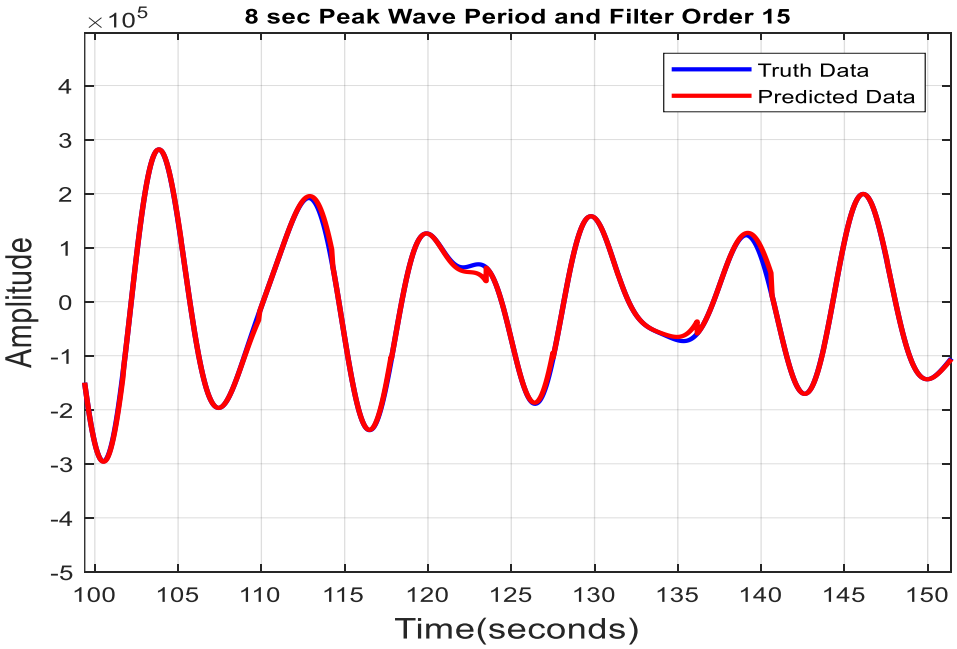

Figure 4.8: Predicted data vs actual data for irregular waves

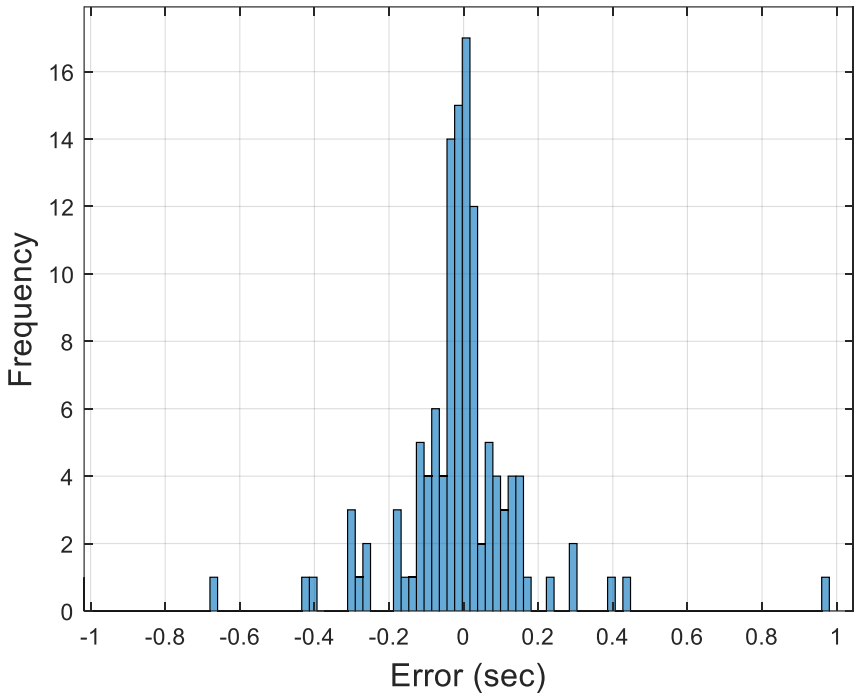

Figure 4.9: Prediction error histogram for irregular waves

In this study, a time offset or cushion of 0.55 seconds is used after each prediction's end to start the new prediction process for the same half period (or two zero crossings). This cushion is a value designed to account for inaccuracies in prediction. By introducing a sufficiently large cushion the same zero crossing are not double counted. However, a half cycle can be smaller than this cushion. If that is the case, the cushion is reduced, and prediction is repeated.

The prediction results are shown in Table 4.5. In this prediction algorithm training and prediction run sequentially, so the algorithm needs to work with a filter order that gives good accuracy and takes less time for the whole process. It's important to note that the error in prediction increases as the peak period increases. This is primarily because the prediction horizon in this study depends on the zero-crossing detection and extends with increasing wave period. As the prediction horizon extends, the sum square error naturally increases.

Table 4.5: Half cycle duration prediction results for irregular waves

Wave Period (Sec)	Filter Order (N)	Average Normalized SSE (Sec) of Half Cycle Duration Prediction	Average Zero Crossings Number	
			Real	Predicted
6	7	0.049	156	156
	10	0.049		156
	15	0.069		156
7	7	0.446	132	132
	10	0.451		132
	15	0.150		132
8	7	0.137	120	120
	10	0.110		120
	15	0.276		120
9	7	0.427	106	106
	10	0.242		106
	15	0.731		106
10	7	1.075	99	97
	10	1.056		97
	15	0.48		99

From the table, it can be observed that, for 6 second wave period, filter order 7 & 10 gives the best prediction in terms of half cycle duration error, which is 0.049 second. However, for 7 second, filter order 15 gives the best prediction with error of 0.150 second. Again for 8 and 9 second wave period, filter order 10 gives the best accuracy with an error of 0.110 & 0.242 second respectively, whereas for 10 second, filter order 15 gives the best prediction with 0.48 second error. However, the average normalized sum square error based on filter order for overall 6, 7, 8, 9 & 10 second together is less in filter order 15, which is 0.341 second, whereas filter order 7 error is 0.427 second and filter order 10 error is 0.382 second. According to the prediction results filter order 15 gives the best prediction. In addition, with a filter order of 15, the number of zero

crossings in the actual and predicted data series is identical. Or, the prediction gives almost the same replica of the actual wave curve.

4.4 Power Extraction Simulation Results with Irregular Waves

Fig. 4.5 highlights various parts of the proposed WEC system with the control features. In the simulation model, the control algorithm is highlighted in blue, in which the half period duration array T_s, the zero-crossing time-stamp array T1_s and the type of zero crossing array pn_flag (positive or negative zero crossing) derived from the prediction algorithm. Simulations are implemented in the following steps. First, as a baseline, the actual data of the irregular wave excitation force is provided to the simulation model. Energy extraction results and system operation waveforms generated from the actual data shows the feasibility of the model. In practice, however, future wave information is never known, and this best-case scenario is not feasible, but just serves as the optimal performance a system can approach. Then the offline wave excitation force prediction data which contains the predicted half cycles, zero crossings and types of zero crossings is tested with the simulation model.

Simulations results are listed in Table 4.6. From the data shown in Table 4.6, it can be observed that, for 6 second wave period the output power is increased overall by 0.99 % with the predicted data. This is possible since perfect phase lock between the excitation force and the generator as in the case of perfect prediction does not necessarily produce best outcome [18].

However, for the other wave periods examined in this study, the output power with predicted data is moderately lower than the power extracted with actual data. For example, with 8 second peak period, the output is reduced by 5.33% with the predicted data. In terms of filter orders of overall examined wave periods, a filter order of 15, gives the best power generation with

a power reduction of 4.29%, whereas filter order 7 gives 5.2% power reduction and filter order 10 gives a power reduction of 4.41%.

Fig. 4.10 shows the cumulative energy generation during the simulation of 9 second peak wave period data. The energy extraction started around 11 seconds after the initialization process of control algorithm, the first wave force affected the system by this time and hence the generator started to produce power. The next wave is then aligned with the slider-crank mechanism and the device extracts more energy than it consumes. This trend continues to produce power at an average rate of 37.6 kw for this specific 9 second wave period.

