`

PREDICTION OF DYNAMIC PERFORMANCE OF NTUA SEMI-PLANING SERIES USING ARTIFICIAL NEURAL NETWORKS

Papantoniou Michail

Diploma Thesis

National Technical University of Athens

School of Naval Architecture and Marine Engineering

Supervisor: Gregory Grigoropoulos

Committee member: George Zaraphonitis

Committee member: George Papadakis

March, 2023

`

Contents

`

`

List of Figures-Tables

`

`

`

Scope of the study & Acknowledgments

It is well established that the prediction of seakeeping behaviour of ships is a complex and challenging task, which requires the use of advanced computational techniques. Traditionally, the seakeeping behaviour of ships has been predicted through model tests and computational fluid dynamics (CFD) calculations. However, these methods have several limitations such as time-consuming and costly process, the limited number of test cases and the dependency on complex mathematical models and the numerical methods.

The study presented therein examines the seakeeping performance of NTUA Semi-Planing Series and highlights the potentials of an application of artificial neural networks (ANNs) to predict the seakeeping performance of these hull forms in head seas with regular waves. Specifically, the systematic experimental results in regular waves for both Fn=0.34 and Fn=0.68 are presented and modelled using mathematical models including splines, polynomials, and ANN to predict the seakeeping behaviour of this Series.

ANNs do not require the explicit mathematical formulation or the numerical solution of differential equations. The application of artificial neural networks (ANNs) has been proposed as a fast and accurate approach, if properly trained and sufficient data provided, to approximate the seakeeping behaviour of any hull forms. ANNs can learn the non-linear and complex relationships between ship motions, ship parameters and sea conditions, at a provided dataset and generate accurate results for intermediate data that never seen before. Additionally, ANNs can provide predictions quickly which can be beneficial for real-time applications or for optimizing the design of a ship at the initial stage.

The above key-advantages inspired me for the development of a data-driven model which is capable to satisfactory predict the seakeeping performance of the NTUA Semi-Planing Series in regular waves. The data utilized for building the ANN model was based on tank test results, provided by the Laboratory of NTUA (Dynamic Performance of the NTUA Double – Chine Series hull forms in Regular waves, Gregory J. Grigoropoulos, Dimitra P. Damala, Theodore A. Loukakis) after extensive model tests of the semi-planing hull forms.

Overall, the study highlights the potential of ANN as a data-driven approach for predicting the seakeeping behaviour of NTUA Semi-Planing Series, which is capable to provide prompt and satisfactory results. This model can serve as a library for obtaining response amplitude operators for heave, pitch, accelerations, and added resistance of such hull forms within the range of λ/L between 0.5 and 3.5, eliminating the need for model tests or computational fluid dynamics (CFD) simulations. This is significant since there are only a few available results in the literature pertaining to semi-planing hull forms.

The successful completion of this thesis would not have been possible without the proper guidance and feedback of my supervisor, *Mr. G.J. Grigoropoulos*. In this respect, I would like to express my sincere gratitude to supervisor and to my family as well for their time, attention, support and encouragement throughout my thesis.

`

1. SEAKEEPING PERFORMANCE OVERVIEW

Seakeeping performance refers to a vessel's ability to maintain stability and manoeuvrability in rough seas. It is an important aspect of vessel design and operation, as it can affect crew comfort and safety, cargo integrity, and overall efficiency. Seakeeping performance is influenced by factors such as hull design, weight distribution, propulsion system, and environmental conditions such as wave height and frequency. Shipbuilders use methods such as physical model tests and numerical simulations, to evaluate and optimize seakeeping performance at the preliminary design stage.

1.1 Seakeeping & vessel motions

In the broadest terms, seakeeping is the ability of the vessel to withstand different sea conditions**.** Seakeeping is considered a measure that indicates the performance of a ship. However, seakeeping performance differs between different types of vessels, missions to accomplish and external conditions. A vessel is considered to be seaworthy if combines efficient operation and human safety when subjected to waves. According to the routes at which the vessel will operate, weather conditions impact significantly the seakeeping performance and this impact has to be obtained at the preliminary design stage. Rough environmental conditions can cause safety issues. Since health and safety onboard is something that should be guaranteed at any sea conditions, the vessel responses should be tested in waves.

In reality seakeeping performance assessment of a vessel in wave conditions is a really complex problem. Vessel dynamic performance in waves is affected of many different factors relating to the environment the vessel is operating, the loading state, type of ship, speed and direction. It is quite a hard procedure to measure and model the above factors with high accuracy.

One way to deal with the seakeeping performance assessment is by conducting tank tests and measure the seakeeping responses at pre-defined environmental and vessel characteristics. Tank test is the most popular way to investigate the performance of the vessel at a preliminary design stage.

Seakeeping performance goes hand in hand with ship motions which are depicted in below figure.

All six motions of a vessel are following oscillation equations, where ξ_{j0} represent the amplitude, ω the vessel frequency and $\varepsilon_{\xi_j}$is the phase. It should be clarified that amplitudes $\xi_{j0} = \xi_{j0}(\omega)$and phases $\varepsilon_{\xi_j} = \varepsilon_{\xi_j}(\omega)$ are functions of the frequency ω. The vessel's motion can be described by the following equation:

$$\xi_j = \xi_{j0} \cos\left(\omega t + \varepsilon_{\xi_j}\right), \qquad j = 1,2 \dots 6$$

`

The first derivatives of time are described as vessel speeds, $\dot{\xi}_j(t)$, $j = 1,2 \dots 6$, while the second derivatives describe the vessel accelerations, $\ddot{\xi}_j(t)$, $j = 1,2 \dots 6$. These responds correspond to specific vessel frequency.

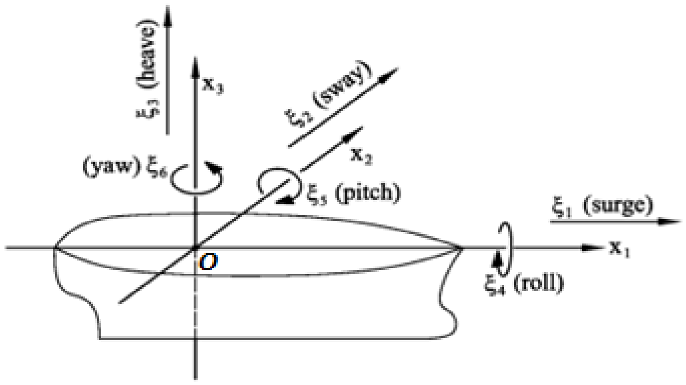


Figure 1: Source: G.A. Athanasoulis,2008, Six degrees of freedom-Vessel motions

Motion	Symbol	Definition
Surge	ξ_1	Linear motion along X_1 longitudinal axis
Sway	ξ_2	Linear motion along X_2 transverse axis
Heave	ξ_3	Linear motion along X_3 vertical axis
Roll	$\xi_4 = \theta_1$	Rotation around X_1 longitudinal axis
Pitch	$\xi_5 = \theta_2$	Rotation around X_2 transverse axis
Yaw	$\xi_6 = \theta_3$	Rotation around X_3 vertical axis

Table 1: Description and notation of vessel motions

1.2 Seakeeping analysis

Seakeeping analysis is the prediction of a vessel's responses in specified sea conditions. After analysing weather data and filtering by vessel characteristics, the vessel responses at specified environmental conditions can be obtained, at preliminary design stage, either by conducting model tests or by performing numerical simulations. To clarify here, in this particular study model test results for regular waves have been used for measuring the seakeeping performance.

The dynamic performance is evaluated by Response Amplitude Operator (RAO)**.** Represents a quantified value of the motion that a specific vessel has under certain external conditions and disturbances. Seakeeping can be considered a deterministic linear input-output system where the inputs(excitation) are represented by external forces the ship underway when at sea and the outputs(responses) are defined by RAOs. Seakeeping as a three-part problem:

`

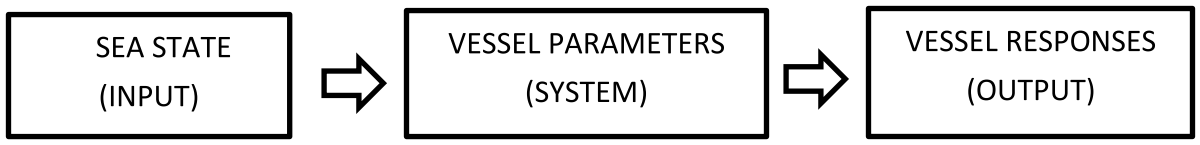


Figure 2: Seakeeping streamlined to three-part system

In the design stage of a vessel, there are three primary considerations that must be taken into account. First, the estimation of all potential environmental conditions that the vessel may encounter. Second, the identification and analysis of the vessel's dynamic response to various sea states, considering both the ship's structural and hydrodynamic characteristics. Third, the establishment of standards and guidelines for evaluating the ship's seakeeping performance under different conditions, with the aim of maximizing operational efficiency without compromising safety and performance. Hence to optimize the vessel's design the following aspects should be taken into consideration:

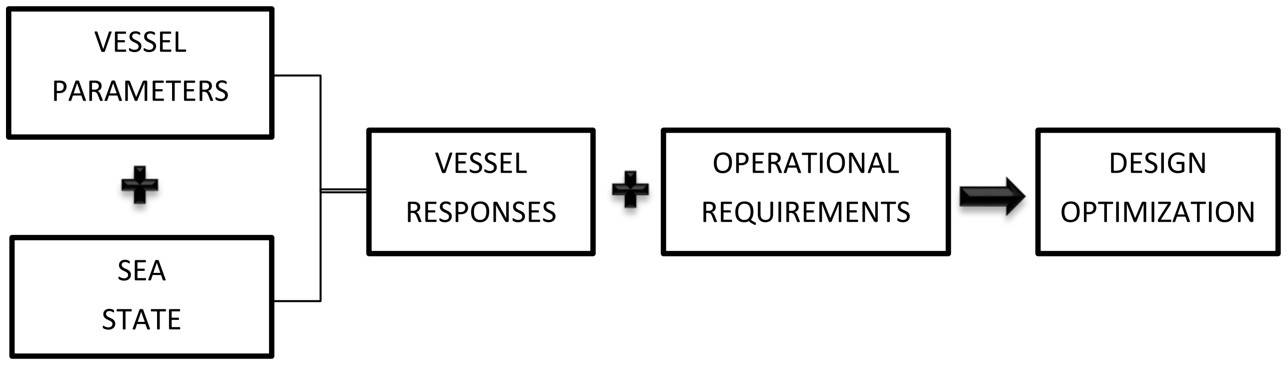


Figure 3: Design Stage Considerations

1.3 Sea State

The ocean/environmental conditions are described by sea state which is the condition of the free surface with respect to wind-waves and swell at a certain geographic region and time. At a particular sea state, the wave heights, and may a statistical parameter referring to wave period (T), should maintain within a close range. When different trends are becoming apparent the sea state changes and cannot be represented by the same statistical parameters for wave heights and periods. Sea state can be approximated by the wave spectral which is used as a statistical tool to determine theoretically the sea state. The wave elevation between t_1-t_2 time intervals is represented below along with the key parameters of sea state:

`

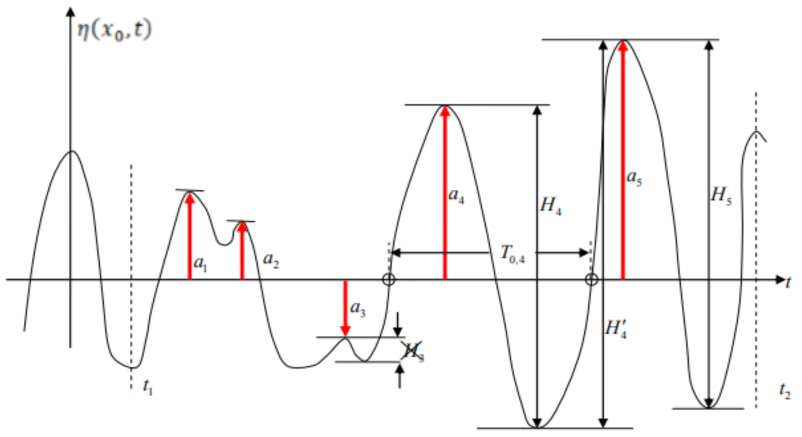


Figure 4: Quoted from G.A. Athanasoulis,2008, Wave Elevation between t1 - t2 time intervals

- Wave height (H): The absolute distance between the elevation of a crest and the next trough or the distance of the trough to the next coming crest. (i.e.: H_4, $H_{4'}$, H_5)
- Mean wave height ($\overline{\mathrm{H}}$): Mean wave heigh of all wave heights included in the sample:

$$\overline{H} = \frac{\sum_{i=1}^{i=N} H_i}{N}$$

Where N defines the sample size (number of occurrences)

- RMS wave height: Root Mean Squared wave heights and calculated as $\left(\overline{\mathrm{H}^2}\right)^{\frac{1}{2}}$
- Significant wave height (HS): The mean of highest third values of wave height included in the sample:

$$\overline{H}_{\frac{1}{3}} = \frac{1}{\frac{N}{3}} \sum_{i=1}^{i=\frac{N}{3}} H_i\,, \qquad \frac{N}{3} : integer\ number$$

- Wave period: is defined as a time duration between two sequential points having the same phase, i.e.:
  1. Zero elevation wave period (T_0): The time duration between two consecutive records of zero elevation level. This parameter is noted as $T_{0,4}$ at Figure 2.
  2. Top/Trough period (Ttop, Ttrough): The time duration between two consecutive tops/troughs respectively.
- Wave frequency (f): Is the reciprocal of the wave period and defines the number of occurrences of a repeated event per unit of time. It's a crucial term for seakeeping analysis considering that high frequencies can lead to rapid motions of the vessel and resonance phenomena may occur more frequent.
- Wavelengths (λ): Defined by the following equation:

$$\lambda = V * T,$$

Where V denotes the wave propagation speed and T is the wave period.

The waves that are often encountered in real-world conditions, such as in open seas are better represented by irregular as they represent a mixture of waves of different heights, frequencies and directions. They can be caused by a variety of factors such as wind, currents, and other vessels. However, regular waves are utilized for this thesis. The regular waves are highly related to the wave amplitude and

`

linear approach is considered sufficient to provide good estimation of the vessel's motion, where the wave heights are small compared to the size of the vessel.

1.4 Design characteristics

While sea state cannot be changed, the way a ship will perform under specified external conditions can be optimized considering design criteria. The vessel's design is what can be modified to impact the seakeeping. If the desired performance is not achieved at the preliminary design stage, the builder should make the appropriate changes for improvement in the hull form. That changes usually refer to the size and form of the hull. Excessive vessel responses should be avoided, while maintaining those values within a safe range is mandatory.

Seakeeping events like deck wetness, propeller and keel emergence, stern, bottom and bow flare slamming can cause damage on lightship structures, main engine and auxiliary engines operations, cargo and on-board safety. Bow acceleration and pitch motions are of critical importance in below seakeeping events.

- *Bottom slamming* can occur when the bottom keel is out of the water and simultaneously the relative velocity between the bottom keel and sea surface exceeds a critical limit defined as slamming speed.
- *Bow flare slamming* occurs when the keel section at the fore slams suddenly into the water and the velocity between the bow and the sea surface exceeds a critical amount at the time of contact.

Bow acceleration determines the speed at which the bow section strikes the water surface. Regarding the *pitch motion*, if the vessel pitches too much, the bow or bottom sections can slam into the water surface with significant force.

- *Deck wetness* will occur in case the relative to sea surface vertical bow motion is greater than the freeboard, causing waves to break over the bow and flood the deck.

Pitch plays a crucial role in deck wetness because excessive pitch motion can cause the bow to move too far upward or downward, allowing waves to wash over the deck. If the pitch motion is too severe, the bow of the vessel can "plunge" into the oncoming waves. Also, higher *bow acceleration* can cause the vessel to ride more roughly over the waves.

- *Propeller emergence* will occur in case the relative to sea surface vertical stern motion is greater than the ¼ of the propeller diameter.

Pitch determines the motion of the vessel's stern in relation to the water surface can cause the propeller to lift out of the water if the pitch motion exceed a certain amount.

In order to avoid the aforementioned seakeeping events and to impact the way a vessel will respond to the head-waves, the length and form of the hull can be altered appropriately to improve the dynamic performance of the vessel. To alter practically the design characteristics a parent hull is to be chosen first. After choosing the parent hull and conducting model tests, the relevant dynamic responses of the vessel can be obtained through the transferring functions (RAOs) and checked for improvement.

`

Hull size altering

All responses are tended to be significant when the wave length is greater than the vessels, while tend to be negligible when the wave length is smaller of a critical value defined as $\frac{3}{4}L_S$ .Heave and pitch responses are getting near to one when the wave length is greater that the vessels and approach zero when wave lengths are smaller than critical value. Vessels tend to follow (move with) the big waves in length and they are not responding for small ones.

As the size of the ship increases, the natural frequency of heave and pitch motion decreases. This means that larger ships will respond less to waves at the same frequency as smaller ships, resulting in a smoother ride for passengers and crew. However, as the size of the ship increases, the wetted surface area also increases, leading to higher drag and therefore a lower speed for the same power.

Larger vessels have a deeper draft and a greater waterline length and consequently greater displacement, which means they are less likely to experience bottom slamming as the bottom keel is less likely to come out of the water. Also, larger vessels typically have a greater bow flare, which can provide more protection against bow flare slamming. Larger vessels are commonly accommodated with a higher freeboard, which means they are less likely to experience deck wetness as the relative to sea surface vertical bow motion is less likely to be greater than the freeboard. Finally, larger vessels typically have a larger displacement and a greater waterline length, which means they are less likely to experience a high vertical bow acceleration as the ship is less affected by the waves.

We conclude that in general as the hull size is increasing the vertical acceleration as well as heave and pitch responses are decreasing, while the performance of the vessel in waves is enhanced. Not-withstanding the above, a detailed analysis of the specific vessel’s design and sea conditions is needed to make accurate predictions for her sea keeping behaviour.

For achieving considerable changes in the performance of the vessel the principal particulars should be altered and the impact of hull size is to be estimated from a series of experiments with models sharing the same hull form while differing the size by altering the length.

Hull form altering

Changes to the overall proportions (Length to Beam ratio, Draft to Beam ratio) can have important impact on the seakeeping motions.

The draft to beam ratio can affect the vessel's stability and can also affect the likelihood of bottom slamming. A vessel with a higher draft to beam ratio will have a greater stability and will be less likely to experience bottom slamming. This is because a vessel with a deeper draft will have more of its weight below the waterline, resulting in a lower centre of gravity and greater stability. Additionally, a vessel with a deeper draft can better resist the effects of waves and currents, which can reduce the likelihood of bottom slamming. However, a vessel with a higher draft to beam ratio may have a lower speed and manoeuvrability. This is because a deeper draft results in more wetted surface area, which can increase frictional resistance and reduce speed. In addition, a vessel with a deeper draft may have a larger turning radius, which can reduce manoeuvrability.

`

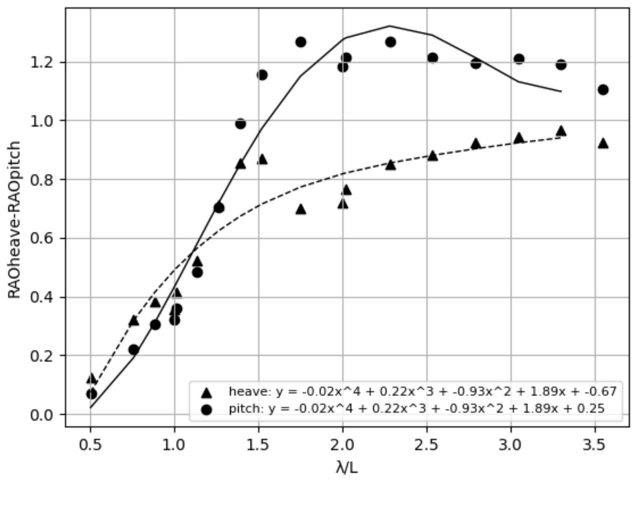
A vessel with a higher L/B ratio has a greater length compared to its beam (width) makes it more difficult for the vessel to roll from side to side. The wider the vessel, the more susceptible it is to rolling due to the increased distance between the vessel's centre of gravity and the waterline. This increased distance creates more leverage for the waves to push against, causing the vessel to roll more easily.

In general, a vessel with a higher freeboard will have a lower likelihood of experiencing deck wetness. Also, the freeboard of a vessel can affect the stability of the vessel. A vessel with a higher freeboard will have a greater stability, as it will have a greater righting arm, which will make it harder for the vessel to capsize. This is because a higher freeboard means there is a greater distance between the vessel's centre of gravity and its metacentre, which increases the lever arm and thus the righting moment. However, a vessel with a higher freeboard may be less manoeuvrable.

The type of bow and stern can also affect the likelihood of seakeeping events, as a vessel with a bulbous bow or a stern with a skeg will have a lower likelihood of experiencing slamming, deck wetness, and other seakeeping events. A vessel with a bulbous bow is redirecting the flow of water around the bow and can reduce wave resistance and also heave and pitch motions.

1.5 Seakeeping assessment (RAOs)

In the maritime industry and especially in the design field, a response amplitude operator (RAO) is a tool used to predict the performance of a vessel when exposed to various sea conditions. It is a set of statistical values that indicate how the ship will respond to different wave frequencies and amplitudes. The vessel (system) is stipulated by the waves and her response is always dependant on the transferring function (RAO(ω)) regarding this motion. To clarify that when referring to transferring function RAO(ω) is to be considered instead of RAO since RAO is obtained by dividing the amplitude of a specific vessel response, by the wave amplitude. A single, specific, measurement is required for estimating the RAO while transferring function represent best fit curves of response amplitude operator measurements. Two examples of transferring functions can be obtained from the parent hull form with Cdl=3.00 and Fn corresponding to 0.34 and 0.68. The illustration of those examples follows:



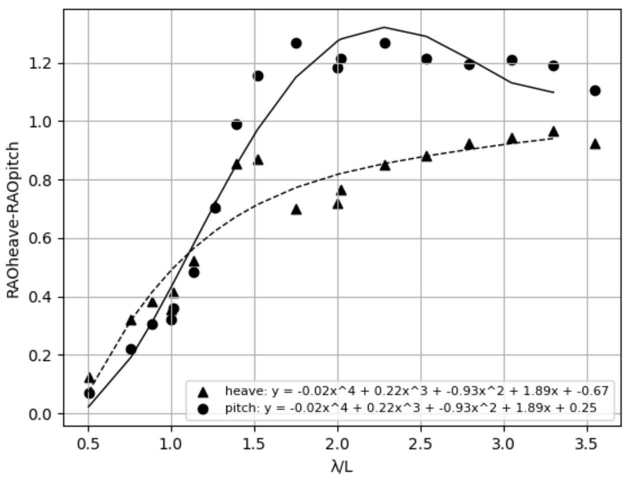


Figure 5: RAO curves for heave-pitch responses (Cdl=3.0, Fn=0.34)

Figure 6: RAO curves for heave-pitch responses (Cdl=3.0, Fn=0.68)

`

RAOs vary for all different types of motion along the 6 degrees of freedom and are strictly dependent on the intensity and physics of the incident disturbance and also to the type, size, and kind of vessel. RAO(ω) is thus the determining factor of the output motion after a disturbance (input) passes through the vessel. The vessel's response is directly proportional to the measure of the RAO.

RAO(ω) is obtained at the preliminary design stage and based on these results, any necessary modifications to the design that would improve the performance of the vessel or prevent cargo from shifting can be identified. Optimize the design of the vessel according to the statistical sea states of the routes the vessel will mainly operate, while ensuring safety operation in different sea states is the target.

By analysing the RAO(ω), the likelihood of certain motions, such as pitch, roll and heave as well as how these motions will affect the ship's performance and stability can be determined. For example, if the RAO indicates that a ship will experience large pitch or roll motions in certain sea states, this could indicate that the ship is at risk of capsizing or that cargo may shift within the vessel. In addition, by analysing the RAO(ω) engineers can identify the frequency range where the ship experiences the largest motions and make design modifications to reduce these motions and increase the comfort of the passengers and crew.

RAOs also provide information about the ship's natural frequency of motion (frequency value of the maximum response) and how the vessel will respond to waves at different frequencies. If the natural frequency of the ship is close to the frequency of the waves it is encountering, it can result in resonance, which can cause large, dangerous motions. By analysing the RAO(ω), these issues can be identified and steps can be taken to mitigate them, such as by modifying the vessel's design or adding enhancing features such as bilge keels, fins or anti-rolling tanks. RAO(ω) can be obtained from existing vessel data, tank tests results, or numerical simulations.

1.6 Seakeeping tests of NTUA Semi-Planing Series

A systematic Series of double-chine, wide-transom hull form with warped planing surface has been developed at the Laboratory for Ship & Marine Hydrodynamics (LSMH) of the National Technical University of Athens (NTUA), during the last decades. The series are appropriate for the design of medium and large monohulls and pleasure crafts intended to operate at high but pre-planning speeds. Pre-planning vessels characterized by a semi-displacement hull, which allows them to operate at high speeds but not able to plane on the water surface. The term "pre-planing" implies that the vessel is in a transitional state between displacement and planing, meaning that it is not quite planing on top of the water but is experiencing some degree of lift and reduced drag due to its hull design. This type of vessels does not posses a displacement type and thus cannot be described by conventional theories such as strip theory. Additionally, these vessels do not meet the criteria for planning hulls and thus cannot be described using methods like Savitsky for predicting the performance and resistance of planing boats at high speeds.

These vessels are often used for military or commercial applications that require high-speed operation over long distances, and they must be designed to provide good seakeeping performance in both calm water and wave conditions. Such hull forms developed by NTUA have been proven to be seaworthy by providing remarkable performance for both calm water and waves. This series is notable as it is the first to present seakeeping results for both regular and random waves. Such hull forms allow vessels to operate efficiently at speeds corresponding to Froude numbers (Fn) greater than 0.4 but less than 1.0.

`

The hull forms of the NTUA series are characterized by two distinct chines that extend forward from the transom for approximately 70% of the hull's length and wrapped planing surface. The bow region of these hull forms is characterized by fine, highly flared lines. Within the series, there are five different hull forms that have a length-to-beam ratio of *(L/B = 4.00, 4.75, 5.50, 6.25 and 7.00)* with the hull form with *L/B = 5.50* being the parent form of the series. These hull forms have a wide transom and a deadrise angle that varies from 10 degrees at the stern to 70 degrees at the bow.

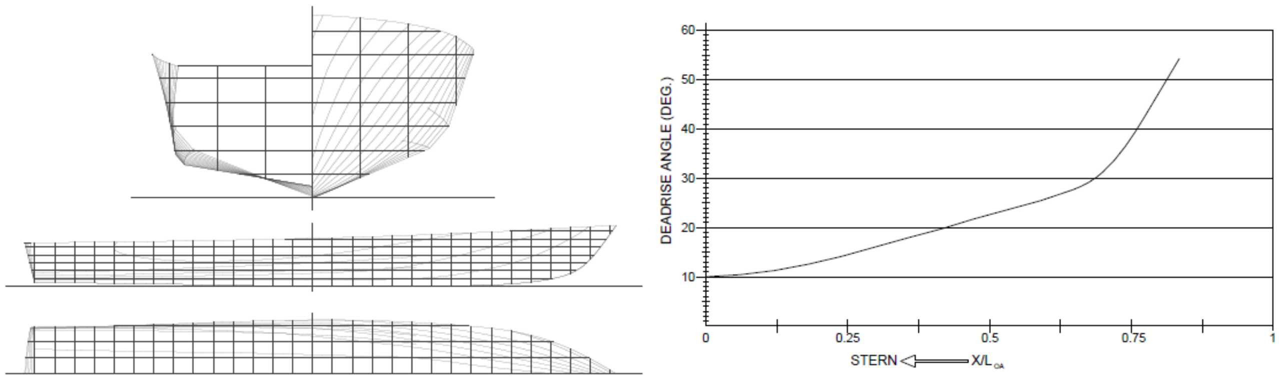


Figure 7: Quoted from G.J. Grigoropoulos & T.A. Loukakis, 2002, Lines plan of the parent hull form and deadrise angle vs x/L_{OA}

NTUA series models tested at three different keel-displacements and two different speeds corresponding to the values of (Cdl = 1.61, 3.00, 4.23) and (Fn = 0.34, 068) respectively. Operating conditions of larger models correspond to lower values of Cdl, while higher values of Cdl are associated with smaller passenger ships and pleasure craft. The model lengths used in the tests were determined based on the suggestion of the 21st ITTC High Speed Marine Vehicles Committee, which recommends using at least two-meter models for such craft. The reason for using a larger model to test a smaller Cdl is because the smaller Cdl values correspond to lighter displacements that cannot be achieved using the standard model lengths. This is because the smaller Cdl values correspond to model displacements that are lower than the combined weight of the bare model and the dynamometer pod. Therefore, a larger model with a higher weight was needed to achieve the required displacement for testing the smaller Cdl values. The scale of the smaller model corresponds 60% of the larger. Depending on the displacement to be tested, either the small or large model was chosen. The characteristics of NTUA Series models are represented below:

`

L/B L_{OA} C_{DL}	4.00-big (163/01) 3.820 m	4.00-small (113/95) 2.292 m	4.75-big (150/99A) 3.820 m	4.75-small (154/99) 2.292 m	5.50-big (118/96) 3.820 m	5.50-small (097/94) 2.292 m	6.25-big (164/01) 4.3417	6.25-small (146/98) 2.605 m	7.00-big (166/01) 4.8617	7.00-small (116/96) 2.917 m
	3.392 1.323		3.415 1.348		3.430 1.365		3.968 1.718		4.457 2.083	
1.00	0.548 39.057		0.571 39.781		0.561 41.443		0.580 63.100		0.390 88.140	
	-0.389 0.080		-0.436 0.077		-0.480 0.077		-0.515 0.088		-0.651 0.097	
	3.445 1.738		3.468 1.696		3.497 1.635		4.029 2.022	2.418 0.728	4.530 2.453	2.718 0.883
1.61	0.762 66.097		0.663 67.118		0.590 69.103		0.593 105.405	0.356 22.767	0.600 150.095	0.360 32.421
	-0.461 0.099		-0.495 0.096		-0.511 0.097		-0.519 0.113	-0.311 0.068	-0.640 0.128	-0.384 0.077
	3.480 2.004		3.505 1.915	2.103 0.689	3.539 1.832	2.123 0.660	4.070 2.265	2.442 0.815		2.731 0.988
2.23	0.788 93.890		0.683 96.021	0.410 20.741	0.597 99.564	0.358 21.506	0.603 150.151	0.362 32.433		0.366 45.436
	-0.494 0.115		-0.511 0.114	-0.307 0.068	-0.490 0.117	-0.294 0.070	-0.500 0.137	-0.300 0.082		-0.369 0.093
	3.514 2.244	2.109 0.808	3.539 2.117	2.124 0.763		2.145 0.728	4.113 2.547	2.468 0.917		2.7830 1.1304
3.00	0.813 130.436	0.488 28.174	133.078	0.416 28.745		0.362 29.615	0.613 208.642	0.368 45.067		0.370 64.618
	-0.509 0.134	-0.305 0.080	-0.508 0.135	-0.301 0.081		-0.297 0.083	-0.472 0.165	-0.283 0.099		-0.345 0.116
		2.110 0.857		2.137 0.812		2.160 0.782		2.482 0.996		2.809 1.245
3.62		0.492 33.993		0.420 35.357		0.366 36.513		0.370 55.489		0.385 80.344
		-0.307 0.087		-0.300 0.086		-0.288 0.095		-0.311 0.114		-0.327 0.135
		2.123 0.903		2.150 0.859		2.175 0.834		2.509 1.074		2.834 1.355
4.23		0.494 40.462		0.422 42.039		0.368 43.530		0.383 66.752		0.394 96.134
		-0.305 0.096		-0.294 0.100		-0.280 0.106		-0.297 0.129		-0.309 0.152

Notes:

1. Each cell of the table contains the following characteristics of the model:

L_{WL} [m]	WS [m^2]
B_{WL} [m]	Δ [Kgr]
LCG [m]	T [m]

2. LCG from amidships, positive forwards.

Table 3: Quoted from G.J. Grigoropoulos & T.A. Loukakis, 2002, Characteristics of NTUA Series models

The model-scale seakeeping tests utilized for this thesis are referred to NTUA Semi-Planing Series models and have been carried out in regular waves. The wooden models were provided with three accelerometers, the first was fitted at the aft peak (FP), the second at the LCG and the last one at fore peak (FP). The models were attached to the carriage via a heave-rod, pitch bearing and resistance measuring assembly and tested in head waves. This assembly allows the models to heave and pitch and the responses were recorded. Additionally, from the total resistance measurements of the models, the added resistance was determined by subtracting the calm water resistance from the total. The data collected from the measuring devices about model responses is used to calculate the RAOs from the known wave spectra.

The wave spectra that have been tested in the NTUA Series follows the Bretschneider model, which is defined by an equation that requires two input parameters: the peak frequency of the waves and their significant wave height.

$$S(\omega) = \frac{\frac{5}{16}\left(\frac{\omega_p}{\omega}\right)\left(\overline{H}_{\frac{1}{3}}\right)^2}{\omega_p} \exp\left(-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4\right), \qquad \text{Bretschneider spectra}$$

To ensure linearity exists, wave heights must be kept low when testing vessels in regular waves. If the wave heights are too high, the non-linear effects of the waves can disrupt the testing process.

`

In contrast, random waves are used to test vessels at higher wave heights, where non-linear effects become more significant. While regular waves are generally sufficient for testing conventional vessels, random waves may be necessary for vessels different hull forms or operating conditions. Understanding a vessel's behaviour in both regular and random wave conditions is essential for ensuring safe and efficient operation in a variety of sea states.