Table 4.6: Simulation power extraction results for irregular waves

Wave Period (Sec)	Filter Order (N)	Average Truth Output Power (kw)	Average Predicted Output Power (kw)	Wave Period wise Average Predicted Power Change (%)	Filter wise Average Predicted Power Change (%)
6	7	21.93	22.21	0.99	1.29
	10		22.22		1.33
	15		22.01		0.36
7	7	31.84	29.82	-5.35	-6.34
	10		29.98		-5.85
	15		30.61		-3.87
8	7	32.82	30.97	-5.33	-5.64
	10		31.40		-4.32
	15		30.84		-6.02
9	7	36.40	33.59	-6.54	-7.72
	10		33.92		-6.83
	15		34.55		-5.08
10	7	32.98	31.26	-4.9	-5.21
	10		30.83		-6.53
	15		31.26		-5.21

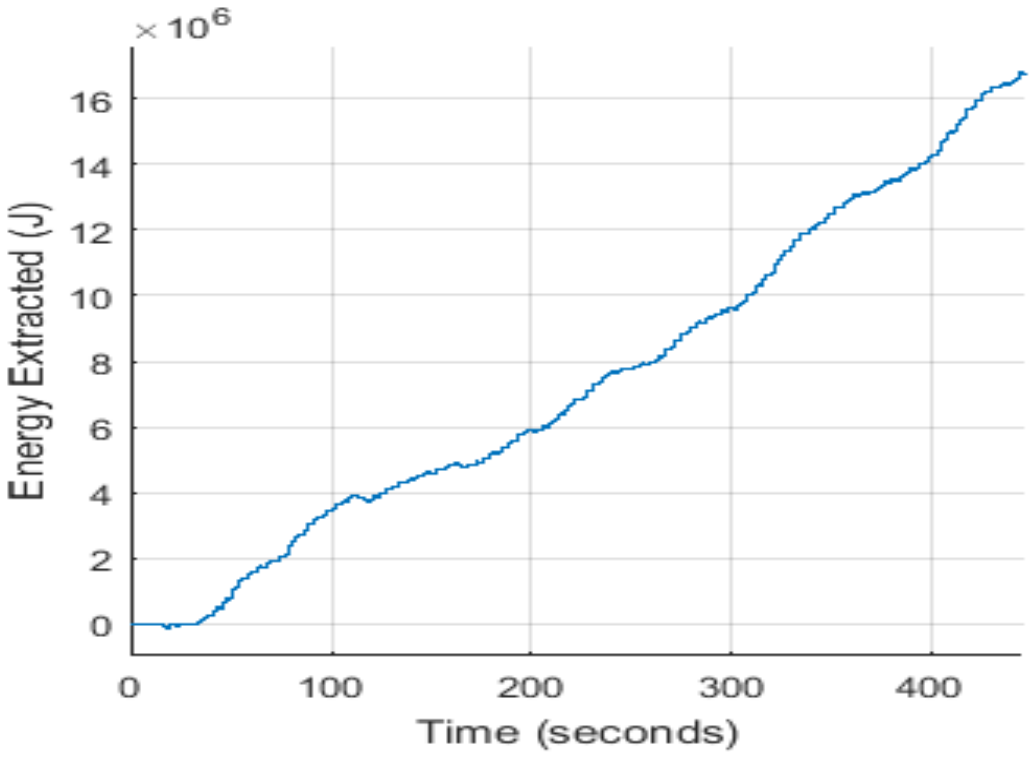

Figure 4.10: Cumulative Electric Energy Production

Results from the 75 simulation cases validate that the system can work under a variety of irregular wave conditions and produce reasonable amounts of energy.

4.5 Prediction & Simulation Result with Noisy Irregular Wave Data

In order to generate the noisy irregular wave data, Signal to Noise Ratio (SNR) has been introduced to the irregular wave force calculator. In this experiment, SNR=0 and SNR=1000:1 has been used to show the noisy environment performance.

In Fig 4.11, the plot shows the predicted data vs actual data for noisy irregular wave. The blue waveform shows the actual or truth data and the red waveform shows the predicted data. From the figure, it can be shown that for 6 second wave period and filter order 15 with