The notations used in the present study are summarized in the following table:

NOMENCLATURE	
L	*Length overall ****
B	*Maximum breadth at upper chine*
L_{WL}	*Waterline length at rest*
B_{WL}	*Waterline breadth at rest*
LCG	*Longitudinal Centre of gravity*
T	*Wave period*
Fn	*Froude number* $(= \frac{V}{\sqrt{gLwl}})$
V	*Ship speed*
f	*Wave frequency*
λ	*Wave length* $(= V_w T)$
V_w	*Wave speed* $(= \frac{\lambda}{T})$
k	*Wavenumber* $(= \frac{2\pi}{\lambda})$
ξ_3	*Heave*
ξ_5	*Pitch*
a_s	*Vertical stern acceleration (at AP)*
a_m	*Vertical mid acceleration (at LCG)*
a_b	*Vertical bow acceleration (at FP)*
g	*Acceleration of gravity*
p	*Water density*
R_T	*Total resistance*
R_{AR}	*Added resistance*
R_{CW}	*Calm water resistance*
C_{DL}	*displacement-length ratio at rest* $\left(= \frac{displ}{(0.1L_{WL})^3}\right)$
A	*Wave amplitude*
$\bar{A}$	*Mean wave amplitude* $\bar{A} = \frac{\sum_{i=1}^{i=N} A_i}{N}$
A_{RMS}	*Root Mean Squared value of wave amplitudes* $(= \sqrt{\overline{A^2}}\ )$
WS	*Wetted surface*
RAO_{ξ_3}	*Response Amplitude Operator for heave* $(= \frac{\xi_3}{A})$*
RAO_{ξ_5}	*Response Amplitude Operator for pitch* $(= \frac{\xi_3}{kA})$*
RAO_{a_s}	*Response Amplitude Operator for* a_s $(= \frac{a_s}{g}\frac{L_{WL}}{A})$*
RAO_{a_m}	*Response Amplitude Operator for* a_m $(= \frac{a_m}{g}\frac{L_{WL}}{A})$*
RAO_{a_b}	*Response Amplitude Operator for* a_b $(= \frac{a_b}{g}\frac{L_{WL}}{A})$*
RAO_{AR}	*Response Amplitude Operator for* R_{AR} $(= \frac{R_{AR}}{pgB_{WL}^2}\frac{L_{WL}}{A^2})$**

Table 4: Principal Notation

`

**Non dimensional ratios recommended by the Seakeeping Committee of the ITTC (1981)*

***Non-dimensional quantity recommended by the ITTC (1984)*

**** based on the suggestion of the 21st ITTC High Speed Marine Vehicles Committee*

2. ARTIFICIAL NEURAL NETWORKS (ANN) BACKGROUND

A neural network is a type of machine learning model that is designed to recognize patterns in data. It is inspired by the structure and function of the human brain, with interconnected nodes or "neurons" that work together to process and analyse information. Neural networks are used in a variety of applications, including image and speech recognition and predictive modelling.

2.1 History of ANN

Artificial Neural Networks (ANNs) have a rich history rooted in human efforts to design intelligent machines. The development of electronic computers drew the attention of some scientists towards the inner workings of the human brain. In the early 1940' Warren McCulloch, a neurophysiologist, teamed up with logician Walter Pitts to create a model of how human brain work, as can be found in the publication titled "A logical calculus of the ideas inherent in nervous activity" in the Bulletin of Mathematical Biophysics. They presented a model for a single neuron that formed the building block for the computational equivalent of a biological neural network in the brain. Just like brain neurons receive electrochemical signals, McCulloch and Pitts' neuron model received inputs from other neuron or external sources, and if these input signals were strong enough, passed them on to other neurons. A figure quoted from the publication of Python Machine Learning (Raschka,2015) is used to illustrate this logic.

`

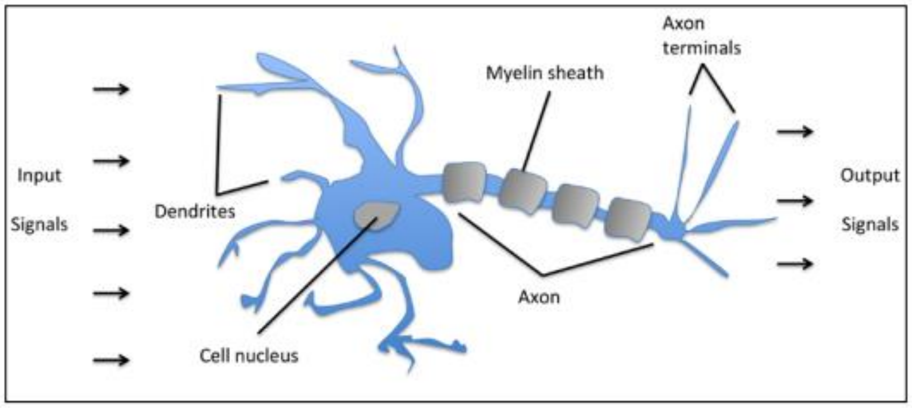


Figure 8: Structure modelling of the human brain

The neuron (cell nucleus) comprises the basic structural component of the human brain. Briefly, a neuron consists of a sensory section where the input signals of previous neurons are collected (known as dendrites), a section of electrochemistry processing and a section through whose output signal is propagated. The dendrites make up the sensory zone of the neuron and collect its inputs from others neurons. The synapses constitute the structural and functional units which are responsible for the interaction between neurons. With the aid of synapse, a neuron can either encourage or discourage the activity of another neuron by increasing or decreasing its electrical activity, respectively. If the accumulated signal exceeds a certain threshold, an output signal is generated that will be passed on by the axon. This is the basic mechanism through which neurons communicate with one another and form networks in the brain, allowing us to process and respond to information. The axon is the line transmission of a neuron and distributes its output signals to other neurons.

2.2 Single Layer Perceptron

Their work inspired many other researchers to contribute to the development of ANNs. It was only a decade later that Frank Rosenblatt achieved the first implementation by building the first perceptron machine and created an algorithm that could learn the weights in order to generate an output. Frank Rosenblatt, at the Cornell Aeronautical Laboratory in 1958, created the first successful neurocomputer, which was capable of seeing and classifying linearly separable objects. It was the algorithm that simulates the operation of a neuron in Rosenblatt's perceptron that laid the foundation for the modern ANNs. Rosenblatt proposed an algorithm that would automatically learn the optimal weights that then multiplied with the inputs in order to be determined whether a neuron fires or not.

`

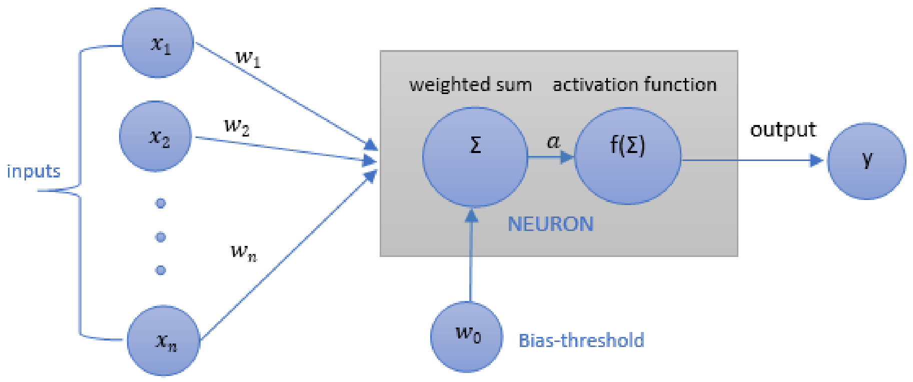


Figure 9: Single layer-perceptron model, invented by McCulloch and Pitts, implemented by Rosenblatt

The computational neuron of figure 7 models the electrical impulses that a biological neuron senses in its Dendrites as scalar numbers. The cell nucleus and axon, which combine and transmit the input signals to the outputs, are represented as a linear combination of the scalar inputs, each receiving a unique weight. The sum of the weighted inputs then passes through an activation function, which simulates the firing or non-firing of the axon terminals.

The term Σ which is a weighted sum of inputs including the bias, is expressed by $\sum_{i=1}^{n} w_i x_i + w_0$ and it is known as activation. The term f( ), refers to the activation function which is a mathematical operation applied to the output of each node (neuron) in the network to determine whether the neuron should be activated or not. The activation of a neuron is strongly depended on bias term.

In this Rosenblatt perceptron, the neuron is treated as a binary threshold device since the algorithm was intended for binary classification using linear predictor activation function combining a set of weights with the inputs. If the weighted sum of inputs to the neuron plus the bias(threshold) is a positive value, the neuron outputs 1, otherwise it outputs 0. This model has been a benchmark for the development of todays' ANN models and is represented below:

$$y(x,w) = \begin{cases} 1, & \sum_{i=1}^{n} w_i x_i + w_0 > 0 \\ 0, & otherwise \end{cases} \qquad (2.2.1)$$

In today's context of neural networks, a perceptron is an artificial neuron using the binary step function as activation function.

The early success of the perceptron algorithm led to high expectations for its capabilities, but it was soon discovered that it had limitations in its ability to recognize many classes of patterns. In 1969, a famous book entitled "Perceptrons" by Marvin Minsky and Seymour Papert showed that it was impossible for these classes of network to learn an XOR(exclusive or) function. XOR is a conditional-logical operation which result is true only if its inputs differ (one is true and the other false). It computes the same result as the inequality operator != , but for bool operands. This realization caused a slowdown in research and development in the field of neural networks.

`

Some years later, it was realized that a feedforward neural network with multiple layers, known as a multilayer perceptron, had superior processing power compared to a single-layer perceptron and can overcome this problem.

2.3 Multi-layer Perceptron

A multi-layer network has additional "hidden" layers between the input and output layers and in order to assigned the notation multi-layer should have at least one hidden layer. The first layer of the network is the input layer, which is constructed according to the data input form. This layer does not include neurons with processing ability. It only forwards the inputs to other neurons (n_j). The last layer of the network is the output layer, includes neurons, and generates the final outcome of the model. The hidden layers are placed between input and output layers and as their name indicate, are not visible to the user, often called as black box. Can perform different transformations on the inputs, and transformed signals flow as inputs to the output layer.

The multi-layer network can be feed-forward or recurrent, depending on the arrangement of its neural synapses. In feed-forward networks, data flows in a single direction from the input layer to the output layer, with neurons arranged in layers and connections only allowed between different layers towards the output, without looping back. Recurrent networks, on the other hand, allow for feedback connections linking the output of one neuron to the input of another neuron, either in the same layer or a previous layer.

Each connection in a neuron is associated with a weight, representing the information and the magnitude of it used in processing, and each neuron has an activation unit based on its inputs. The activation of each neuron is determined by an equation incorporating the weighted sum of inputs and bias, with a transfer function applied to each layer connections.
In mathematical terms the neuron j is described by writing the following pair of equations (Bishop,2006):

$$a_j = \sum_{i=1}^{I} w_{ji}^{(1)} x_i + w_{j0}^{(1)} = \sum_{i=0}^{I} w_{ji}^{(1)} x_i \ , \qquad \mathrm{j} = 1,2,\ldots,\mathrm{J} \qquad (2.3.1)$$

The bias parameters have been absorbed into the set of weight parameters by defining an additional input variable $x_0 = 1$.

The term j = 1,2, …, J corresponds to the number of neurons/parallel perceptrons and the superscript (1) is referred to the first layer. The $w_{ji}^{(1)}$ is the matrix of weights controlling the connections between the consecutive layers i and j. The resulting values $a_j(activations)$ then undergo the necessary transformations with the aid of activation functions f( ) which finally produce the output of the neuron j:

$$n_j = f\left(a_j \right) \qquad (2.3.2)$$

`

Consecutively, these values n_j are passed to the next layer as inputs and the same computations are performed for output layer, and the activation for each neuron is mathematically expressed as per below:

$$a_k = \sum_{j=1}^{J} w_{kj}^{(2)} x_j + w_{k0}^{(2)} = \sum_{j=0}^{J} w_{kj}^{(2)} x_j \; , \qquad k = 1,2, \dots, K \qquad (2.3.3)$$

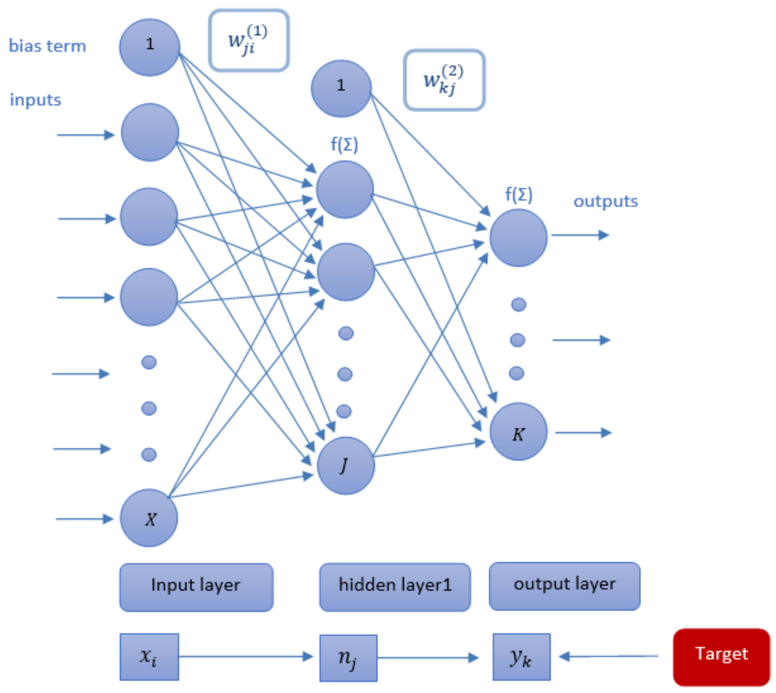
Where k =1,2, ..., K corresponds to the number of outputs. The $w_{kj}^{(2)}$ is the matrix of weights controlling the connections between the consecutive the layers j and k. The above activations a_k are referred to the second layer of the network, and the $w_{k0}^{(2)}$ are bias parameters. Finally, the output activations are transformed using an appropriate activation function to produce the outputs y_k .

By combining the above equations, the overall network function can be obtained:

$$y_k(x,w) = f\left(\sum_{j=0}^{J} w_{kj}^{(2)} f\left(\sum_{i=0}^{I} w_{ji}^{(1)} x_i\right)\right) = f\left(\sum_{j=0}^{J} w_{kj}^{(2)} f(a_j)\right) \qquad (2.3.4)$$

It can be easily understood that if a neural network has n_j nodes in layer j and n_{j+1} nodes in layer j+1, the w_{j+1j}, corresponds to the matrix of weights controlling the connections of j to j+1 layer. So that w_{j+1j}, will be dimensioned as (n_j+1, n_{j+1}) and the +1 corresponds to the bias term.

A simple multilayer perceptron with one hidden layer is presented below:



Network diagram of a multilayer feed forward perceptron with 2 layers. The input, hidden, and output variables are represented by nodes, and the weight parameters are represented by links between the nodes, in which the bias parameters are denoted by links coming from additional input and hidden variables. Arrows denote the direction of information flow through the network during forward propagation. Finally, the output layer result is compared with the target and if error is minimum the process stops, otherwise the error is backpropagating.

Figure 10: Network diagram for 2-layer feedforward perceptron.

`

2.4 Activation Functions

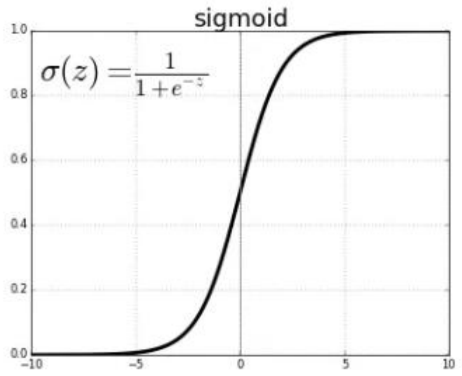
There are many activation functions that can be used in neural networks. Before we present the most popular activation function and its usage, the perspective of finding the ideal activation function should be well clarified. The activation function used in artificial neurons should possess two key characteristics: non-linearity and differentiability.

Firstly, the activation function should be able to introduce non-linear relationships into the neural network, particularly in the neurons of hidden layers. This is because most real-world scenarios involve complex, non-linear relationships, rather than simple linear ones.

Secondly, the activation function should be differentiable, meaning that its derivative can be calculated. This is because during the training phase of a neural network, the backpropagation algorithm uses the derivative of the activation function to adjust the weights of the neurons in hidden layers in a way that the error in the next epoch to be reduced. The term epoch is used to determine training cycles-iterations.

In summary, the activation function should provide non-linearity and differentiability in order to effectively train a neural network and capture complex relationships in real-world scenarios. Some of the most commonly used families of activation functions are presented below:

- Sigmoid: A sigmoid activation function maps any input to the range of 0 and 1, producing an output that can be interpreted as a probability. Sigmoid function commonly used in the output layer and may be avoided in hidden layers since it suffers from vanishing gradient problem.



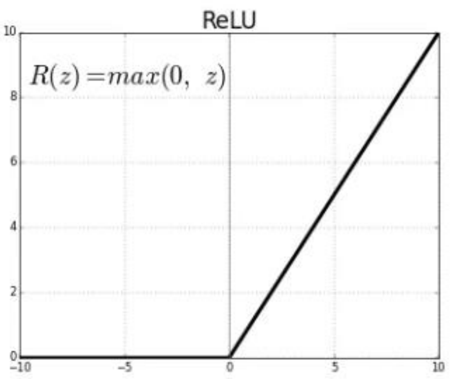
$$z = \sum_{i=1}^{n} w_i x_i + w_0$$

$$f(z) = \frac{1}{1+e^{-z}}$$

Figure 11: The output of the sigmoid function is always between 0 and 1, which makes it particularly useful for classification tasks where the output represents the probability of a given class.

- ReLU (Rectified Linear Unit): The ReLU activation function sets any negative input to 0 and retains positive inputs unchanged. This function has become widely popular in deep learning due to its computational efficiency and ability to avoid the vanishing gradient problem. Those facts make ReLU appropriate for hidden units.

`

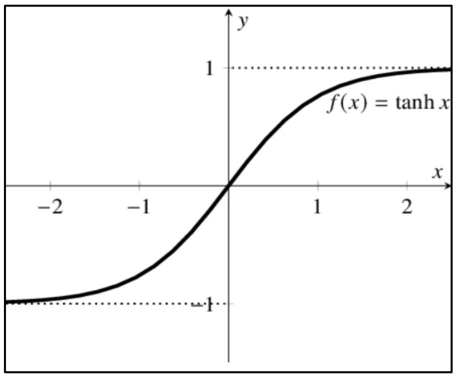


$$z = \sum_{i=1}^{n} w_i x_i + w_0$$

$$f(z) = \max(0, z) = \begin{cases} z, & z > 0 \\ 0, & otherwise \end{cases}$$

Figure 12: The non-linear function ReLU maps any input value less than or equal to zero, and any input value greater than zero to the input value itself

- Tanh (Hyperbolic Tangent Activation Function): The tanh is an activation function commonly used in neural networks. The range of the tanh function is between -1 and 1, which means that it can produce negative or positive values, with the magnitude of the output decreasing as the input moves farther away from zero.

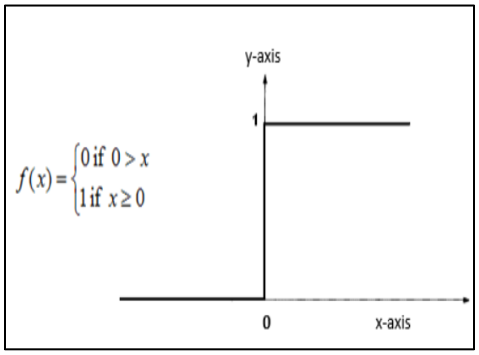


$$z = \sum_{i=1}^{n} w_i x_i + w_0$$

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

*Figure 13: Tanh is a non-linear function that **maps** the input values to a range between -1 and 1, making it useful for normalizing data.*

- BSU (Binary Step Unit): The BSU activation function depends on the threshold value that decides if the neuron is to be activated or not and it is commonly used for classification problems.



$$z = \sum_{i=1}^{n} w_i x_i + w_0$$

$$f(z) = \begin{cases} 1, & z \geq 0 \\ 0, & otherwise \end{cases}$$

Figure 14: The binary step function is a simple activation function that outputs a binary value of either 0 or 1 based on whether the input is less than or greater than a specified threshold

It is important to experiment with different techniques and to understand the trade-offs associated with each approach in order to determine the best strategy for handling the limitations of activation functions.

`

It's important to choose the right activation function for the problem at hand and to experiment with different functions to determine which one works best.

Although activation functions are a fundamental component of Artificial Neural Networks (ANNs), they are not without their limitations. Fortunately, there are several techniques that can be used to address these limitations, the most common of them are summarized below:

1. Weight Initialization: The choice of weight initialization can have a significant impact on the convergence and stability of the network. It is recommended to use weight initializations that are well-suited to the activation functions used in the network.

2. Batch Normalization: Batch normalization is a technique that normalizes the activations of a layer in the network during training, which can help to reduce the impact of vanishing gradients and exploding gradients.

3. Non-Linear Activation Functions: Non-linear activation functions, such as ReLU or sigmoid, can also help to improve the performance of ANNs. These functions allow the model to learn more complex and non-linear relationships between the input and output variables.

4. Early Stopping: Overfitting can be addressed by using techniques such as early stopping, which involves monitoring the performance of the network on a validation set and stopping training when the performance on the validation set starts to degrade.

5. Regularization: Regularization is a technique that involves adding a penalty term to the loss function to discourage overfitting. This penalty term encourages the model to have smaller weights, which in turn can help to prevent it from overfitting the training data.

2.5 ANN Layout

The appropriate number of nodes in each layer of a feedforward neural network can be determined through experimentation and trial-and-error. There is no certain approach determining the number of units in each layer, as it depends on factors such as the complexity of the problem, the size of the input data, and the desired level of accuracy. The layout of the neural network with respect to the number of nodes in each layer can be described as follows:

$$[X - H_1 - \cdots - H_N - Y]$$

- The number of nodes in the input layer [X] is determined by the dimension of input features.
- The number of nodes in the output layer [Y] is determined by the dimension of target features.

`

- The number of hidden layers ($H_1, \ldots, H_N$) could be adjusted from the developer, usually 1-2 hidden layers are sufficient for small dataset problems.
- The number of nodes in hidden layers is adjusted at the needs of the developer. In general, multiple hidden layers can learn more complex representations of the input data, but it may also require more training data and often cause overfitting. It's important, when building Artificial Neural Networks to find the balance between the bias(error) and variance which represent the variety of the model predictions. For example, if a model performs well in training test and poor in testing set, that is an effect of overfitting and actions to reduce the complexity of the model should be implemented. On the other hand, if a model performs quite good in the training set and also have similar performance in the test set, this might be considered as closer to the optimum model. Note that reaching the optimum model is a procedure that requires a lot of tunning.

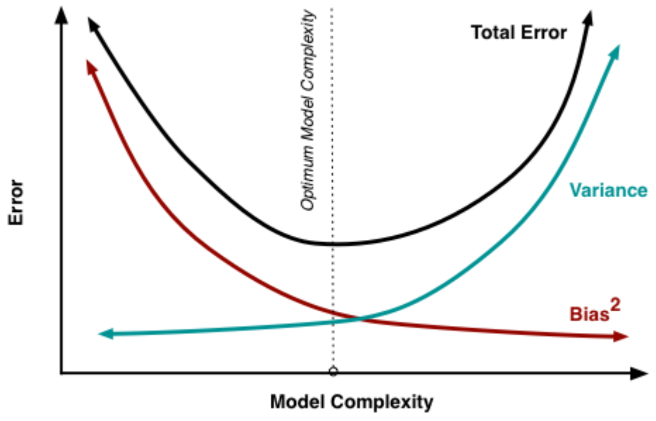


Figure 15: Error and Variance trade off

2.6 ANN Training

The goal of a training process of an ANN refers to the estimation of the adaptive parameters that lead to minimum error of the model. The most crucial parameter in the training of ANN is the selection of the training algorithm which will be responsible for adjusting the weight connections. One of the key ingredients of supervised machine learning algorithms is to define an objective function that is to be optimized during the learning process. This objective function is often a cost function that we want to minimize. The training algorithm for a neural network teaches the network to identify the impact of each input signal on the output of a processing unit and to map a contribution of a certain input to the desired output.

First, the input signal is passed through all the neurons and layers, and the network generates an output based on the input and the defined weights. The network's response is then evaluated using a cost function, usually mean square error, to measure the deviation between the actual output and the target-desired output. In the second step, the deviation-error is then used to adjust the weights in the network,

`

so that the cost function is minimized. This process is repeated for some iterations n until the network reaches the desired performance level. The most commonly cost function used to represent the learning performance is mean square error and it is described as:

$$mean\ square\ error\ E(w) = \frac{1}{z}\sum_{i=1}^{z}\frac{1}{2}e_k^2 = \frac{1}{z}\sum_{i=1}^{z}\frac{1}{2}(d_k - y_k(x,w))^2 \qquad (2.6.1)$$

The term 1/2 is just added for convenience as it will make it easier to derive the gradient, as it can be seen in the following paragraphs. Supposing the number of target values to be k = 1,2, …, z the deviation-error can be calculated for the output neuron k at iteration n:

$$e_k(n) = d_k(n) - y_k(x,w)(n) \qquad (2.6.2)$$

Where d_k denotes the desired-target-output and y_k the actual output.
An alternative regression algorithm, ridge regression adds a penalty to the least squares objective function in order to shrink the coefficients towards zero and mathematically expressed as:

$$\text{minimize}: E(w) = \sum_{i=1}^{z}\frac{1}{2}e(w)^2 = \sum_{i=1}^{z}\frac{1}{2}(y_z(x,w) - d_z)^2 + \lambda\Sigma \mathrm{w}^2 \qquad (2.6.2)'$$

This penalty term denoted as λ is proportional to the sum of the squares of the model coefficients, and it helps to shrink the coefficients towards zero. With w we denote the weight vector. The value of the regularization parameter λ controls the trade-off between the fit of the model to the training data and the size of the coefficients. Larger values of λ lead to smaller coefficients and a simpler model, but may result in a poorer fit to the training data. On the other hand, smaller values of λ lead to larger coefficients and a more complex model, which may result in overfitting. The optimal value of λ can be determined using cross-validation.

This technique can be helpful when dealing with high-dimensional data or complex models as it can help to reduce overfitting and improve the generalization performance of the model. Ridge regression is often used when there are a large number of predictors that are potentially correlated with each other, as it can help to stabilize the estimates of the coefficients.

The calculation of weight adjustments in a neural network is heavily dependent on the computation of the error signal $e_k(n)$. When the output node is involved, the error signal calculation is relatively straightforward since the target response is already provided. However, for neurons located in the hidden layers, the problem becomes more complex, as they are not directly linked to a desired response and must share responsibility for any error made at the final output. Therefore, the challenge is to determine how to penalize or reward these hidden neurons based on their contribution to the final error. To solve this problem, the error signal needs to be determined recursively by considering the error signal of all the neurons that the hidden neuron is directly connected to. This problem is commonly referred to as the

`

credit-assignment problem, which is a crucial and challenging aspect of neural network training described in (Haykin, 1999).

The objective of a learning process is to minimize the cost function by adjusting weights and bias of the network. The adjustments of weights are made in accordance with the respective errors calculated for each pattern. To perform the minimization, we make use of the back propagation algorithm. At each iteration any backpropagation algorithm applies a correction $dw_{kj}(n)$ to the weights, which is proportional to the partial derivative of a cost function as follow:

$$\nabla E(w) = \frac{dE(n)}{dw_{kj}(n)} = 0 \qquad (2.6.3)$$

By considering the definition of the derivative of a function we get the physical meaning of dE/dw_{kj},which is how much a small change in the value of w_{kj} will affect the value of the error. But the error is a function of the output of the network itself, and the output of the network is a function of the inputs and the weights.

According to the chain rule of calculus, this gradient can be expressed as:

$$\frac{dE(n)}{dw_{kj}(n)} = \frac{dE(n)}{de_k(n)}\frac{de_k(n)}{dy_k(n)}\frac{dy_k(n)}{dv_k(n)}\frac{dv_k(n)}{dw_{kj}(n)} \qquad (2.6.4)$$

The partial derivative of $\frac{dE(n)}{dw_{kj}(n)}$ is determining the direction of search in weight space for the synaptic weight w_{kj}.

In order to compute the associated derivatives, the formulation of neuron is to taken into account first:

$$v_k(n) = \sum_{i=0}^{m} w_{kj}(n)x_j(n) \qquad (2.6.5)$$

$$y_k(n) = \varphi\big(v_k(n)\big) \qquad (2.6.6)$$

By making use of the equations 2.6.1, 2.6.2, 2.6.5 & 2.6.6 the chain rule of calculus yields to:

$$\frac{dE(n)}{dw_{kj}(n)} = -e_k(n)\varphi_k'\big(v_k(n)\big)x_j(n) \qquad (2.6.7)$$

The correction $dw_{kj}(n)$ is defined by the delta rule:

$$dw_{kj}(n) = -\eta\frac{dE(n)}{dw_{kj}(n)} = n\delta_k(n)x_j(n) \qquad (2.6.8)$$

We can describe the principle behind gradient descent as climbing down a hill until a local or global cost minimum is reached. In each iteration, we take a step away from the gradient where the step size is determined by the value of the learning rate as well as the slope of the gradient:

`

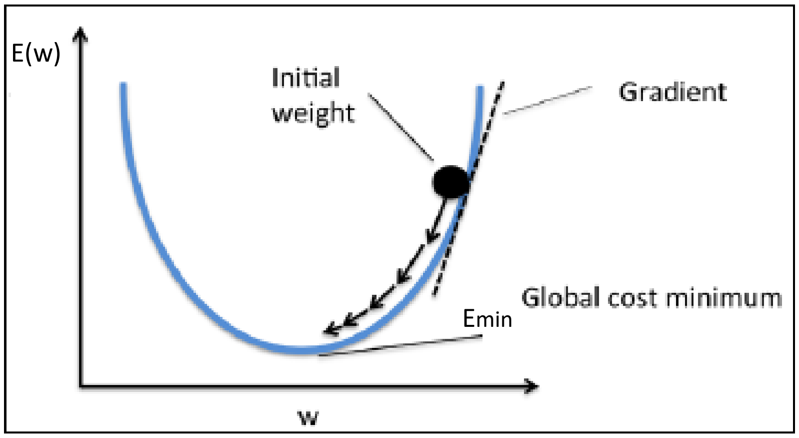


Figure 16: Gradient descent backpropagation, Source: Raschka,2015

Where n is the learning rate of the backpropagation algorithm and the term x_j refers to the input signal of neuron j. The use of minus accounts for gradient descent in weight space, seeking a direction for weight change that reduces the $E(w)$ value at iteration. The term $\delta_k(n)$ correspond to the local gradient for output neuron j is equal to the product of the error signal and the derivative $\varphi_k'(v_k(n))$ of the associated activation function.

$$\delta_k(n) = -\frac{dE(n)}{dv_k(n)} = -\frac{dE(n)}{de_k(n)}\frac{de_k(n)}{dy_k(n)}\frac{dy_k(n)}{dv_k(n)} = -e_k(n)\varphi_k'(v_k(n)) \quad (2.6.9)$$

The process of error correction learning is illustrated below:

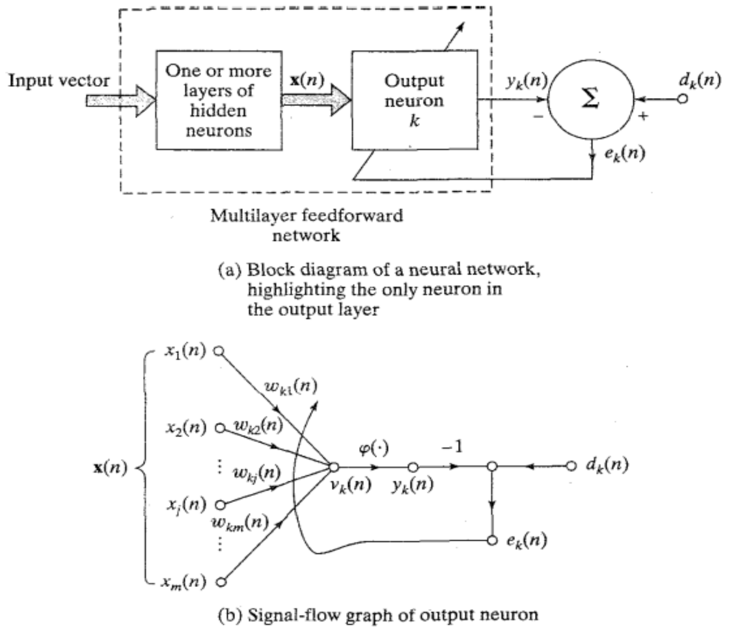

Input vector
One or more layers of hidden neurons
x(n)
Output neuron k
$y_k(n)$
Σ
$d_k(n)$
−
+
$e_k(n)$
Multilayer feedforward network
(a) Block diagram of a neural network, highlighting the only neuron in the output layer
$x_1(n)$
$w_{k1}(n)$
$x_2(n)$
$w_{k2}(n)$
$\varphi(\cdot)$
−1
x(n)
$w_{kj}(n)$
$v_k(n)$
$y_k(n)$
$d_k(n)$
$x_j(n)$
$w_{km}(n)$
$e_k(n)$
$x_m(n)$
(b) Signal-flow graph of output neuron

Figure 17: Quoted from Haykin,1999, Learning process of ANN with single output neuron k

`

2.7 Convergence of optimization algorithms

The learning rate, a crucial parameter in algorithmic optimization, determines the rate at which the system is driven in the direction of error reduction. The dilemma between convergence and speed is directly related to the value chosen for learning rate. Large values result in rapid error reduction with big steps, but can compromise the discernment capability of the algorithm: Considering a narrow local error minima on the surface, algorithms with large learning rates become trapped in oscillations between this part of the surface without a way to escape. Below figure demonstrates the processing progress of a gradient descent algorithm that utilized a high learning rate. Beginning at point A, the system is propelled towards point B based on the slope of the error at point A and the size of the step determined by the learning rate. The algorithm follows a similar path from point B to point Γ, where the sign of the error surface slope changes, and subsequently, the system is driven to the opposite direction, back towards point B. This initiates a sequence of oscillations between points B and Γ.

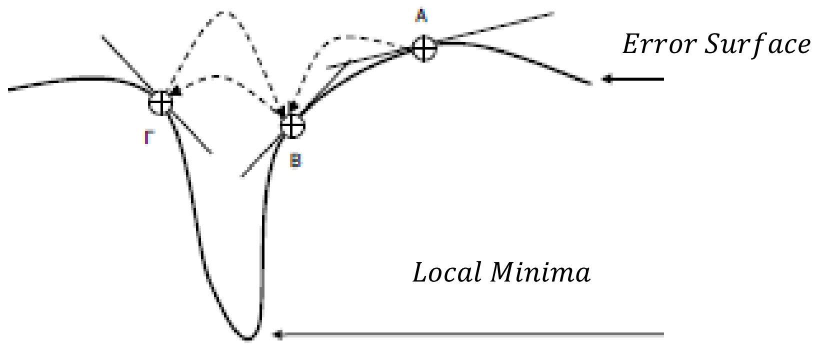


Figure 18: Quoted from S. Raptis Computational intelligence, gradient descent algorithm trapped in a sequence of oscillations between B & Γ points.

Although the error backpropagation algorithm is the basic tool, it may result in some problems, many of which have been addressed either by additions to the basic logic of the algorithm, or with modified versions of it.

Some practices that are often followed in conjunction with the back propagation to improve its performance are listed below:

- Normalization of training data: It's effective for best training to normalize features that use different scales and ranges. Features are multiplied by the model weights and as a result the scale of the outputs and the scale of gradients are affected by the scale of inputs.
- Variable learning rate: The convergence/speed dilemma concerning the choice of learning rate value, often treated with the use of a variable rate. During the training stage, learning rate initially takes large values so that the error is quickly driven to low error areas. Then the rate is reduced to improve the discriminative ability of the algorithm.

`

- Momentum parameter: Adding a momentum parameter to the logic of the algorithm can be quite an important aid in dealing with situations of "trapping" in local minima.