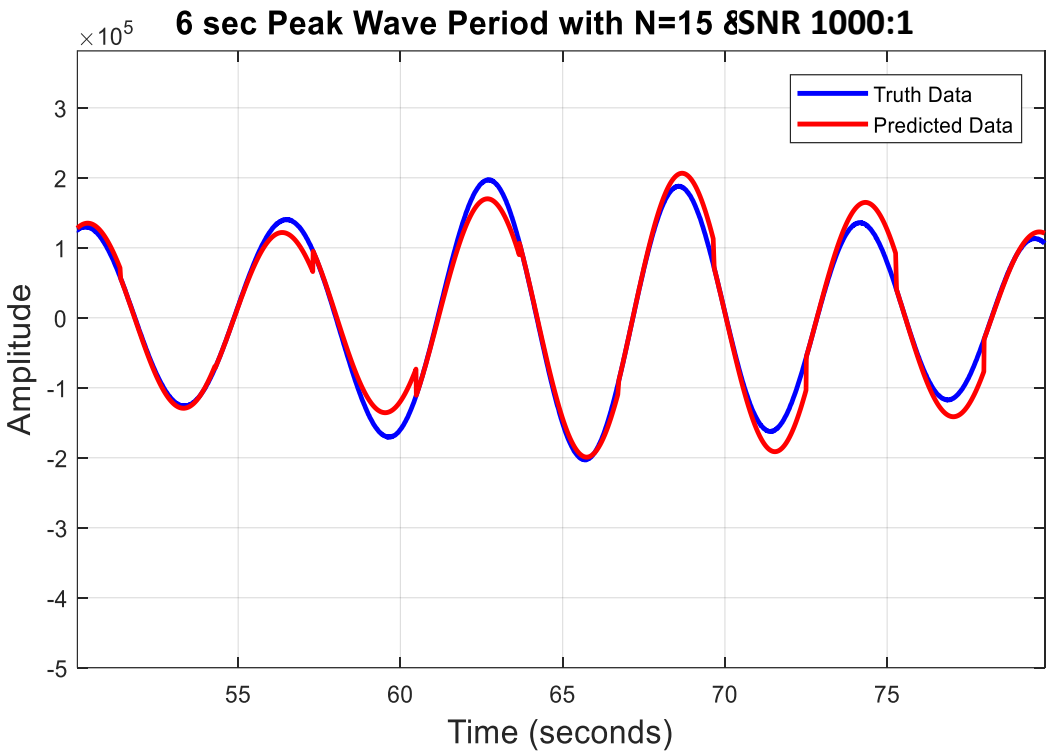

Figure 4.11: Predicted data vs actual data for noisy irregular waves

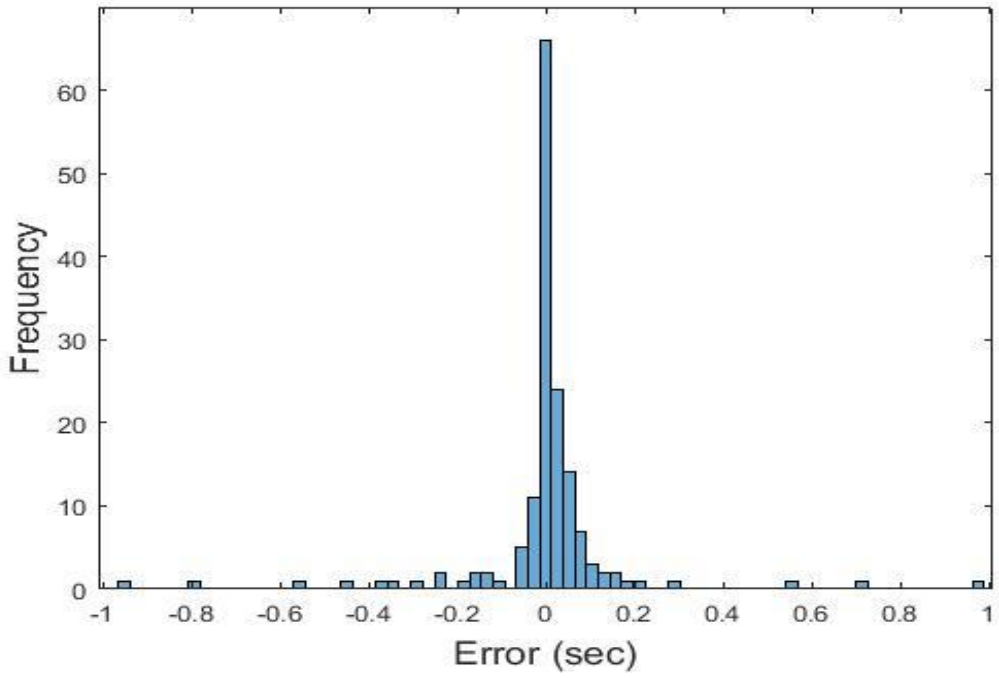

Figure 4.12: Prediction error histogram for noisy irregular waves

SNR=1000:1, the predicted data moderately overlaps with the actual data. This indicates noise sensitivity of the prediction. The small spikes in predicted (red) curve represent the reinitialization of a new training/prediction cycle. Fig. 4.12 shows the prediction error histogram for 6 second sample wave period with filter order 15 and SNR=1000:1. Table 4.7 shows the prediction results with noisy data. From the table, it can be observed that, for 6 second wave period with SNR=0, all the filter orders give almost the same replica of the truth half cycle duration with no error. However, when the SNR=1000:1, filter order 15 gives the best prediction with 0.08 half cycle duration error, rest of the filter orders give 0.14 second error. For 7 second wave period with SNR=0, filter order 15 again gives the best prediction with no error, where filter order 7 and 10 give 0.3 second error. However, with SNR=1000:1 all 3 filter orders give error of 0.9 second. Again for 8 second wave period and SNR=0, filter order 7 and 10 gives the best accuracy with an error of 0.01 second. For the same wave period when the SNR is 1000:1, all 3 filter orders gives almost the same accuracy. Again for 9 second wave period with SNR=0, filter order 7 gives the best accuracy with an error of 0.01 second. Although with SNR=1000:1 filter order 7 and 15 gives almost the same accuracy. For 10 second wave period with SNR=0, filter order 10 gives the same accuracy with an error of 0.03 second, although with SNR=1000:1, filter order 15 gives the best accuracy with an error of 1.25 second. However, the normalized sum square error based on examined filter order for overall 6,7,8,9 & 10 second and both SNR=0 and SNR=1000:1 is less in filter order 15, which is 0.28 second, whereas filter order 7 error is 0.32 second and filter order 10 error is 0.3 second. According to the prediction results filter order 15 gives the best prediction. In addition, with a filter order of 15, the number of zero crossings in the actual and predicted data series is identical or, almost the same replica.

Power Extraction simulation results for noisy irregular waves are shown in Table 4.8.

Table 4.7: Half cycle duration prediction results for noisy irregular waves

Wave Period (Sec)	SNR	Filter Order (N)	Normalized SSE (Sec) of Half Cycle Duration Prediction	Zero Crossings Number	
				Real	Predicted
6	0	7	0	158	158
		10	0		158
		15	0		158
	1000:1	7	0.14	138	138
		10	0.14		138
		15	0.08		138
7	0	7	0.3	136	134
		10	0.3		134
		15	0		136
	1000:1	7	0.9	138	138
		10	0.9		138
		15	0.9		138
8	0	7	0.01	120	120
		10	0.01		120
		15	0.03		120
	1000:1	7	0.07	121	121
		10	0.07		121
		15	0.08		121
9	0	7	0.01	109	109
		10	0.05		109
		15	0.03		109
	1000:1	7	0.23	109	109
		10	0.25		109
		15	0.24		109
10	0	7	0.22	100	100
		10	0.03		100
		15	0.22		100
	1000:1	7	1.27	102	102
		10	1.27		102
		15	1.25		102

Table 4.8: Simulation power extraction results for noisy irregular waves

Wave Period (Sec)	SNR	Filter Order (N)	Truth Output Power (kw)	Predicted Output Power (kw)	Predicted Power Change (%)
6	0	7	29.28	29.27	-0.034
		10		29.27	-0.034
		15		29.27	-0.034
	1000:1	7	29.29	28.03	-4.495
		10		28.93	-1.244
		15		29.10	-0.652
7	0	7	31.74	28.79	-10.247
		10		28.79	-10.247
		15		31.70	-0.126
	1000:1	7	31.76	28.02	-13.348
		10		28.80	-10.277
		15		28.70	-10.662
8	0	7	35.02	34.90	-0.343
		10		34.90	-0.343
		15		34.90	-0.343
	1000:1	7	35.65	34.90	-2.149
		10		35.00	-1.857
		15		35.00	-1.857
9	0	7	37.38	37.09	-0.782
		10		37.12	-0.700
		15		37.04	-0.917
	1000:1	7	37.38	34.46	-8.474
		10		34.17	-9.394
		15		34.22	-9.234
10	0	7	32.37	32.40	0.093
		10		32.40	0.093
		15		31.60	-2.437
	1000:1	7	31.53	30.10	-4.751
		10		30.20	-4.404
		15		30.20	-4.404

From the data shown in Table 4.8, it can be observed that for 6 sec wave period with SNR=0, the power is reduced by 0.034 %. For SNR=1000:1, filter order 15 gives the best power extraction with 0.625 % reduction. Again, for wave period of 7 second with SNR=0, filter order 15 extracts higher power with a reduction of 0.126 % but with SNR=1000:1, filter order 10 gives less percentage reduction of power, which is 10.277 %. For wave period of 8 second with SNR=0, the power is reduced by 0.343 %, where with SNR=1000:1, filter order 10 and 15 both give the best result with 1.857 % reduction. However, for wave period of 9 second with SNR=0, filter order 10 results less reduction of power. Finally, for wave period of 10 second with SNR=1000:1, filter order 15 gives the best result with a power reduction of 4.404 %. In terms of filter orders of overall examined wave periods, a filter order of 15, gives the best power generation with a power reduction of 3.06 %, whereas filter order 7 gives 4.45 % power reduction and filter order 10 gives a power reduction of 3.84 %.

CHAPTER 5: CONCLUSION AND FUTURE WORK

A control methodology for regular and irregular ocean waves is applied in a slider crank WEC system to maximize energy extraction by ensuring that the WEC generator can rotate in resonance with the wave excitation force. The wave excitation force prediction algorithm is based on the AR filter model and examined with three filter orders 7,10 & 15 and with five commonly found ocean wave periods of 6, 7, 8, 9 & 10 seconds. The prediction algorithm first tested with the regular waves to prove its validity. After that, it was tested with irregular waves and finally tested with noisy irregular wave data. Moreover, the prediction strategy requires only the future half-period duration, not the future amplitude of the wave force, which greatly alleviates the prediction challenges. The buoy used in this slider crank model is semi-submerged spherical buoy. The irregular waves are generated with the JONSWAP spectrum and the irregular wave force generator, in order to generate the noisy data Signal to Noise Ratio (SNR) is introduced in the wave force generator. In this study, the AR filter model is identified and utilized in MATLAB^{TM} environment, which renders fairly accurate prediction with forward-backward estimation approach. Then the offline prediction results are implemented to the simulation model. The simulation model is developed in SIMULINK^{TM}.

This study presents a unique prediction strategy, where the prediction horizon is adapted continuously with the change of length of half cycle duration, whereas in the initial research [11], prediction horizon was considered constant, which is not practically viable for real time applications. The prediction and simulation results show that, a filter order of 15 gives a fairly accurate prediction results for the most energy extraction cases of regular and irregular waves, satisfying the real-time processing requirement to validate the feasibility of the system under

practical ocean wave conditions. The prediction methodology proposed in this study can also be applied to other WEC control schemes such as latching control, which heavily relies on phase prediction.

Future work of this study includes the extensive noise analysis of this prediction algorithm with a more noise corrupted sensor measurements of wave data. In real life applications, sensor data includes noise at different levels. This is something that needs to be tested to validate the performance of proposed prediction algorithm. Once the performance under the presence of noise is satisfactory, the system will be tested in a hardware in the loop simulation environment.

REFERENCES

[1] Nada Kh. M. A. Alrikabi, “Renewable Energy Types”, Journal of Clean Energy Technologies, vol. 2, No. 1, January 2014.

[2] T. Brekken, Fundamentals of ocean wave energy conversion, modeling, and control, in: 2010 IEEE International Symposium on Industrial Electronics. (ISIE), pp. 3921, 3966, 4–7 July 2010.

[3] K. Rhinefrank et al. “Comparison of direct-drive power takeoff systems for ocean wave energy applications,” IEEE J. Ocean. Eng., vol. 37, no.1, pp. 35–44, Jan. 2012.

[4] J. R. Halliday, D. G. Dorrell, and A. R. Wood, "An application of the Fast Fourier Transform to the short-term prediction of sea wave behavior," Renew Energ, vol. 36, pp. 1685-1692, Jun 2011.

[5] D. Q. Truong and K. K. Ahn, "Wave prediction based on a modified grey model MGM(1,1) for real-time control of wave energy converters in irregular waves," Renew Energ, vol. 43, pp. 242-255, Jul 2012.

[6] Liang Li, Zhiming Yuan, Yan Gao , Xinshu Zhang, ““Wave force prediction effect on the energy absorption of a wave energy converter with real-time control” IEEE Transactions on Sustainable Energy, 29 May, 2018.

[7] M. Ge and E. C. Kerrigan, "Short-term ocean wave forecasting using an autoregressive moving average model," in Control (CONTROL), 2016 UKACC 11th International Conference on, 2016, pp. 1-6

[8] N. M. Tom, Y. H. Yu, A. D. Wright, and M. J. Lawson, "Pseudo-spectral control of a novel oscillating surge wave energy converter in regular waves for power optimization including load reduction," Ocean Eng, vol. 137, pp. 352-366, Jun 1 2017.

[9] F. Fusco, and J. V. Ringwood, “Short-term wave forecasting for real-time control of wave energy converters,” IEEE Trans. Sustainable Energy, vol. 1, no. 2, pp. 99, 106, July 2010.

[10] Y. Sang et al., “Resonance control strategy for a slider crank WEC power take-off system,” in Proc. MTS/IEEE OCEANS ’14, St. John’s, Canada, 2014, pp. 1–8

[11] Y. Sang et al., “Irregular Wave Energy Extraction Analysis for A Slider Crank WEC Power Take-off System,” ACEMP-OPTIM-Electromotion 2015 Joint Conference, Side, Turkey, 2-4 September 2015.

[12] Y. Sang et al., “Energy Extraction from A Slider-Crank Wave Energy Converter under Irregular Wave Conditions,” IEEE/MTS Oceans 15, Washington D.C., USA, 19-22 October 2015.

[13] Matlab 2017a System Identification Toolbox.

[14] Md Rakib Hasan Khan, H. Bora Karayaka,Yanjun Yan,Peter Tay,Yi-Hsiang Yu, “Slider Crank WEC Performance Analysis with Adaptive Autoregressive Filtering” IEEE Southeast Conference, 2019, Huntsville, Alabama, USA, 11-14 April 2019.

[15] K. Hasselman et al., “Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP),” German Hydrographic Institute, Germany, Tech. Rep. 12, 1973.

[16] W. E. Cummins, “The impulse response function and ship motions,” Schiffstechnik, vol. 9, pp. 101–109, 1962.

[17] E. Tedeschi et al., “Effect of control strategies and power take-off efficiency on the power capture from sea waves,” IEEE Trans. Energy Convers., vol. 26, no. 4, pp. 1088–1098, Dec. 2011.

[18] D. Wallace et. al., “Optimum Parameter Search for a Slider-Crank Wave Energy Converter under Regular and Irregular Wave Conditions,” IEEE SoutheastCon, Concord NC, USA, March 30-April 2 2017.

APPENDIX A: SOURCE CODE

%% Irregular Wave Force Generation “Irregular_Wave_Force.m”

```
clear;clc;close all;
%=================================================================%
% initial inertia: 10
% initial viscous friction coefficient: 0.32
%=================================================================%
%Callback for the simulink model
Ts=20e-6; % Sampling time
Td=1e-3; % Discrete Sampling time
%%% setting 1 %%%
gr=110;    % Gear ratio
 %=====================
aa=20e-6/(.5+20e-6);
%=================================================================%
%Slider-Crank initialization
global r      % Radius of crank. used again in the rk4sys_step function and slider crank function.
global l              % Length of rod, used again in the slider crank function.
global dr_dsb      % (Used to be r+A) Distance between the lowest edge of the crank and the
reference water surface
r=.5;          % Radius of crank. used again in the rk4sys_step function and slider crank function.
l=1;                  % Length of rod, used again in the slider crank function.
```

```
lambda=r/l;              % used again in the slider crank function.
B=0.01;                  % Viscous friction, used again in the slider crank function.
J=10;                    % inertia of flywheel, used again in the slider crank function.
dr_dsb=l;                % (Used to be r+A) Distance between the lowest edge of the crank and the
reference water surface
mcrp=10;                 % Total of mass of piston (or slider) and connecting rod respectively.
%================================================================%
% Hydrodynamics initialization (frequency domain)
delta_omega=0.01;
omega=0.5:delta_omega:1.4;
N=length(omega);
fn=omega/2/pi;% frequencies of the wave components
%%%=======================================%%%
%%% Settings for irregular wave parameters %%%
% Equivalent energy transfer: Hm0=2*sqrt(2)*A  (A is the amplitude of the regular wave)
Hm0=sqrt(2); % significant wave height of the irregular wave. The same value is used as that in
"Effect of..."
Tp=6; % If this changes, int_S_star has to be recalculated. Peak period of the irregular wave. In
"Effect of...", they used an average period of 6. We can use our own to make the spectrum fit our
need.
%%%=======================================%%%
fp=1/Tp;
g=9.81; % gravity acceleration
```

```
rho=1020;% water density
%=========================================================================%
% Choose Spectrum for the System:
flag = 1; % 0 for Breschneider model and 1 for JONSWAP Model
switch flag
   case 0
% ====================  Bretschneider model  ============================%
%        R=(Tp/1.057)^(-4); % These are calculated separately for the sake of the organing the
formula
%        Q=R*Hm0^2/4;% These are calculated separately for the sake of the organing the
formula
%        S=Q*fn.^(-5).*exp(-R*fn.^(-4)); % Bretschneider spectrum ("sea spectra revisited" or
MIT OCW slides)
       S=Hm0^2/4*(1.057*fp)^4*fn.^(-5).*exp(-5/4*(fp./fn).^4); %According to
WEC_Sim_User_Manual_v1.0.pdf
   case 1
% ==============  JONSWAP Model ==================================%
       m0=sqrt(Hm0/4); % wave field variance. See "On control ...".
 %alpha=0.0081; % a given constant which is used in most references, see "sea spectra
revisited".
       gamma=6;% If this changes, int_S_star has to be recalculated. The average of gamma is 3.3
(see "sea spectra revisited"). enhancement factor by which the P_M peak energy is multiplied to
get the peak energy value of the spectrum.
```

```
%Increasing gamma has the effect of reducing the spectral bandwidth,
%thereby increasing periodicity of the wave field. See "On control ...".
for i2=1:N
    if fn(i2)<=fp
        sigma=0.07;%if f<fp  sigma is the width factor of the enhanced peak, see "sea spectra revisited". The numbers are given in "sea spectra revisited".
    elseif fn(i2)>fp
        sigma=0.09;%if f>fp
    end
%=====================================================================%
    % the following eqn is from On Control of a Pitching and Surging Wave Energy Converter-HYavuz.pdf
    %     S(i2)=5*m0/fp*((fp/fn(i2))^5)*exp(-5/4*((fp/fn(i2))^4))*gamma^exp(-(fn(i2)/fp-1/(2*sigma^2)));
%=====================================================================%
    % the following eqn is from sea_spectra_revisited.pdf and Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP)_Jonswap-Hasselmann1973.pdf
    %     S(i2)=alpha*g^2*(2*pi)^(-4)*fn(i2)^(-5)*exp(-5/4*((fp/fn(i2))^4))*gamma^exp(-(fn(i2)-fp)^2/(2*sigma^2*fp^2));
%=====================================================================%
    % The following eqn uses basic spectrum from "On control ..." and peak enhancement factor from "Sea_spectra_revisited".
```

```
        S(i2)=5*m0/fp*((fp/fn(i2))^5)*exp(-5/4*((fp/fn(i2))^4))*gamma^exp(-(fn(i2)-
fp)^2/(2*sigma^2*fp^2));
%=========================================================================%
        % The following eqn is according to WEC_Sim_User_Manual_v1.0.pdf
        % integral of
        % 9.81^2/(2*pi)^4*x^(-5)*exp(-5/4*(0.125/x)^4)*6^exp(-((x/0.125-1)/(sqrt(2)*0.07))^2)
        % from 0 to 0.125 = 37.61 calculated by Wolframalpha
        % integral of
        % 9.81^2/(2*pi)^4*x^(-5)*exp(-5/4*(0.125/x)^4)*6^exp(-((x/0.125-1)/(sqrt(2)*0.09))^2)
        % from 0.125 to infinity=65.8056 calculated by Wolframalpha
        switch Tp
           case 6
              int_S_star=11.9001+20.8213;
           case 7
              int_S_star=22.0463+38.574;
           case 8
              int_S_star=37.61+65.8056;
           case 9
              int_S_star=60.244+105.408;
           case 10
              int_S_star=91.8214+160.658;
        end
```

```
        alpha=Hm0^2/(int_S_star*16); %int_S_star should be changed when Tp or gamma
changes.
        GAMMA=exp(-((fn(i2)/fp-1)/(sqrt(2)*sigma))^2);
        S(i2)=alpha*g^2/(2*pi)^4*fn(i2)^(-5)*exp(-5/4*(fp/fn(i2))^4)*gamma^GAMMA;
    end
end
 plot(omega/(2*pi),S)
 grid on
 axis([0.08 0.26 0 3.5])
 xlabel('f (Hz)')
 ylabel('Spectral Density (m^2s)')
 title('JONSWAP Spectrum')
%=========================================================================%
% Wave elevation and excitation force (time domain)
Start_Time=0;           % time start
End_Time=500;           % final time
Interval=0.01;          % sampling time interval
t=Start_Time:Interval:End_Time;
M=length(t);
%%% setting 2 %%%
a=5; % buoy radius
%======================
c=rho*g*pi*a^2; % a coefficient that is used later
```

```
%%% setting 3 %%%
A=sqrt(2*S*delta_omega/2/pi); % calculate amplitude for each wave component
%======================
%%% setting 5 %%%
Phase=2*pi*rand(1,N); % randomly generate the initial phase of each wave component
%======================
Ka=[0 0.05 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.2 1.4 1.6 1.8 2.0 2.5 3.0 4.0 5.0 6.0 7.0 8.0
9.0 10.0]';
Amass=[0.8310 0.8764 0.8627 0.7938 0.7157 0.6452 0.5861 0.5381 0.4999 0.4698 0.4464
0.4284 0.4047 0.3924 0.3871 0.3864 0.3884 0.3988 0.4111 0.4322 0.4471 0.4574 0.4647 0.4700
0.4740 0.4771]';
Damping=[0 0.1036 0.1816 0.2793 0.3254 0.3410 0.3391 0.3271 0.3098 0.2899 0.2691 0.2484
0.2096 0.1756 0.1469 0.1229 0.1031 0.0674 0.0452 0.0219 0.0116 0.0066 0.0040 0.0026 0.0017
0.0012]';
len=length(Ka);
kappa=zeros(1,len);
imkap=zeros(1,len);
rekap=zeros(1,len);
mm=rho*(2*pi/3)*a^3;
Sb=rho*g*pi*a^2;%785890;
kappa(1)=1;
imkap(1)= 2*Damping(1)*Ka(1)/3;
rekap(1)= sqrt(kappa(1)^2-imkap(1)^2);
```

```
for j=2:len
    kappa(j)= sqrt(4*Damping(j)/(3*pi*Ka(j)));
    imkap(j)= 2*Damping(j)*Ka(j)/3;
    rekap(j)= sqrt(kappa(j)^2-imkap(j)^2);
end
Kaq=omega.^2/g*a;
kappa_im=zeros(1,N);
kappa_re=zeros(1,N);
kappa_angle=zeros(1,N);
kappa_abs=zeros(1,N);
for i1=1:N
    kappa_abs(i1)=interp1(Ka,kappa,Kaq(i1),'cubic');
    kappa_im(i1)=interp1(Ka,imkap,Kaq(i1),'cubic');
    kappa_re(i1)=interp1(Ka,rekap,Kaq(i1),'cubic');
    kappa_angle(i1)=atan(kappa_im(i1)/kappa_re(i1));
end
%%%
% kap=0.502764572022028;
%%%
% eta=zeros(1,M);
% Fe=zeros(1,M); % initialization for wave force at each time point
Fe=@(t)0;
eta_total=@(t)0;
```

```
%%% setting 5 %%%
% omega=2*pi/6*ones(1,N);
% kappa_angle=0;
%=====================
for i=1:N
    eta{i}=@(t)A(i)*sin(omega(i)*t+Phase(i)+kappa_angle(i));
    Fe_components{i}=@(t)c*kappa_abs(i)*eta{i}(t);
    Fe=@(t)Fe(t)+Fe_components{i}(t);
    eta_total=@(t)eta_total(t)+eta{i}(t);
end
%
Fe=@(t)kap*rho*g*pi*a^2*(eta{1}(t)+eta{2}(t)+eta{3}(t)+eta{4}(t)+eta{5}(t)+eta{6}(t)+eta{7
}(t)+eta{8}(t)+eta{9}(t)+eta{10}(t));%zw(t);

% for i=1:M
%     eta(i)=sum(A.*sin(omega*t(i)+Phase));
%     Fe(i)=sum(c*kappa_abs.*A.*sin(omega*t(i)+Phase+kappa_angle));
% end
figure;
% subplot(2,1,1)
% plot(t,eta);
% grid
% title('wave elevation')
```

```
% subplot(2,1,2)
plot(t,Fe(t));
grid
%title('excitation force')
xlabel('Time(s)')
ylabel('Excitation Force')
% hold on;
figure;
plot(t,eta_total(t));
grid
title('wave elevation')
Ocean_Wave_AccP.signals.values=Fe(t)';
Ocean_Wave_AccP.time=t';
```