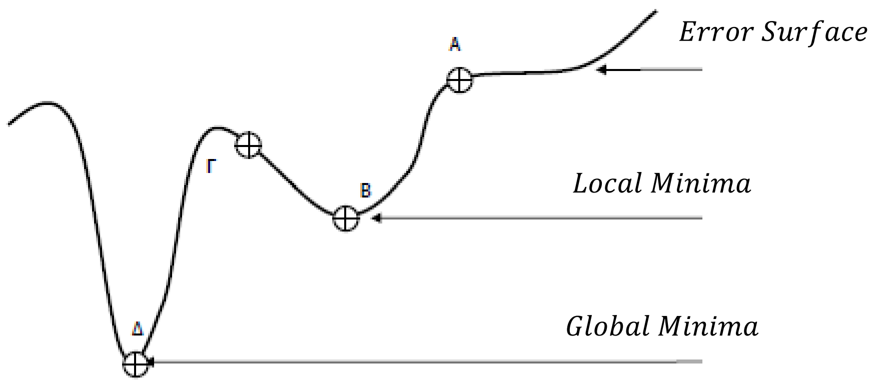


Figure 19: Quoted from S. Raptis Computational intelligence, Multiple error local minima and gradient descend algorithm detects global minima with the aid of momentum parameter

Considering that the training algorithm starts from point A on the surface fault, moving according to its slope will head towards point C. But getting there, he will be forced to move in the opposite direction since apparently the error increases to the left. It will thus end up at point B, which is a local minimum. The momentum parameter will contribute for the algorithm to maintain its course in the direction had originally chosen, allowing to avoid point B and identify global minimum D. If optimization algorithm is moving in a general direction the momentum causes it to resist changes in directions which results in dampening of oscillations for high curvature surfaces.

3. MODELLING OF TANK TEST RESULTS

The objective of this thesis is to develop a neural network model capable of predicting the seakeeping response of a vessel based on inputs of sea state and vessel parameters. However, effective training of a neural network necessitates not only an adequate quantity of input data but also the selection of appropriate input features is essential for the accurate development of the model. Data pre-processing is an essential step in the development of neural network models as it can significantly impact the performance and accuracy of the model. The process involves transforming and cleaning the raw data, removing any missing or irrelevant features, and standardizing the data to improve its quality and consistency. Effective data pre-processing can help reduce overfitting, improve generalization capability, and enhance the efficiency of the model.

`

3.1 Data Overview

The dataset consists of 15 seakeeping tests conducted for each Froude number of 0.34 and 0.68. Each experiment was characterized by three main parameters: Froude number (Fn), length-to-beam ratio (L/B), and non-dimensional ship load factor (Cdl). Five different hull models with L/B ratios of 4.00, 4.75, 5.50, 6.25, and 7.00 were tested under three loading conditions (Cdl=1.61, 3.00, and 4.23). The dataset comprises 30 seakeeping experiments, with 330 samples for Fn=0.34 and 304 for Fn=0.68. It is worth noting that in the absence of data for at least one Fn, discourages the potential merging of the data as it is well known that for a valid curve to be derived 3 points are required as minimum. In this respect, the datasets for the 2 different Fn are examined separately. The seakeeping tests involve various parameters that are measured and recorded during the experiments. These parameters are depicted below:

Variables	Symbol
Waterline length at rest	L_{WL}
Waterline breadth	B_{WL}
Vessel speed	V_S
Froude number	Fn
Displacement-length ratio	C_{DL}
Wave frequency	f
Wave length	λ
Heave	ξ_3
Pitch	ξ_5
Bow acceleration	a_b
Mid acceleration	a_m
Stern acceleration	a_s
Wave Amplitude (rms)	A_{RMS}
Total resistance (mean)	R_T
Calm water resistance	R_{CW}
Wave length	λ
RAO Heave	RAO_{ξ_3}
RAO Pitch	RAO_{ξ_5}
RAO Bow acceleration	RAO_{a_b}
RAO mid acceleration	RAO_{a_m}
RAO stern acceleration	RAO_{a_s}
RAO added resistance	RAO_{AR}

Table 5: Variables of model test results

`

The variables highlighted in red denote the target variables, which are the variables that we seek to predict in this thesis based on the input data (rest variables). The target variables represent the desired outcome or results of the experiments, and should be identified prior to the modelling stage. By analysing the input data and identifying patterns and relationships, the ANN model can make predictions about the target variables, allowing for informed decision-making and improved understanding of the system being studied. However, effective data pre-processing is essential for a good performing network.

3.2 Data manipulation

A manual fixed file which contains the paths for all experimental excel spreadsheets (15total) has been created. The goal is to produce a dictionary object for each Froude number which can be able to hold separately the 15 original experiments and 15 repeated experiments, contained in 30 dataframe objects. Even in case there is no repeated experiment, and make modifications and manipulations to each of them easily and practically. Notting also the original excels files contain empty rows or unwanted comments, embedded Pandas functions were utilized for handling the messy datatypes and perform the essential data cleaning of each experiment. The repeated experiments were carried out to verify or drop ambiguous results. The term ambiguous refers to any discrepancies with the neighbouring points or altering in the trend of the best fit curves. Only if the results of the repeated tests were consistent the point is to be added to the graphical representation of the data. Also, some new points were identified in the repeated dataset, with the term new to correspond to wave frequency parameter.

In order to be determined whether this point is consistent or not, the error of duplicated wave frequency values between the repeated and actual measurements for the target variables is calculated and if it's found to be greater than the threshold, this data point it's dropped out of the working dataset. To underline here, that the target variables refer to RAO heave, RAO pitch, RAO bow acceleration, RAO mid acceleration, RAO stern acceleration and RAO added resistance. The error threshold value was set to be 15% in order the picks to be enclosed.

As mentioned above in some of the repeatability tests, new points (unique wave frequency) have been tested. For determining whether to include these results in the working dataset, best fit curves were applied to the target features and if the error between the results from best fit curves is found to be small than the threshold, this data point is included in the working dataset, otherwise it's excluded. The dataset was subjected to various curve-fitting techniques, including linear interpolation, polynomial regression with a variety of orders as well as piecewise polynomial interpolation techniques for a variety of orders to identify the best fit curves for the target features. It's worth to mention that if no repeated experiment is identified, the dictionary automatically creates an empty dataframe for this specific repeated experiment identified by its unique key.

The aforementioned actions were performed this the aid of PyCharm. PyCharm is an Integrated Development Environment (IDE) used for developing applications in Python. It provides a wide range of features that make the development process easier, faster and more efficient. A brief description of the objects, libraries and functions utilized for this thesis are enclosed in the Appendix.

`

3.3 Polynomial regression

Polynomial regression is a type of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an nth degree polynomial. The polynomial function is defined as:

$$y = w_o + w_1 x + w_2 x^2 + \cdots + w_n x^n = \sum_{i=0}^{n} w_i x^n$$

where y is the dependent variable, x is the independent variable, $w_0, w_1, w_2, \ldots, w_n$ are the coefficients of the polynomial function, and n is the degree of the polynomial.

The procedure for polynomial regression involves fitting this polynomial equation to the data points using a suitable regression algorithm, such as least squares regression method, or ridge regression. The algorithm finds the values of the coefficients that minimize the sum of the squared differences between the predicted values and the actual values and it's represented as:

$$\text{minimize:} E(w) = \sum_{i=1}^{n} \frac{1}{2} e(w)^2 = \sum_{i=1}^{n} \frac{1}{2} (y_i(x, w) - d_i)^2$$

Where y is the actual values also referred as the dependent variable, w coefficient vector, x is the independent variable matrix, and *E(w)* is the objective function(error) to be minimized. The term d_i denotes the actual values, while ½ is added for our convenience.

Polynomial regression can be used to model non-linear relationships between the independent and dependent variables. However, it is important to choose the right degree of polynomial function to balance between overfitting and underfitting the specific dataset. Overfitting occurs when the model is too complex and fits the noise in the data, while underfitting occurs when the model is too simple and fails to capture the underlying relationship in the data. The degree of polynomial that better describes the dataset is highly dependent on the subject dataset. Second, third, fourth and higher order polynomial models were tested in order to identify the most suitable polynomial regression model, as illustrated in the below figure:

`

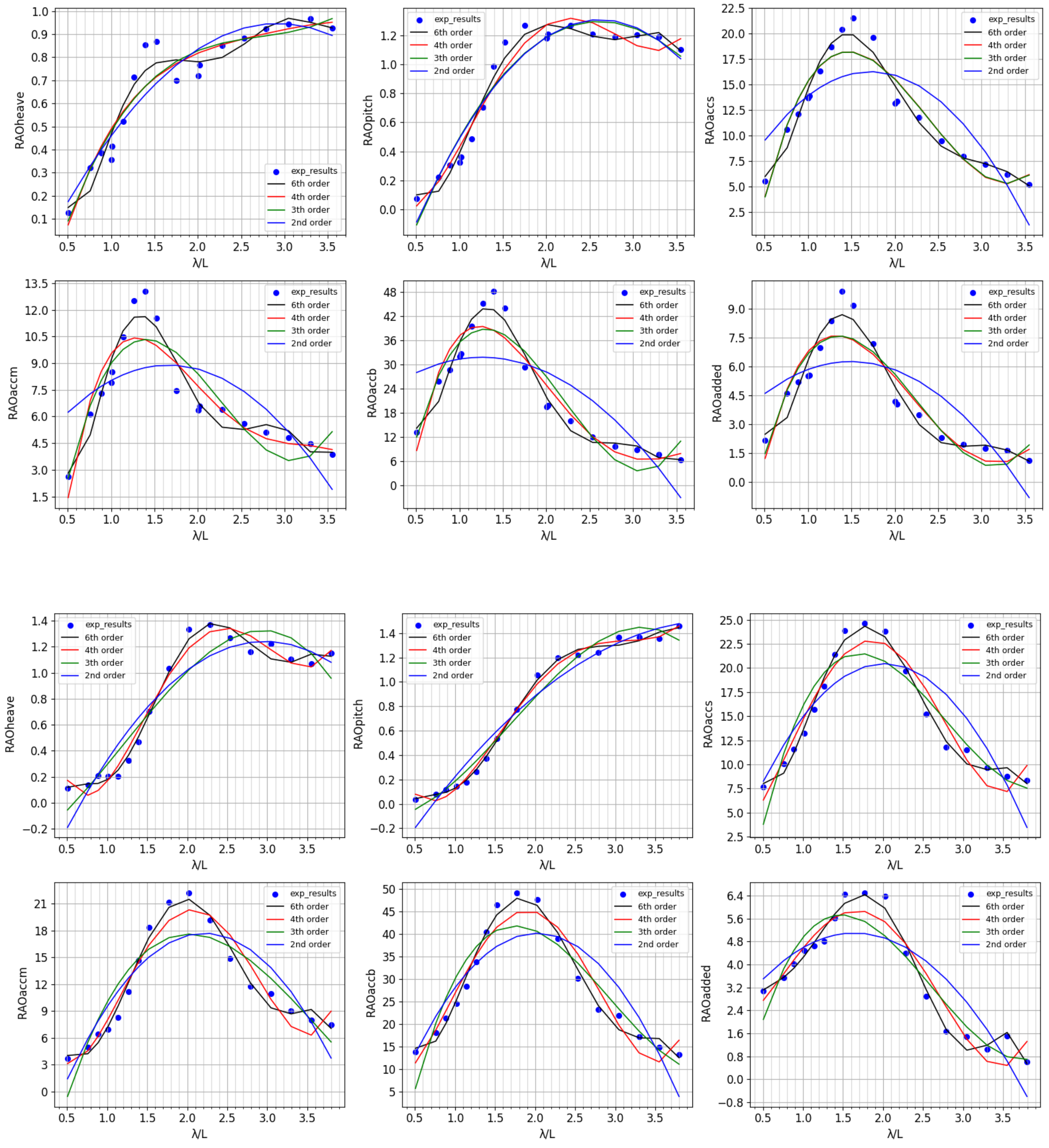


Figure 20: *Parent* hull form RAO curves with different order polynomials for Fn=0.34 and Fn=0.68 and Cdl=3.0.

The former is depicted first and the latter follows.

`

The heave and pitch responses of the model can be well approximated by lower order polynomials, whereas the accelerations and added resistance require higher order polynomials for an accurate representation. This can be confirmed by fitting a third order polynomial in the above dataset for heave and pitch and higher order polynomial for added resistance and bow acceleration and visualize the results:

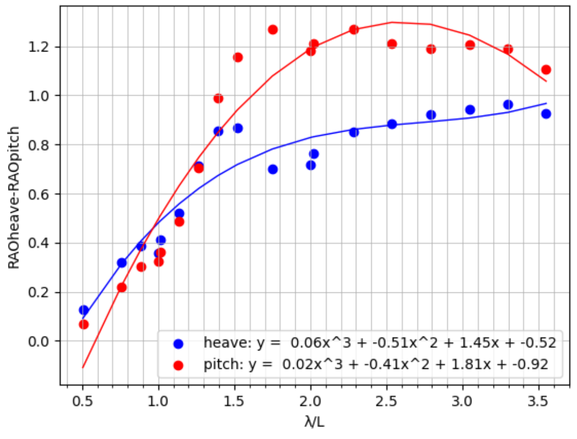


Figure 21: RAO curves for heave and pitch responses

(L/B=5.0, Cdl=3.00, Fn=0.34)

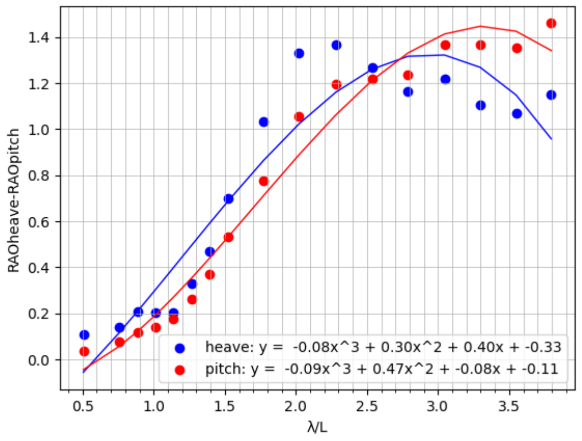


Figure22: RAO curves for heave and pitch responses

(L/B=5.0, Cdl=3.00, Fn=0.68)

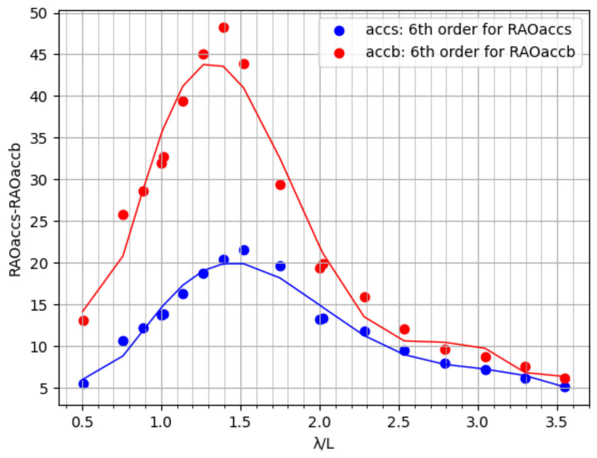


Figure 23: RAO curves for bow and stern accelerations

(L/B=5.0, Cdl=3.0, Fn=0.34)

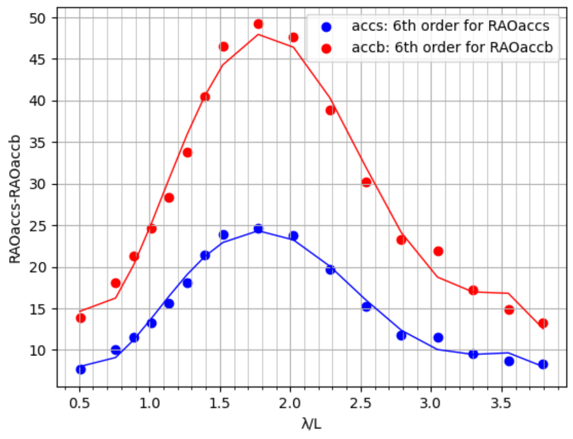


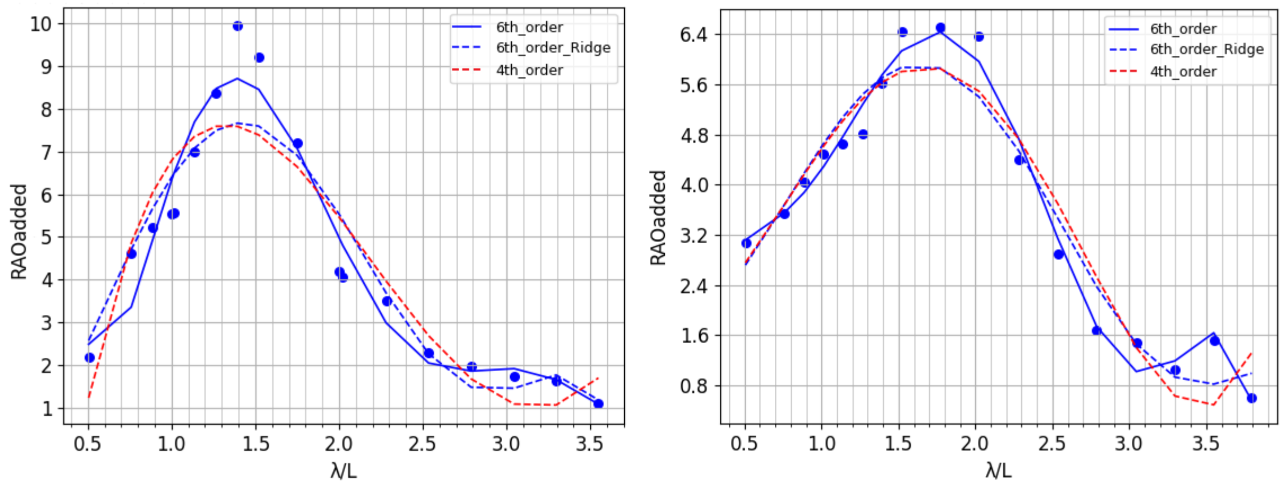
Figure 24: RAO curves for bow and stern accelerations

(L/B=5.0, Cdl=3.0, Fn=0.68)

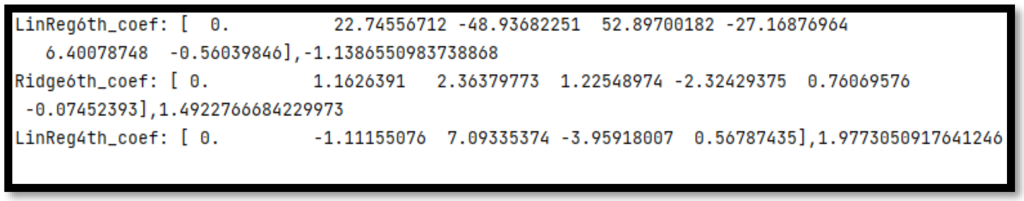
`

Indeed, 3rd order polynomial is capable of capturing the trend of heave and pitch with good accuracy. As for accelerations seem from a cursory look, seems to be well approximated by the high order polynomials. However, further investigation is needed for higher than 4rth order polynomials since they frequently lead to overfitted models. One symptom of overfitting is when the polynomial has large coefficients which are highly sensitive to small changes in the input data. This indicates that the model is trying to fit the data exactly and capture even the smallest variations in the data. Such a model may not generalize well to new data and may instead fit the random fluctuations to the dataset, rather than fit the underline trend.