%% Regular Wave Force Generation “mechanical_energy.m”

```
% clear;clc;close all;
%=================================================================%
% initial inertia: 10
% initial viscous friction coefficient: 0.32
%=================================================================%
%Callback for the simulink model
Ts=20e-6; % Sampling time
T_d=1e-3; % Discrete Sampling time
%%% setting 1 %%%
gr=110;    % Gear ratio
%======================
aa=20e-6/(.5+20e-6);
%=================================================================%
%=========================Initialization==============================%
% global Interval A r l lambda B J L_af V_f r_f I_f L_aa r_a kv dr m R Sb
% %Hydrodynamics initialization
% Start_Time=0;            % time start
% End_Time=500;             % final time
% Interval=0.01;           % simpling time interval
rho=1020;                 % the density of water
g=9.81;                   % acceleration of gravity
a=5;%0.9533;              % buoy radius
```

```
% Rv=10;              % Viscous force coefficient
% Rf=0;               % Friction force coefficient
% %omega=1;           % The angular velocity of water wave
% A=0.5;              % The maximum amplitude of water wave, initialized again in the slider
crank function.
% f=1/10;             % The frequency of water wave
% omega=2*pi*f;       % The angular velocity of water wave
% k=omega^2/g;        % Wave number for infinite water depth
% Kaq=k*a;            % ka
% zw=@(t)A*sin(omega*t+288*pi/180);    % the function of water wave
%Slider-Crank initialization
global r              % Radius of crank. used again in the rk4sys_step function and slider crank
function.
global l              % Length of rod, used again in the slider crank function.
global dr_dsb         % (Used to be r+A) Distance between the lowest edge of the crank and
the reference water surface
r=0.5;                % Radius of crank. used again in the rk4sys_step function and slider crank
function.
l=1;                  % Length of rod, used again in the slider crank function.
lambda=r/l;           % used again in the slider crank function.
B=0.01;               % Viscous friction, used again in the slider crank function.
J=10;                 % inertia of flywheel, used again in the slider crank function.
```

```
dr_dsb=l;              % sqrt(l^2-r^2);              % Distance between the lowest edge of the crank
and the reference water surface
mcrp=10;               % Total of mass of piston (or slider) and connecting rod respectively.
%Fu=zeros(1,(End_Time-Start_Time)/Interval+1);
% %Generator initialization
% L_af = 1.234; % Mutual inductance between the field and the rotating armature coils.
% V_f = 220; % Field voltage.
% r_f = 150; % Resistance of field windings
% I_f = V_f/r_f; % Current of field windings
% L_aa = 0.016; % Self-inductance of the field and armature windings.
% r_a = 0.78;  % Resistance of the armature coils.
% kv = L_af*I_f; % Stator constant
%=================================================================%
% %===Calculating mu, epsilon and kappa through graphical observation======%
% Ka=[0 0.05 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.2 1.4 1.6 1.8 2.0 2.5 3.0 4.0 5.0 6.0 7.0 8.0
9.0 10.0]';
% Amass=[0.8310 0.8764 0.8627 0.7938 0.7157 0.6452 0.5861 0.5381 0.4999 0.4698 0.4464
0.4284 0.4047 0.3924 0.3871 0.3864 0.3884 0.3988 0.4111 0.4322 0.4471 0.4574 0.4647 0.4700
0.4740 0.4771]';
% Damping=[0 0.1036 0.1816 0.2793 0.3254 0.3410 0.3391 0.3271 0.3098 0.2899 0.2691
0.2484 0.2096 0.1756 0.1469 0.1229 0.1031 0.0674 0.0452 0.0219 0.0116 0.0066 0.0040 0.0026
0.0017 0.0012]';
% kappa(1)=1;
```

```
% for i=2:length(Ka)
%     kappa(i)=sqrt(4*Damping(i)/(3*pi*Ka(i)));
% end
% Mu = interp1(Ka,Amass,Kaq','cubic');
% Ep = interp1(Ka,Damping,Kaq','cubic');
% kap= interp1(Ka,kappa,Kaq','cubic');
%=====================================================================%
%Calculating Coefficients of the Differential Equation of Buoy Displacement
dr_dsb=1;
Sb=rho*g*pi*a^2;%785890;
mm=rho*(2*pi/3)*a^3;
% m=mm*(1+Mu);%267040+156940;
% R=Rv+Rf+Ep*omega*mm;%91520;
% Fe=@(t)kap*rho*g*pi*a^2*zw(t);
% t = Start_Time:Interval:End_Time;
% figure;
% % subplot(2,1,1)
% % plot(t,eta);
% % grid
% % title('wave elevation')
% % subplot(2,1,2)
% plot(t,Fe(t));
% grid
```

```
% title('excitation force')
% % hold on;
% figure;
% plot(t,zw(t));
% grid
% title('wave elevation')
% Ocean_Wave_AccP.signals.values=Fe(t)';
% Ocean_Wave_AccP.time=t';
%Call to find initial angle
Theta_Initial=Initial_Angle_Solver();
[bs,as]=RadiationKomega(a,T_d);
% load az
% load bz
Wave_Analysis;
%Calculate initial position in case of complex conjugate control
% init_z=-max((Fe(t)/(4*R*pi*f)))
```

```
%% Initial angel solver function “Initial_Angle_Solver.m”
function Theta_Initial=Initial_Angle_Solver()
format long;
%=================================================================%
%Slider-Crank initialization
global r                % Radius of crank. used again in the rk4sys_step function and slider crank
function.
global l                % Length of rod, used again in the slider crank function.
global dr_dsb           % (Used to be r+A) Distance between the lowest edge of the crank and
the reference water surface
%=================================================================%
f1=@(u)(dr_dsb-sqrt(l^2-(r*sin(u))^2))/r;
f2=@(u)cos(u);
Theta_Initial=pi/2;
err=1;
while err>1e-12
    f1n=f1(Theta_Initial);
    f2n=f2(Theta_Initial);
    Theta_Initial=acos(f1n);
    err=abs(f1n-f2n);
end
disp('The Initial Angle is (in radian): ');
disp(Theta_Initial);
```

```
disp('In degrees: ');
disp(Theta_Initial/pi*180);
```

```
%% Wave analysis program “Wave_Analysis_Prediction.m”
% ==============Output==============
% T_s are the half periods
% T1_s are the time point of zero-crossings
Excitation_Force=Ocean_Wave_AccP.signals.values;
Ocean_Wave_AccP.time=Ocean_Wave_AccP.time;
l_Fe=length(Excitation_Force);
i_T=1;
for index=2:l_Fe
   if Excitation_Force(index)*Excitation_Force(index-1)<=0  %0-crossing detection
      if Excitation_Force(index)>Excitation_Force(index-1)
         pn_flag(i_T)=1;
      else pn_flag(i_T)=0;
      end
      T1_s(i_T)=t(index-1);
      if i_T>1
         T_s(i_T)=T1_s(i_T)-T1_s(i_T-1);
      else
         T_s(i_T)=0;
      end
      i_T=i_T+1;
   end
end
```

```
%% Noisy irregular wave generation
SNR=1/1000;
for i=1:N
    eta{i}=@(t)A(i)*(sin(omega(i)*t+Phase(i)+kappa_angle(i))+(randn(size(t))*SNR));
    Fe_components{i}=@(t)c*kappa_abs(i)*eta{i}(t);
    Fe=@(t)Fe(t)+Fe_components{i}(t);
    eta_total=@(t)eta_total(t)+eta{i}(t);
end
```