To address this issue, one approach is to use either lower order polynomials or regularization techniques, such as L2 regularization (Ridge regression). L2 regularization adds a penalty term to the coefficients in the polynomial equation, which encourages the model to approach zero. This penalty term can help to reduce the magnitude of the coefficients and prevent overfitting. By using regularization techniques, we can strike a balance between fitting the data accurately and generalizing well to new data. This can help to ensure that the model is robust and reliable, and can be used to make accurate predictions for new datasets. An example of the effect of the L2 regularization technique is demonstrated for both Froude numbers and 6th order polynomial, along with 4th order polynomial. All models trying to approximate the RAOs for added resistance.

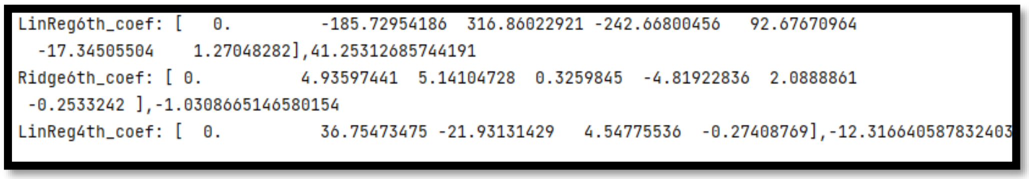


Coefficients for Fn=0.68 (right figure)



```
LinReg6th_coef: [  0.          22.74556712 -48.93682251  52.89700182 -27.16876964
   6.40078748  -0.56039846],-1.1386550983738868
Ridge6th_coef: [ 0.          1.1626391   2.36379773  1.22548974 -2.32429375  0.76069576
 -0.07452393],1.4922766684229973
LinReg4th_coef: [ 0.         -1.11155076  7.09335374 -3.95918007  0.56787435],1.9773050917641246
```

Coefficients for Fn=0.34 (left figure)



```
LinReg6th_coef: [   0.         -185.72954186  316.86022921 -242.66800456   92.67670964
  -17.34505504    1.27048282],41.25312685744191
Ridge6th_coef: [ 0.          4.93597441  5.14104728  0.3259845  -4.81922836  2.0888861
 -0.2533242 ],-1.0308665146580154
LinReg4th_coef: [  0.          36.75473475 -21.93131429   4.54775536  -0.27408769],-12.316640587832403
```

Figure 25: Linear and Ridge regression techniques for least square error minimization of RAOs for added resistance and the corresponding coefficients for Fn=0.34 (left) and Fn=0.34 (right)

`

3.4 Peacewise interpolation methods

Piecewise interpolation is a general method for approximating a function using a combination of simpler functions defined over smaller intervals. The idea behind piecewise interpolation is to divide the domain of the function into a set of smaller intervals, and then to construct a function that agrees with the original function at a set of predetermined points within each interval. The resulting function is a combination of simpler functions, each of which is chosen to match the behaviour of the original function at the interpolation points.

Piecewise interpolation is particularly useful when the function to be approximated is too complex or too expensive to evaluate directly, or when the function is only known at a set of discrete points. By dividing the domain into smaller intervals, we can approximate the function using simpler functions that are easier to evaluate or integrate. This can be especially helpful when working with data that contains noise or irregularities, as piecewise interpolation can smooth out these irregularities and produce a more accurate approximation of the underlying function.

There are several different methods of piecewise interpolation, including linear interpolation, quadratic interpolation, and cubic spline interpolation. Each of these methods involves constructing a function that is defined piecewise over smaller intervals and agrees with the original function at the interpolation points. The choice of interpolation method depends on the specific application and the desired level of accuracy or smoothness in the resulting approximation.

Peacewise linear interpolation

Supposing a set of interpolation points $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ and we want to approximate the function f(x) between these points using a piecewise linear function. The piecewise linear function f(x) is given by:

$$f(x) = y_i + (y_{i+1} - y_i) * \frac{x - x_i}{x_{i+1} - x_i}, \qquad for\ x_i \le x \le x_{i+1}$$

Where yi and yi+1 are the y-values of the interpolation points that bracket the interval containing x. This function is a piecewise linear function that agrees with the original function at the interpolation points $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$. By applying the peace-wise linear interpolation method to the subject dataset, the following plots are derived for the response amplitude operators:

`

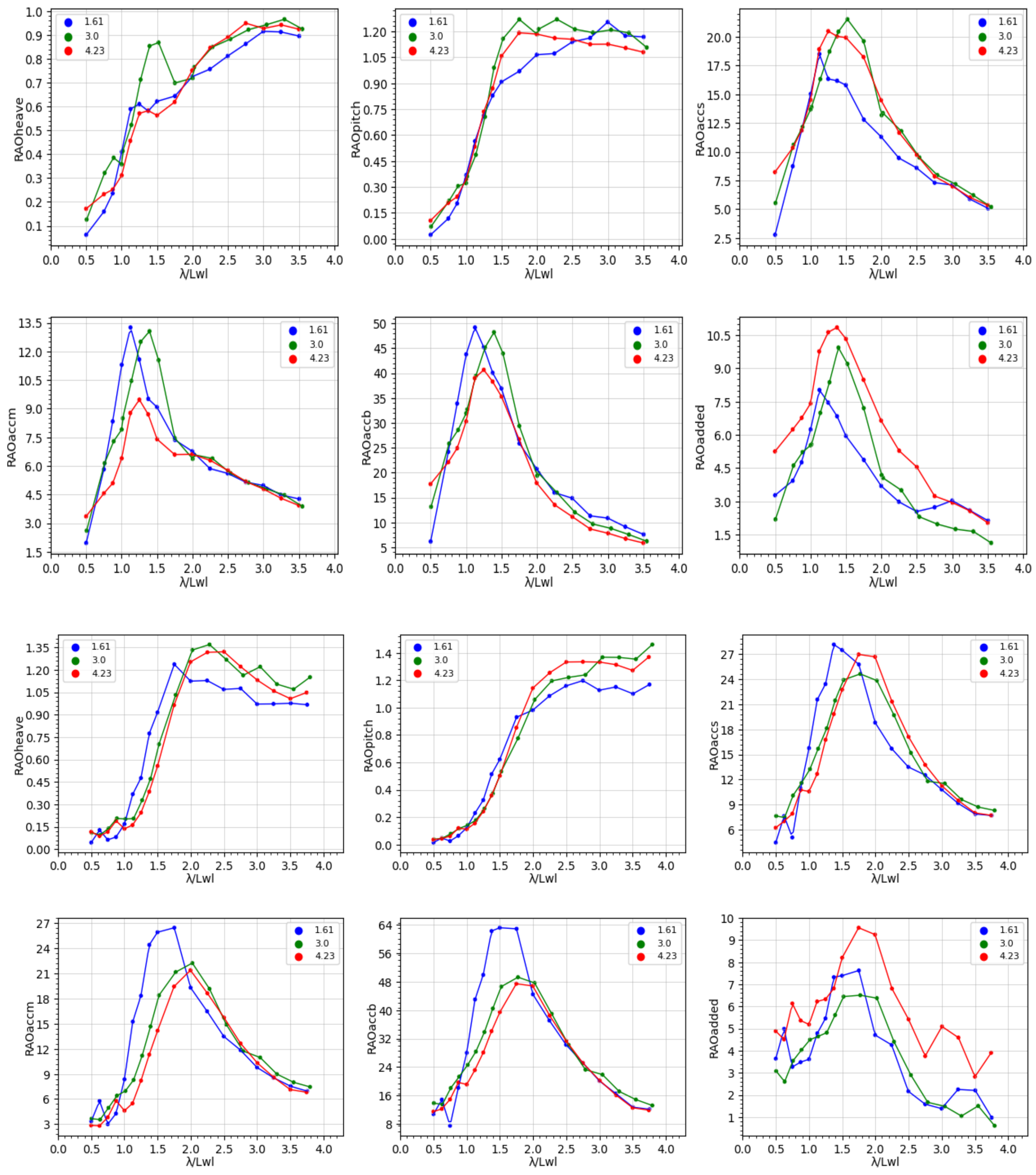


Figure 26: Parent hull form RAO curves with linear interpolation for Fn=0.34 and Fn=0.68 and Cdl=[1.61,3.00,4.23]. The former is depicted first and the latter follows.

`

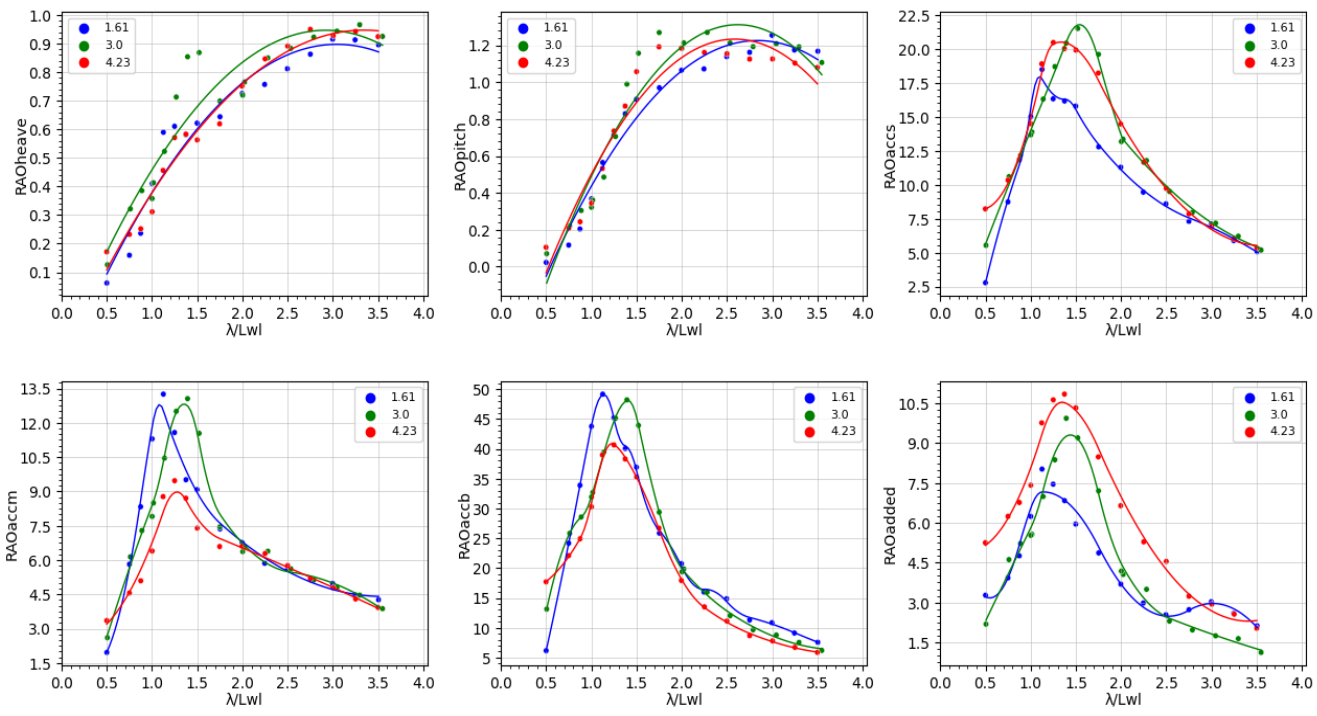
Peacewise quadratic interpolation

Suppose we have a set of interpolation points $(x_0, y_0), (x_1, y_1), \ldots , (x_n, y_n)$ and we want to approximate the function f(x) between these points using a piecewise quadratic function. The piecewise quadratic function f(x) is given by:

$$f(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 \quad , \qquad for \;\; x_i \leq x \leq x_{i+1}$$

where ai, bi, and ci are the coefficients of the quadratic polynomial that approximates the function f(x) over the interval [xi, xi+1]. These coefficients are determined by solving a system of equations that ensures the resulting function is continuous and has continuous first derivatives across the interval boundaries.

By applying the quadratic interpolation combined with a smoothing factor for the corresponding curve, the following results for the parent model (L/B = 5.00) for both Fn=0.34 and Fn=0.68 are derived. It can be observed that a smoothing factor is applied, the shape of the quadratic functions is altered appropriately to describe the underline trend of the datapoints. This becomes apparent if someone compares piecewise quadratic interpolation with and without the smoothing factor, as it is illustrated below. In the context of peace-wise function, a smoothing factor (also known as a regularization parameter or a penalty term) controls the trade-off between fitting the data closely and avoiding overfitting. Overfitting refers to capturing the random fluctuations in the dataset.



`

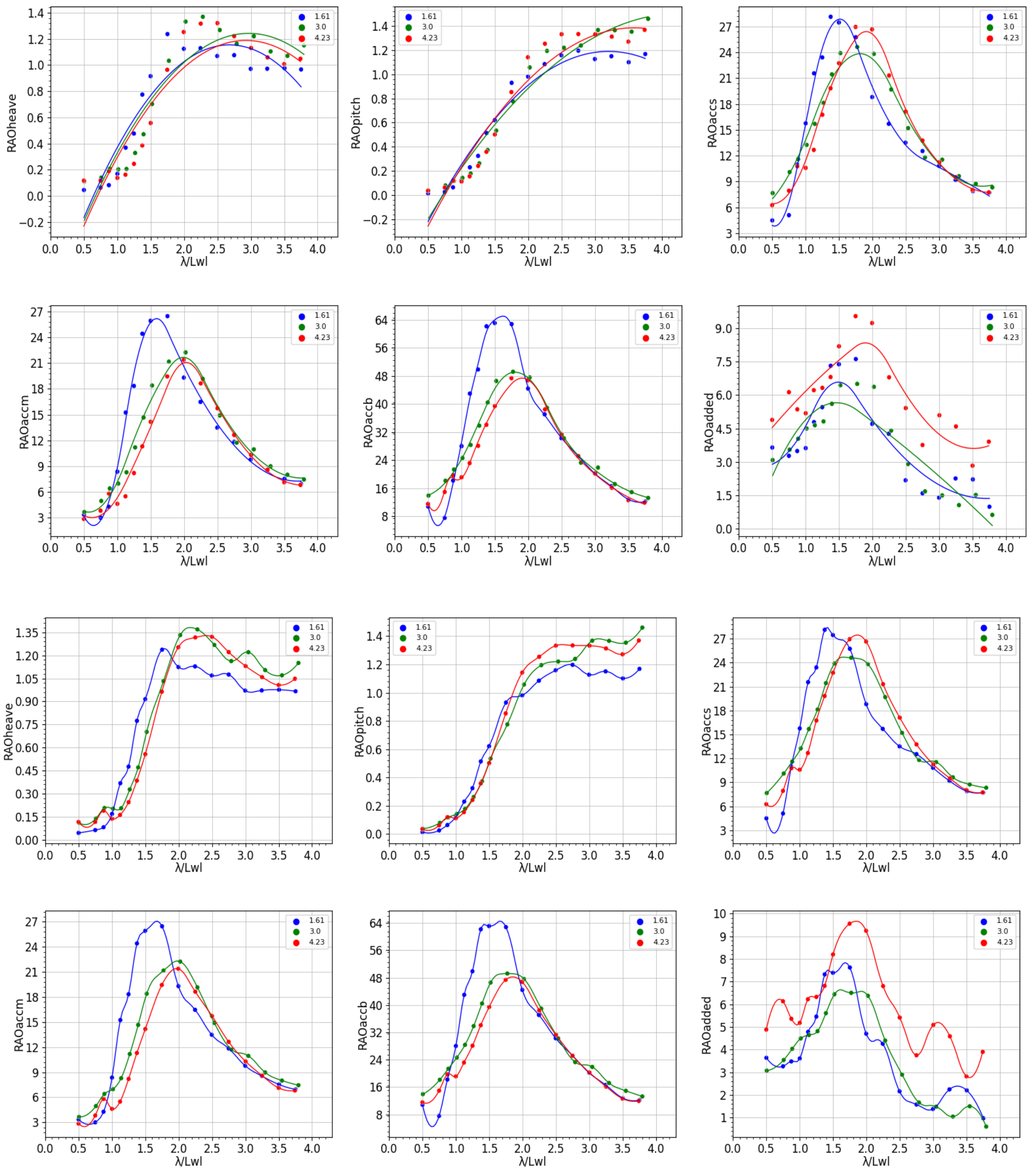


Figure 27: The first two figures correspond to parent hull form RAO curves with quadratic interpolation combined with smoothing factor for Fn=0.34 and Fn=0.68 respectively. The 3rd figure corresponds to quadratic interpolation without the smoothing factor for Fn=0.68.