%% Prediction Code “half_period_even_updated_stats.m”

```
clear all;
close all;
clc;
load c_5_t_9_snr_1_500
%% 10 ms sampling
t = Ocean_Wave_AccP.time;
% x = Ocean_Wave_AccP.signals.waves;                % wave elevation
y = Ocean_Wave_AccP.signals.values;                % excitation force
%% Collecting data until ten zero crossing to train
f_ori=0;                                  % Variable initialize to store the original value
lnp_ori=0;                                % Variable for zero crossing number
b_ori=1;                                  % Variable to loop the actual value index
Y_data_zero=[];                           % Array for zero crossing index of actual data
actual_zero_crossing_ori=[];              % Array for zero crossing time of actual data
actual_number_of_zero_crossing_ori=[];    % Array contains number of zero crossing in
actual data
init_zeroc = 10;                          % Initial number of zero crossings to be detected
while lnp_ori<init_zeroc                  % Condition to run the loop until ten zero crossing
   yN_first(f_ori+1)=y(b_ori);            % Store the original value to yN_first
   if f_ori>0                             % Condition to check the zero crossing of actual data
        e_index_ori=f_ori;
```

```
            if (y(e_index_ori)<0 && y(e_index_ori+1)>0)|| (y(e_index_ori)>0 &&
y(e_index_ori+1)<0) % Positive & Negetive zero crossing condition
                Y_data_zero=[ Y_data_zero,(e_index_ori+1)];                % Array
to store zero crossing index
                actual_zero_crossing_ori=[actual_zero_crossing_ori,t(e_index_ori+1)];          %
Array to store zero crossing time
                actual_number_of_zero_crossing_ori=[actual_number_of_zero_crossing_ori
length(actual_zero_crossing_ori)];  % Array for zero crossing length
                lnp_ori=length(actual_number_of_zero_crossing_ori);                %
Number of zero crossing
            end
    end
    f_ori=f_ori+1;
    b_ori=b_ori+1;
end
Ocean_Wave_AccP.time = t(f_ori:end)-t(f_ori);      % Cut the time of the truth based on the first
10 zero crossing length
Ocean_Wave_AccP.signals.values = y(f_ori:end);      % Cut the excitation force of the truth
based on the first 10 zero crossing length
yN=length(yN_first);                    % Use five cycles or ten zero crossings to train
y_test = y(yN+1:end);                   % Data needs to be predicted
Data_for_iteration=length(y_test);          % The length of the data to be predicted
YH_data=zeros(length(y_test),1);            % Preallocating the size of predicted data
```

```
%% Prediction horizon length selection
T_step =ceil(4*yN/init_zeroc);                % Prediction horizon limit
N =15;                                        % filter order
var=55;                                       % Cushion size based on our experiment best results
%% initialize necessary variable to implement the filter equation
f=0;                                          % Variable for predicted data index in the prediction loop
g=0;                                          % For i>j, g=0
p=0;                                          % Variable for predicted data index for filter equation when
i<j
b=N;                                          % Filter order
d=T_step;                                     % Prediction horizon limit
aN=0;                                         % Variable for training window shifting or next prediction
horizon starting
bN=aN;                                        % Varibale used in filter equation when counter>2
%% Initialize necessary variable for while loop
total_data=0;                                 % Variable for number of data predicted
counter=0;                                    % Variable which count while loop iteration
%% Initialize necessary variable to find the zero crossing for each predicted window
twice=4;                                      % Varible to identify 4,6,8,10.... even zero crossing
estimate_zero_crossing_odd=[];                % Array contains time when  zero crossing occurs
in odd prediction.
estimate_number_of_zero_crossing_odd=[];      % Array contains number of  zero crossing
in YH_data_odd
```

```
estimate_half_cycle_duration_odd=[];              % Array contains duration between two zero
crossings of odd prediction
estimate_zero_crossing_even=[];                   % Array contains time when  zero crossing occurs
in even prediction.
estimate_number_of_zero_crossing_even=[];         % Array contains number of  zero crossing
in YH_data_even
T_s=[];                                           % Array contains duration between two zero crossings of
even prediction
YH_data_zero_odd=[];                              % Array to store zero crossing index of odd
prediction
YH_data_zero_even=[];                             % Array to store zero crossing index of even
prediction
Zero_crossing_array=[];                           % Array to store final zero crossing
T1_s=[];                                          % Array to store final zero crossing time of even prediction
pn_flag_total=[];                                 % Determine positive or negetive zero crossing
pn_flag=[];                                       % Determine positive or negetive zero crossing for only
even prediction
num=1;                                            % Variable for first zero crossing index for each horizon
started from 2nd counter
ep=0;
lnp_odd=0;                                        % Variable for zero crossing number for odd prediction
lnp_even=0;                                       % Variable for zero crossing number for even prediction
```

```
iteration_number=(Data_for_iteration);              % Iteration go on in second while loop until
total_data cross this value
wq=4;
qw=4;
%% For y_test/ actual data, the number of  zero crossing and their half cycle duration
determination
truth_zero_crossing=[];                    % Array contains time when zero crossing occurs.
number_of_true_zero_crossing=[];                % total number of  zero crossing in actual data
truth_data_zero=[];                    % Array contains time when zero crossing occurs.
truth_half_cycle_duration=[];                % Array contains the duration between two zero
crossings.
tr_lnp=0;                                % Initialize number of zero crossing
tr_ep=0;
for truth_index=1:(length(y_test)-1)
 t_index=truth_index ;                    % Index number
   if (y_test(t_index)<0 && y_test(t_index+1)>0)|| (y_test(t_index)>0 && y_test(t_index+1)<0)
% Positive & Negetive zero crossing condition
      truth_data_zero=[truth_data_zero,t_index+1];                    % Array contains
index when zero crossing occurs.
      truth_zero_crossing=[truth_zero_crossing,t(t_index+1)];                % Array
contains time when zero crossing occurs.
      number_of_true_zero_crossing=[ number_of_true_zero_crossing length(
truth_zero_crossing)];
```

```
        tr_lnp=length(number_of_true_zero_crossing);                    % Number of
zero crossings
    end
    %% truth half cycle duration determine
    if tr_lnp >=2 && tr_lnp>tr_ep
        truth_half_cycle_duration=[truth_half_cycle_duration,(truth_zero_crossing(end)-
truth_zero_crossing(end-1))]; % Array contains the duration between two zero crossings.
    end
    tr_ep=tr_lnp;
end
%% Starting of while loop until at the end of prediction
while total_data<(iteration_number)                % Loop will be continue until
total_data<Data_for_iteration
    counter=counter+1;
   switch counter
      case {1,2}
      if counter==1                          % This if loop ultimately for storing the ar coeeficient
based on counter
          f=0;                               % Initialize the index of the predicted array, see 148 line
          y_train = y(1+(counter-1)*aN:yN+(counter-1)*aN); % Train data , for counter=1;
y_train=y(1:yn); five cycles to train
          model = ar(y_train, N);            % ar filter  model
          k=model.Structure.a.Value(1:end);  % ar filter coefficient
```

```
    else
        %Prediction starts from the index point z.
        %Where, z= every first zero crossing of odd prediction cycle – a cushion length
       if bN<0 % if the first zero crossing occurs before the var value
          var=ceil(0.01*55);
          bN=YH_data_zero_odd(1)-var;
          aN=bN;
       end
        y_train = y(1+bN:yN+bN);                % Training window shifting and starts from the
point where we start our predicttion.
        model = ar(y_train, N);                 % ar filter  model
        k=model.Structure.a.Value(1:end);       % ar filter coefficient
        f=YH_data_zero_odd(1)-var;              % Prediction horizon starting point index
    end
    temp_b=b;                                   % Actual predicted data + filter order
  %% implement filter equation
 if  counter==1
  for j=1:d                                     % Loop to cover each prediction horizon limit (Predifined 2
cycles) until two zero crossings
        sum=0;
        p=j-1;                                  % Predicted data index when i<j
        for i=2:N+1                             % loop to use the AR filter co-efficients
          if i>j                                % Condition to choose actual data in filter equation
```

```
            c=-((k(i)*y_train(end-(i-j-1))));% filter equation for MATLAB:a(1)*y(n) =
b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb) - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
            g=0;
          else                          % Condition to choose predicted data in filter equation
            g=-(k(i)*YH_data(p));       % filter equation for MATLAB when i<j
            p=p-1;
            c=0;
          end
          sum=sum+c+g;                  % Sum all N data value for one predicted data, when
N=7 OR 10 OR 15
          YH_data(f+1)=sum;             % predicted data array
        end
   %% start finding zero crossing after predicting every data for first prediction window
      if f>0
      e_index=f;
          if (YH_data(e_index)<0 && YH_data(e_index+1)>0)|| (YH_data(e_index)>0 &&
YH_data(e_index+1)<0)          % Positive & Negetive zero crossing condition
              YH_data_zero_odd=[ YH_data_zero_odd,(e_index+1)];
% Array to store zero crossing index of odd prediction
              estimate_zero_crossing_odd=[estimate_zero_crossing_odd,t(e_index+1)];
% Array to store zero crossing time of odd prediction
```

```
estimate_number_of_zero_crossing_odd=[estimate_number_of_zero_crossing_odd
length(estimate_zero_crossing_odd)];% Array for zero crossing length of odd prediction
                    lnp_odd=length(estimate_number_of_zero_crossing_odd);
% Number of zero crossing in odd prediction
                end
               if lnp_odd>=2 && lnp_odd>ep              % Condition when half cycle dration needs
to be calculated
estimate_half_cycle_duration_odd=[estimate_half_cycle_duration_odd,(estimate_zero_crossing
_odd(end)-estimate_zero_crossing_odd(end-1))]; % Half cycle duration of odd prediction
              end
                ep=lnp_odd;
                  end
          b=temp_b+j;
          f=f+1;                                % Increment the index
          if lnp_odd==2                          % Condition to stop the iteration for the ongoing
prediction horizon (two zero crossings)
              aN=YH_data_zero_odd(1)-var;              % Calculating the index value to start to
predict the second zero crossing once again
              bN=aN;
              break                              % Stop iteration for current prediction horizon limit
which starts from 131 line (for j=1:d)
          end
```

```
    end
  else                                % Else condition for counter 2
      for j=1:d                       % Loop to cover each prediction horizon limit until two
zero crossings
          sum=0;
          p=aN+(j-1);                 % Predicted data index started from the second
horizon starting point and used where i<j
          for i=2:N+1                 % loop to use the AR filter co-efficients
             if i>j                   % Condition to choose actual data in filter equation
                c=-((k(i)*y_train(end-(i-j-1))));       % filter equation for MATLAB
                g=0;
             else                     % Condition to choose predicted data in filter equation
                g=-(k(i)*YH_data(p));          % filter equation for MATLAB
                p=p-1;
                c=0;
             end
             sum=sum+c+g;             % Sum all N data value for one predicted data,
when N=7,10,15
             YH_data(f+1)=sum;        % predicted data array for even prediction
          end
    %% start finding zero crossing after predicting every data for first prediction window
      if f>0
      e_index=f;
```

```
            if (YH_data(e_index)<0 && YH_data(e_index+1)>0)|| (YH_data(e_index)>0 &&
YH_data(e_index+1)<0) % Positive & Negetive zero crossing condition
                if YH_data(e_index)<0 && YH_data(e_index+1)>0
                   flag=1;                                          % Positive zero crossing
                   pn_flag_total=[pn_flag_total, flag];             % Total Flag array
                else
                   flag=0;                                          % Negetive zero crossing
                   pn_flag_total=[pn_flag_total, flag];             % Total Flag array
                end
                YH_data_zero_even=[ YH_data_zero_even,(e_index+1)];
% Array to store zero crossing index of even prediction
                estimate_zero_crossing_even=[estimate_zero_crossing_even,t(e_index+1)];
% Array to store zero crossing time of even prediction
estimate_number_of_zero_crossing_even=[estimate_number_of_zero_crossing_even
length(estimate_zero_crossing_even)];% Array for zero crossing length of even prediction
                lnp_even=length(estimate_number_of_zero_crossing_even);
% Number of zero crossing in even prediction
              end
             if lnp_even>=2 && lnp_even>ep                          %
Condition when half cycle dration needs to be calculated
               Zero_crossing_array=[Zero_crossing_array, YH_data_zero_even(num)];
% Array to store zero crossing index of even prediction
```

```
                T1_s=[T1_s, estimate_zero_crossing_even(num)];                    %
Array to store final zero crossing time of even prediction
%                T_s=[T_s,(estimate_zero_crossing_even(end)-
estimate_zero_crossing_even(end-1))];          % Half cycle duration of even prediction
                pn_flag=[pn_flag,pn_flag_total(num)];                    %
Ultimate positive or negetive zero crossing for only even prediction
                num=num+2;
              end
               ep=lnp_even;
                end
        b=temp_b+j;                                % Actual predicted data + filter order
        f=f+1;                                      % Increment the index
        if lnp_even==2          % Condition to stop the iteration for the ongoing prediction
horizon (two zero crossings)
            bN=YH_data_zero_even(end)-var;                                    %
Calculating the index value to start to predict the second zero crossing once again
             break                                                  % Stop iteration for
current prediction horizon limit which starts from 174 line counter 2 case (for j=1:d)
        end
      end
  end                                                          % End of filter
equation prediction and zero crossing detection section if-else condition of counter=1 or 2, starts
from if counter==1 (134 line)
```

```
    total_data=bN;                                  % Total data already predicted with the var
      otherwise                                     % Condition when the counter starts from 3
    y_train = y(1+bN:yN+bN);                                                    % Training
window shifting and starts from the point where we start our predicttion from counter 3.
    f=bN;                                       % Starting index value for the prediction horizon
    model = ar(y_train, N);                                  % ar filter  model
    k=model.Structure.a.Value(1:end);                        % ar filter coefficient
    temp_b=b;
    %% implement filter equation
    for j=1:d               % Loop to cover each prediction horizon size until two zero crossings
        sum=0;
        p=bN+(j-1);                     % Predicted data index
        for i=2:N+1                     % loop to use the AR filter co-efficients
          if i>j                        % Condition to choose actual data in filter equation
            c=-((k(i)*y_train(end-(i-j-1))));  % filter equation for MATLAB
            g=0;
          else                          % Condition to choose predicted data in filter equation
            g=-(k(i)*YH_data(p));           % filter equation for MATLAB
            p=p-1;
            c=0;
          end
          sum=sum+c+g;          % Sum all N data value for one predicted data, when N=7,10,15
          YH_data(f+1)=sum;                 % predicted data array
```

```
        end
    %% start finding zero crossing after predicting every data for first prediction window
      if f>0
      e_index=f;
            if (YH_data(e_index)<0 && YH_data(e_index+1)>0)|| (YH_data(e_index)>0 &&
YH_data(e_index+1)<0) % Positive & Negetive zero crossing condition
             if YH_data(e_index)<0 && YH_data(e_index+1)>0
               flag=1;
               pn_flag_total=[pn_flag_total, flag];
            else
               flag=0;
               pn_flag_total=[pn_flag_total, flag];
            end
            YH_data_zero_even=[ YH_data_zero_even,(e_index+1)];
% Array to store zero crossing index of odd prediction
            estimate_zero_crossing_even=[estimate_zero_crossing_even,t(e_index+1)];
% Array to store zero crossing time of odd prediction
estimate_number_of_zero_crossing_even=[estimate_number_of_zero_crossing_even
length(estimate_zero_crossing_even)];% Array for zero crossing length of odd prediction
            lnp_even=length(estimate_number_of_zero_crossing_even);
% Number of zero crossing in odd prediction
          end
           % estimate  half cycle duration determine
```

```
                % 1st positive zero crossing occurs
            if lnp_even==qw                                          % Condition
when half cycle dration needs to be calculated
                Zero_crossing_array=[Zero_crossing_array, YH_data_zero_even(num)];
                T1_s=[T1_s, estimate_zero_crossing_even(num)];
                T_s=[T_s,(estimate_zero_crossing_even(end-1)-estimate_zero_crossing_even(end-
3))];              % Half cycle duration of odd prediction
                pn_flag=[pn_flag,pn_flag_total(num)];
                num=num+2;
            end
%                ep=lnp_even;
                end
        b=temp_b+j;
        f=f+1;
        if lnp_even==qw                                              % Condition to
stop the iteration for the ongoing prediction horizon (two zero crossings)
            zi1=estimate_zero_crossing_even(end)-estimate_zero_crossing_even(end-1);
            if zi1<=0.55
              bbN=ceil(0.1*55);
              bN=YH_data_zero_even(end)-bbN;
              qw=qw+2;
              break                                                  % Stop iteration
            else
```

```
            bN=YH_data_zero_even(end)-var;
            qw=qw+2;
            break
          end
        end
    end
    total_data=bN;                                          % Total data
already predicted with var
  end                                                       % End of switch
statement
end
%% Error: difference between actual and predicted half cycle duration
% min_length_error=min([length(truth_half_cycle_duration) length(T_s)]);
% half_period_prediction_even_error=(truth_half_cycle_duration(1:min_length_error)-
T_s(1:min_length_error));
% mean_error=mean(half_period_prediction_even_error)
% max_error=max(abs(half_period_prediction_even_error))
% st_deviation_error=std(half_period_prediction_even_error)
% histogram(truth_half_cycle_duration(1:min_length_error)-T_s(1:min_length_error),
min_length_error)
sq1=1;
sse=0;
min_length_error=min([length(truth_half_cycle_duration) length(T_s)]);
```

```
while sq1<=min_length_error
 half_period_prediction_even_error=(truth_half_cycle_duration(sq1)-T_s(sq1));
 sse=sse+(half_period_prediction_even_error)^2;
 sq1=sq1+1;
end
sum_sq_error=sse;                                                    % Sum
square error
norm_sum_sq_error=sse/min_length_error                                    %
Normalizing the error to produce apples to apples comparison
length(truth_zero_crossing)
length(Zero_crossing_array)
```

```
%% Simulation code for truth “half_period_calc_for_truth.m”
clear all;
close all;
clc;
load Ocean_Wave_AccP_10_3
%% 10 ms sampling
t = Ocean_Wave_AccP.time;
% x = Ocean_Wave_AccP.signals.waves;              % wave elevation
y = Ocean_Wave_AccP.signals.values;             % excitation force
%% Collecting data until ten zero crossing to train
f_ori=0;                                % Variable initialize to store the original value
lnp_ori=0;                              % Variable for zero crossing number
b_ori=1;                                % Variable to loop the actual value index
Y_data_zero=[];                         % Array for zero crossing index of actual data
actual_zero_crossing_ori=[];            % Array for zero crossing time of actual data
actual_number_of_zero_crossing_ori=[];  % Array contains number of zero crossing in
actual data
init_zeroc = 10;                        % Initial number of zero crossings to be detected
while lnp_ori<init_zeroc                % Condition to run the loop until four zero crossing
   yN_first(f_ori+1)=y(b_ori);          % Store the original value to yN_first
   if f_ori>0                           % Condition to check the zero crossing of actual data
       e_index_ori=f_ori;
```

```
            if (y(e_index_ori)<0 && y(e_index_ori+1)>0)|| (y(e_index_ori)>0 &&
y(e_index_ori+1)<0) % Positive & Negetive zero crossing condition
                Y_data_zero=[ Y_data_zero,(e_index_ori+1)];                % Array
to store zero crossing index
                actual_zero_crossing_ori=[actual_zero_crossing_ori,t(e_index_ori+1)];          %
Array to store zero crossing time
                actual_number_of_zero_crossing_ori=[actual_number_of_zero_crossing_ori
length(actual_zero_crossing_ori)];  % Array for zero crossing length
                lnp_ori=length(actual_number_of_zero_crossing_ori);                %
Number of zero crossing
            end
    end
    f_ori=f_ori+1;
    b_ori=b_ori+1;
end
Ocean_Wave_AccP.time = t(f_ori:end)-t(f_ori);
Ocean_Wave_AccP.signals.values = y(f_ori:end);
```