`

Cubic spline interpolation

Cubic spline interpolation is a piecewise interpolation method that uses cubic polynomials to approximate a function f(x) between a set of predetermined points. The cubic polynomials are chosen so that they pass through each of the interpolation points and have continuous first and second derivatives at the boundaries between the intervals.

To construct the piecewise cubic function, we start by defining a set of n intervals $[x_0, x_1]$, $[x_1, x_2]$, … , $[x_{n-1}, x_n]$, where the interpolation points $x_0, x_1, \dots, x_n$ are ordered such that $x_0 < x_1 < \cdots < x_n$. Then a set of n-1 cubic polynomials $S_0(x), S_1(x), \dots, S_{n-1}(x)$, is defined such that:

- $S_i(x)$ is a cubic polynomial defined over the interval. $[x_i, x_{i+1}]$, for i = 0, 1, ..., n-2.
- $S_i(x)$ passes through the interpolation points (x_i, y_i) and (x_{i+1}, y_{i+1}), for i = 0, 1, ..., n-2
- $S_i(x)$ has continuous first and second derivatives at the boundaries between the intervals, i.e.,
  $S''_{i-1}(x_i) = S_i''(x_i)$ & $S_{i-1}'(x_i) = S_i'(x_i)$, for i = 1, 2, ..., n-2.

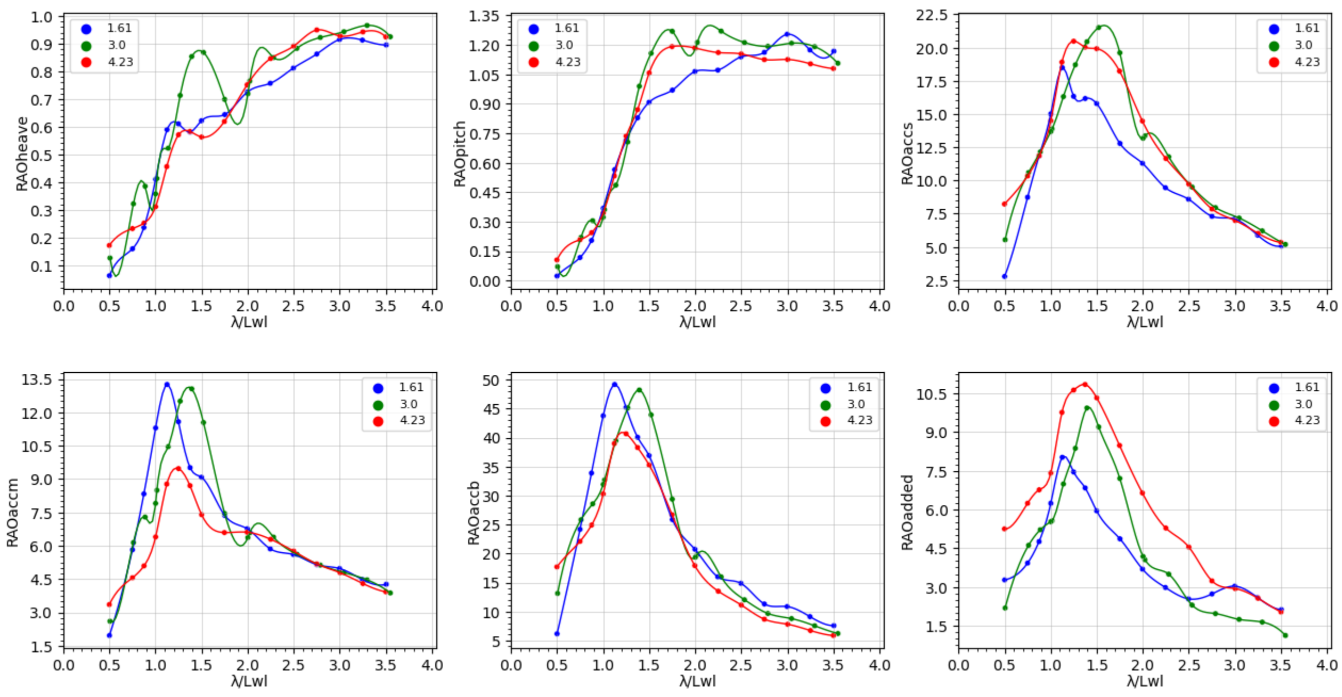
The coefficients of the cubic polynomials Si(x) can be determined by solving a system of linear equations based on the above conditions. Once the coefficients of the cubic polynomials are determined, we can define the piecewise cubic function f(x) as:

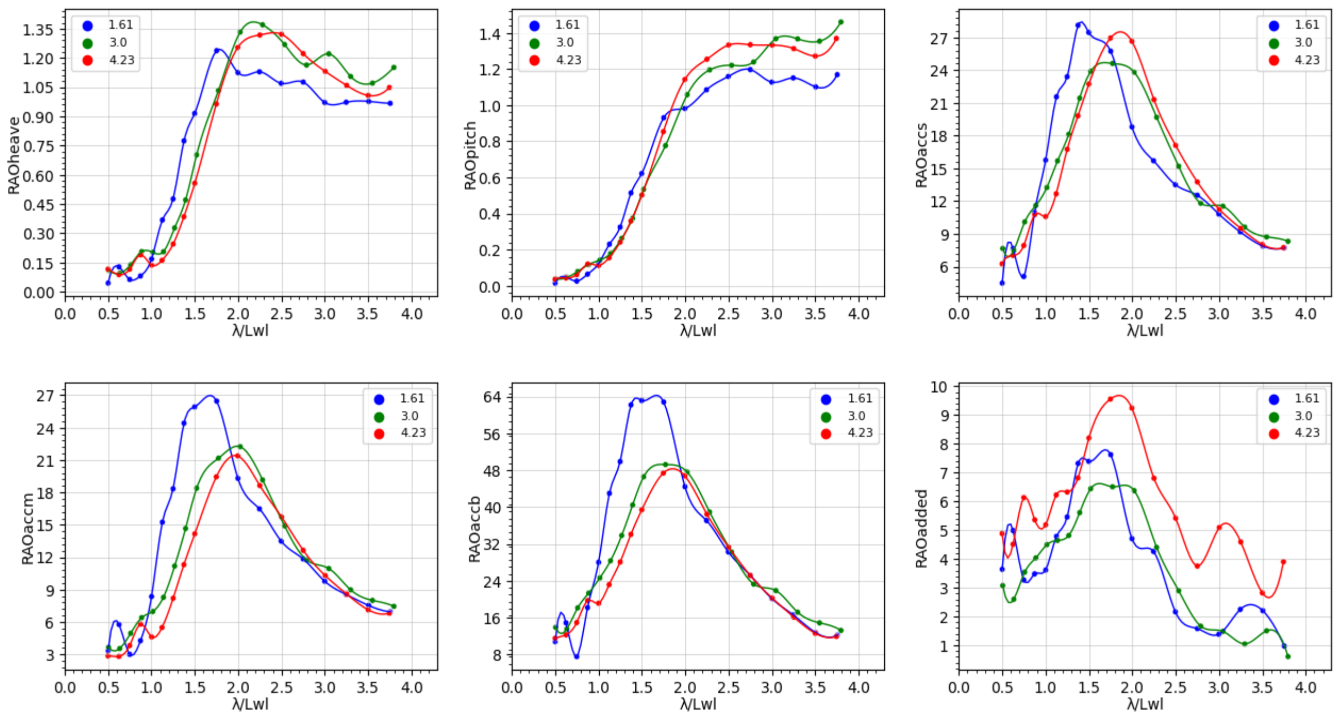
$$f(x) = S_i(x), \qquad for\ x_i < x < x_{i+1}$$

where Si(x) is the cubic polynomial defined over the interval [xi, xi+1].

The following results are for the parent model (L/B = 5.00) when fitting data points using cubic spline for Fn=0.34 and Fn=0.68. Additionally, a visualization of the data with cubic spline and a smoothing factor is included to demonstrate the beneficial effect of this regularizer on the non-linear dataset corresponding to Fn=0.68.

`


RAOheave
RAOpitch
RAOaccs
RAOaccm
RAOaccb
RAOadded
λ/Lwl
● 1.61
● 3.0
● 4.23


RAOheave
RAOpitch
RAOaccs
RAOaccm
RAOaccb
RAOadded
λ/Lwl
● 1.61
● 3.0
● 4.23

`

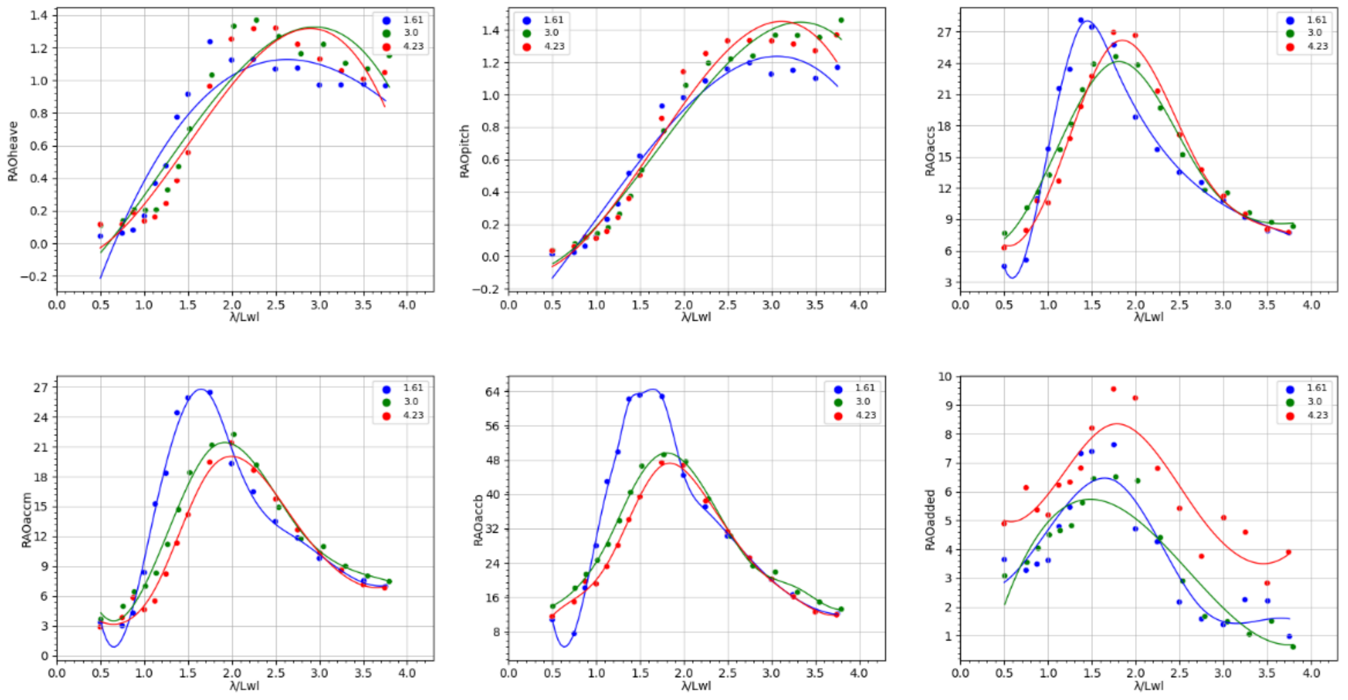


Figure 28: Parent hull form RAO curves with cubic spline interpolation for Fn=0.34 and Fn=0.68. The former is depicted first and the latter follows. The 3rd figure shows the effect of smoothing factor for cubic spline (Fn=0.68).

From the above figures it can be concluded that piecewise interpolation methods can be sensitive to the data points and result in oscillations that make the curve less smooth and less accurate. To address this issue, a smoothing factor can be introduced to these methods. The smoothing factor helps to reduce the impact of individual data points and results in a curve that more accurately represents the overall trend of the data. This is because the smoothing factor imposes a penalty on the roughness of the curve, effectively forcing it to follow a smoother path. The resulting curve obtained with a smoothing factor can be considered a more accurate approximation of the data points, as it is less likely to be influenced by picks and valleys.

For the specific dataset being analysed, which has the target values of RAO(ω) functions, it is critical to capture the local minima and maxima accurately, without being overfitted to peaks and valleys. After careful evaluation of different interpolation methods, it was decided that the most suitable model for this dataset is the quadratic piecewise function combined with smoothing factor. This method provides the required flexibility and smoothness to capture the non-linear trends in the data, while also remaining stable and not overly affected by the peaks in the data. The resulting datapoints from this approach are accurate enough and provide a good representation of the underlying data.

In conclusion, quadratic piecewise interpolation combined with smoothing factor was used for the determination of including or excluding new datapoints that included in the repeatability tests. Additionally, repeatability tests were conducted for the same wave frequency results to ensure or discourage consistency in the dataset. Based on these evaluations, the working dataset was updated to

`

include 247 samples for Fn = 0.34 and 250 for Fn = 0.68. In the next table, the effect of the data pre-processing is reflected:

Fn = 0.34		**Fn = 0.68**	
Original datapoints	320	Original datapoint	304
Datapoints after pre-processing	247	Datapoints after pre-processing	250
% Reduction	22.8%	% Reduction	17.8%

Table 6: final datasets

It is important to note that the final dataset described above was not derived solely by removing outliers, but also by addressing issues related to the repeatability of the data. Specifically, many data points with the same wave frequency were tested for repeatability, and were treated differently depending on their proximity to the best fit curve. If a data point with a certain wave frequency was close to the best fit curve, then the mean value was obtained for that frequency. However, if a data point with the same wave frequency was found to be significantly different from the best fit curve, then that result was discarded.

3.5 Selection of input parameters for ANN

Prior the final dataset passes to the neural network model as inputs, the correlation between the input features and the target values should be checked and verified. The greater the correlations between two variables the greater the impact to each other. In the concept of neural networks, if the input features are not related to the target outputs, drives to misleading models. On the other hand, if the input features are heavily correlated with the targets may lead to overfit the models since the models would be highly depended on that features and may result in generalization issues. Another question that should be addressed is if an input feature is highly correlated with other input features and provide the same information to the model. If so, one of those features is to be applied to the model inputs. Before conducting the process, it is crucial to make an initial estimation of the potential correlation between the parameters of interest, based on the governing physical equations of the relevant phenomena. Some equations exist that directly relate certain parameters, which can be useful for our analysis are presented below:

`

Parameters	*Physical Relationships*
Response Amplitude Operator for heave	$RAO_{\xi_3} = \frac{\xi_3}{A}$
Response Amplitude Operator for pitch	$RAO_{\xi_5} = \frac{\xi_5}{kA}$
Response Amplitude Operator for stern acceleration ($\boldsymbol{a_s}$)	$RAO_{accs} = \frac{a_s L_{WL}}{gA}$
Response Amplitude Operator for mid acceleration ($\boldsymbol{a_m}$)	$RAO_{accm} = \frac{a_m L_{WL}}{gA}$
Response Amplitude Operator for bow acceleration ($\boldsymbol{a_b}$)	$RAO_{accb} = \frac{a_b L_{WL}}{gA}$
Response Amplitude Operator for added resistance ($\boldsymbol{R_{AR}}$)	$RAO_{AR} = \frac{R_{AR}}{pgB_{WL}^2}\frac{L_{WL}}{A^2}$
Wave length (λ)	$\lambda = \frac{V_w}{f}$
Froude number (Fn)	$Fn = \frac{V}{\sqrt{gLwl}}$

Table 7: Physical equations relating the parameters of interest

Heat-maps are commonly used in data analysis to explore relationships between variables. In the context of neural networks, heatmaps can be used to identify which features in the input data are most relevant for making predictions. By visualizing the relationships between the input features and the output, a heatmap can provide insight into which features have the most impact on the model's predictions.

In the below heatmap, Pearson correlation is reflected which is a measure of the linear correlation between two variables. It measures the strength and direction of the linear relationship between two continuous variables x and y. It has a value between -1 and +1, where a value of -1 indicates a perfect negative linear correlation, 0 indicates no linear correlation, and +1 indicates a perfect positive linear correlation. Pearson correlation is expressed mathematically as:

$$r = \frac{cov(x, y)}{std(x)std(y)}$$

The covariance between x and y, cov (x, y) measures the joint variability of the two variables, while the standard deviations of x and x measure the variability of each variable separately. By dividing the covariance by the product of the standard deviations, the Pearson correlation coefficient normalizes the covariance to be between -1 and +1, regardless of the scales of the two variables.

The covariance is defined as the average of the product of the deviations of x and y from their respective means, weighted by the number of observations (n):

$$cov(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

`

where i ranges from 1 to n, x_i and y_i are the observations of x_i and y_i, respectively, and $\bar{x}$ and $\bar{y}$ are the sample means of x and y, respectively and n is the number of observations.

The variance of the observations can be estimated for a single variable x, using the formula:

$$var(\sigma^2) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^\wedge 2 \rightarrow std(\sigma) = \sqrt{var}$$

where i ranges from 1 to n, x_i is the i^{th} observation of x, μ is the population mean of *x,* and s is the sample standard deviation of *x*. The heatmaps generated for the inputs and target variables provide valuable insights into the relationships between variables. The below heatmaps reflects the Pearson correlation coefficient for visualizing the correlations of x and y variables. The first depicts the correlations for each pair of variables, while the second reflects the correlations of input and target variables.