Irregular_Wave_Force_Prediction

```
%% Irregular_Wave_Force_Prediction.m
%=============================================================%
%Callback for the simulink model
Ts=20e-6; % Sampling time
T_d=1e-3; % Discrete Sampling time
```

```
Wm_s=5e-4;
%%% setting 1 %%%
gr=110;    % Gear ratio
%=====================
aa=20e-6/(.5+20e-6);
%=================================================================%
%Slider-Crank initialization
global r                % Radius of crank. used again in the rk4sys_step function and slider crank
function.
global l                % Length of rod, used again in the slider crank function.
global dr_dsb               % (Used to be r+A) Distance between the lowest edge of the crank and
the reference water surface
r=.5;                % Radius of crank. used again in the rk4sys_step function and slider crank
function.
l=1;                 % Length of rod, used again in the slider crank function.
lambda=r/l;              % used again in the slider crank function.

dr_dsb=l;                % (Used to be r+A) Distance between the lowest edge of the crank and the
reference water surface
mcrp=10;              % Total of mass of piston (or slider) and connecting rod respectively.
a=5;                % buoy radius
```

```
g=9.81; % gravity acceleration

rho=1020;% water density

mm=rho*(2*pi/3)*a^3;

Sb=rho*g*pi*a^2;


%======================

% save ExFcC1 Ocean_Wave_AccP

Theta_Initial=Initial_Angle_Solver();

[bs,as]=RadiationKomega(a,T_d);

%load as

%load bs

Wave_Analysis_Prediction; %Replace this with Rakib's prediction algorithm


%Calculate initial position in case of complex conjugate control

%init_z=-max((Fe(t)/(4*R*pi*f)))
```

%% Simulation code for prediction “half_period_calc_for_prediction.m”

```
%modify for different prediction parameters

half_period_even_updated_statss

%%after running half_period_even_updated_stats

T_s_pred=T_s;

T1_s_pred=T1_s;

pn_flag_pred=pn_flag;

Irregular_Wave_Force_Prediction

%%after running Irregular_Wave_Force_Prediction

T_s=[0 T_s_pred(1:end-1)];

T1_s=T1_s_pred;

pn_flag=pn_flag_pred;

%Then run simulation

%zb1.Data(end)/500/1000
```