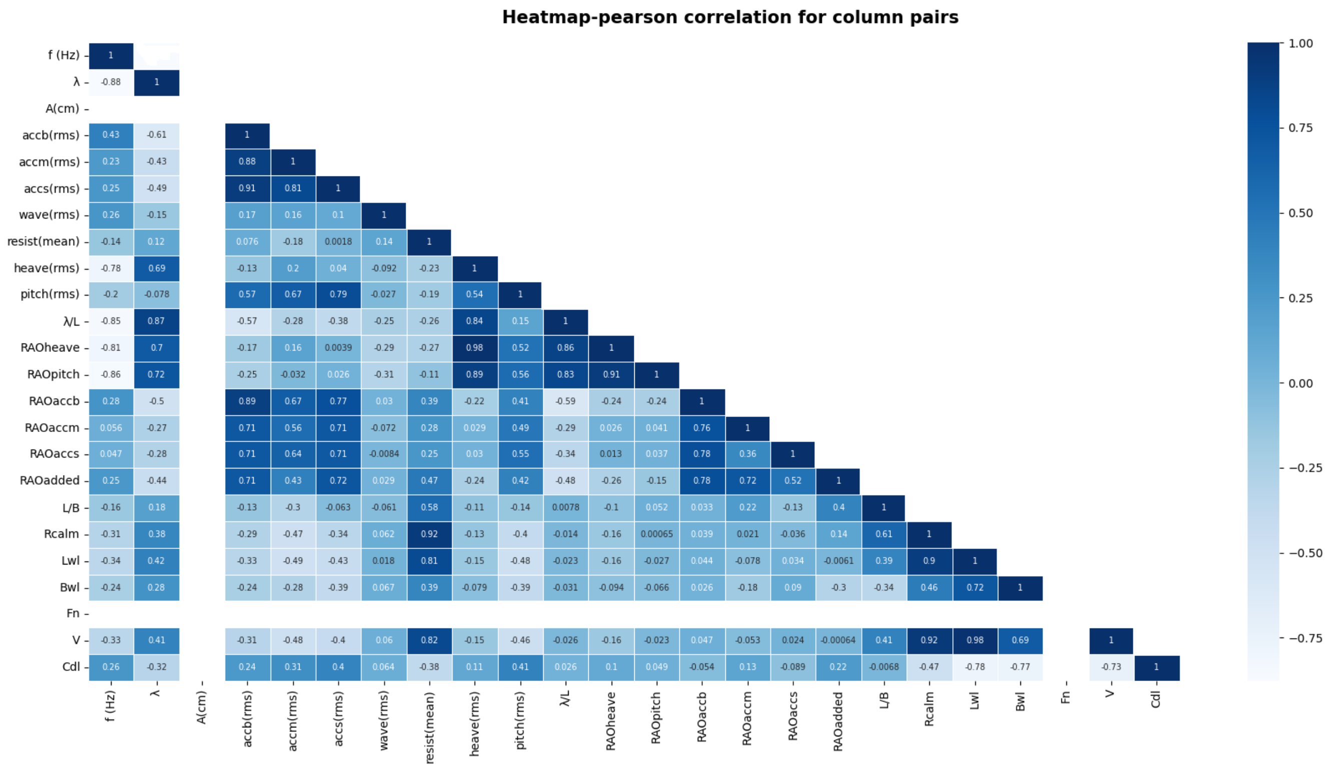
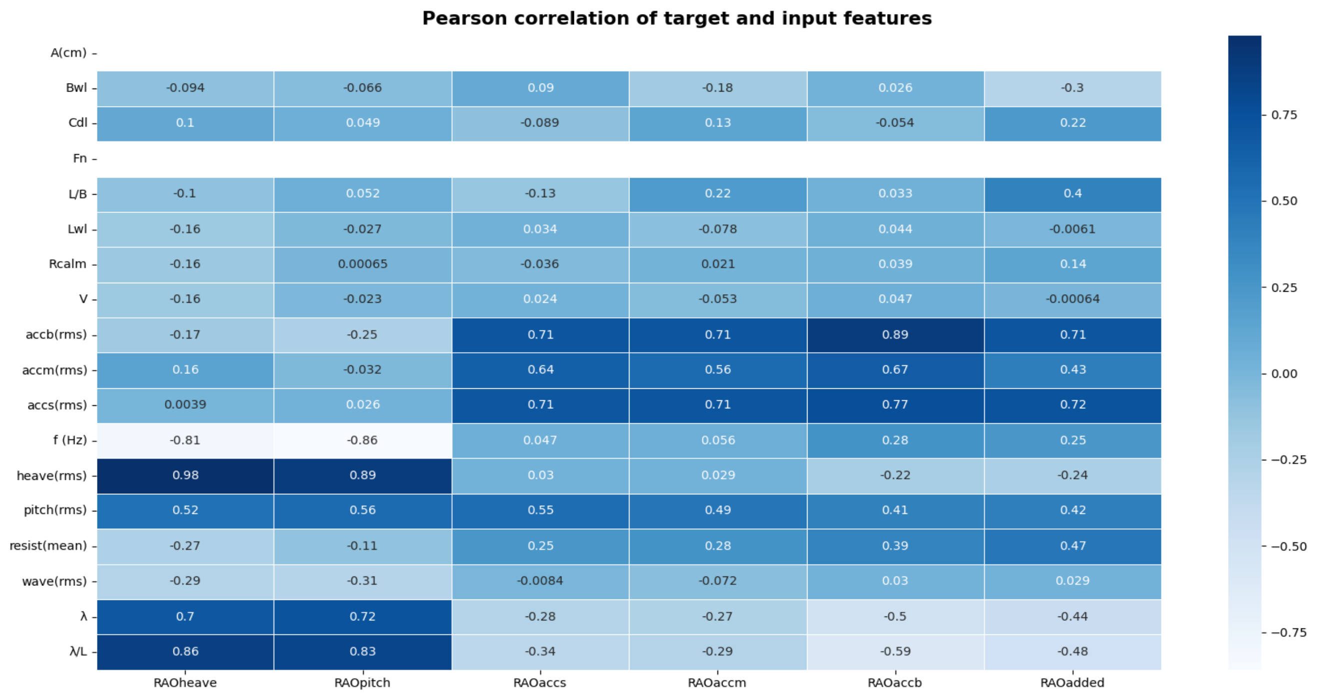


Figure 29: The theoritical expected correlation of the available parameters for Fn=0.34

`



	RAOheave	RAOpitch	RAOaccs	RAOaccm	RAOaccb	RAOadded
A(cm)						
Bwl	-0.094	-0.066	0.09	-0.18	0.026	-0.3
Cdl	0.1	0.049	-0.089	0.13	-0.054	0.22
Fn						
L/B	-0.1	0.052	-0.13	0.22	0.033	0.4
Lwl	-0.16	-0.027	0.034	-0.078	0.044	-0.0061
Rcalm	-0.16	0.00065	-0.036	0.021	0.039	0.14
V	-0.16	-0.023	0.024	-0.053	0.047	-0.00064
accb(rms)	-0.17	-0.25	0.71	0.71	0.89	0.71
accm(rms)	0.16	-0.032	0.64	0.56	0.67	0.43
accs(rms)	0.0039	0.026	0.71	0.71	0.77	0.72
f (Hz)	-0.81	-0.86	0.047	0.056	0.28	0.25
heave(rms)	0.98	0.89	0.03	0.029	-0.22	-0.24
pitch(rms)	0.52	0.56	0.55	0.49	0.41	0.42
resist(mean)	-0.27	-0.11	0.25	0.28	0.39	0.47
wave(rms)	-0.29	-0.31	-0.0084	-0.072	0.03	0.029
λ	0.7	0.72	-0.28	-0.27	-0.5	-0.44
λ/L	0.86	0.83	-0.34	-0.29	-0.59	-0.48

Table 8: The theoritical expected correlation of the target and input parameters for Fn=0.34

The evaluation of the correlation coefficients was performed by creating a matrix of pair-plots, which displays the strength of linear relationships between the dataset parameters.

A group of pair-plots was generated for the input features, which allowed for the visualization of the correlation between each pair of parameters. Before we further proceed to ANN section, it is useful to obtain an overview of the variables in a system and their interrelationships, which can provide a clearer understanding of the big picture. When analysing a dataset, it is advisable to identify highly correlated pairs of columns. If these columns do not provide any additional information beyond what is already captured by the other column, it may be redundant to keep both of them. In such cases, it is recommended to remove one of the correlated parameters to avoid redundancy and simplify the dataset. For example, the parameters f and λ are related through the equation $\lambda = \frac{V_w}{f}$, which furthers supports the recommendation to remove one of these parameters from the dataset. The λ parameter was decided to be removed since the effect of λ exists in the parameter λ/Lwl. As for constant parameters like Fn, should be avoided since they are not providing any useful information to the model.

Regarding heave, pitch, and accelerations parameters, it is obvious that they are highly correlated with the target outputs $(RAO_{\xi_3}, RAO_{\xi_5}, RAO_{acc})$. These features will be included in the inputs of neural networks, because it would be impractical to have a model that can predict the response amplitude operators without considering the responses.

The figures below illustrate the underlying relationships between the input variables as well as between the input and target variables. The inputs were grouped based on the loading factor Cdl. Blue points correspond the lighter Cdl (1.61), green points to the heavier Cdl (4.23) and orange points to Cdl=3.00.

`

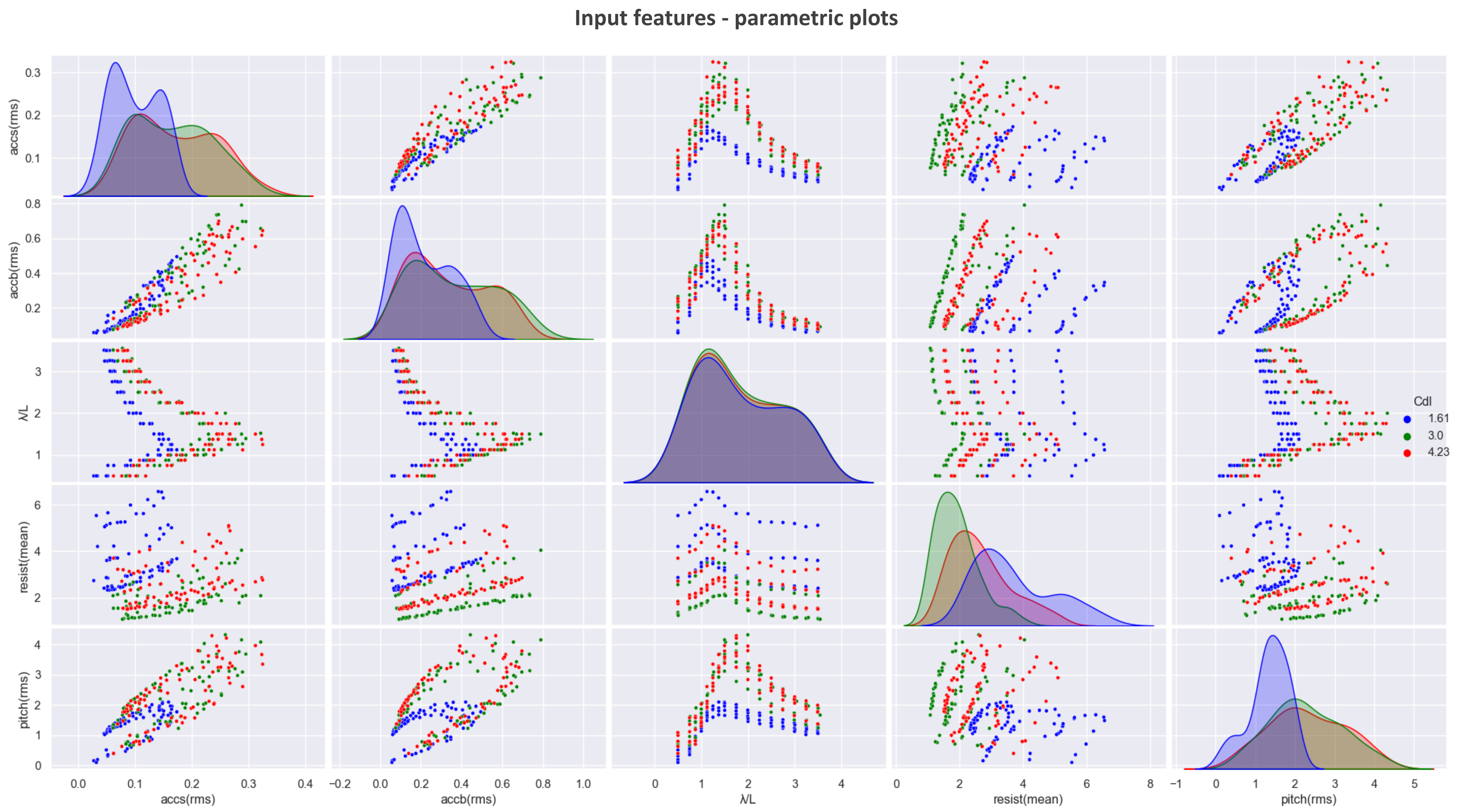


Figure 30: Correlation of input variables, pair-plots for inputs & density distribution for each input variable (diagonal)

`

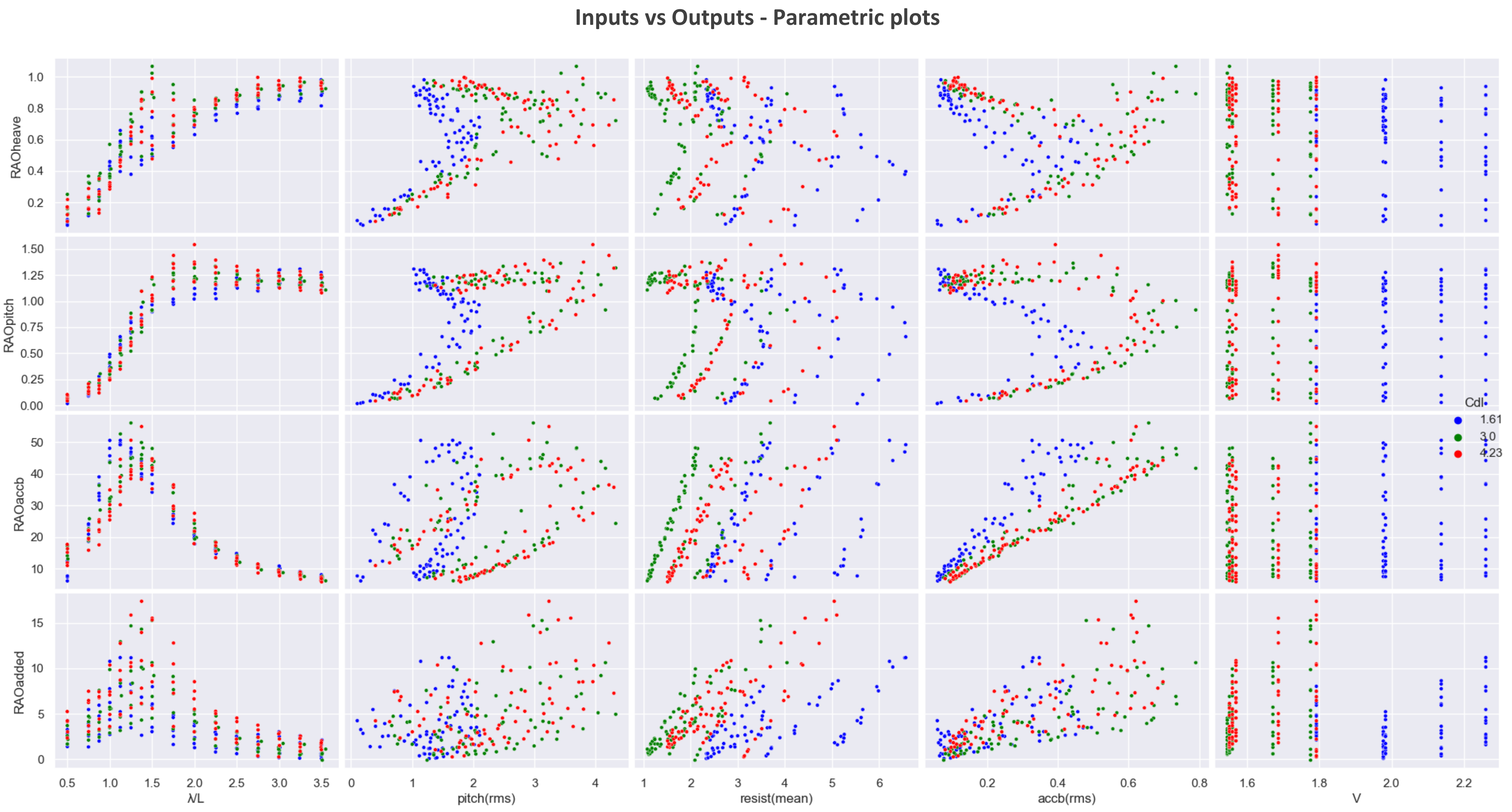


Figure 31: Pair-plots for the correlation of input and target variables

`

4. ANN IMPLEMENTATION

Understanding the architecture and parameters of artificial neural networks (ANNs) is crucial for developing effective and high-performing models. This idea is reinforced in this chapter, which presents the results of ANN model training and testing considering a variety of architectures before choosing final. The chapter aims to demonstrate the fundamental importance of ANNs in this study, and to provide insight into the key factors that influence their performance.

The typical workflow for building a neural network model is summarized in the following diagram:

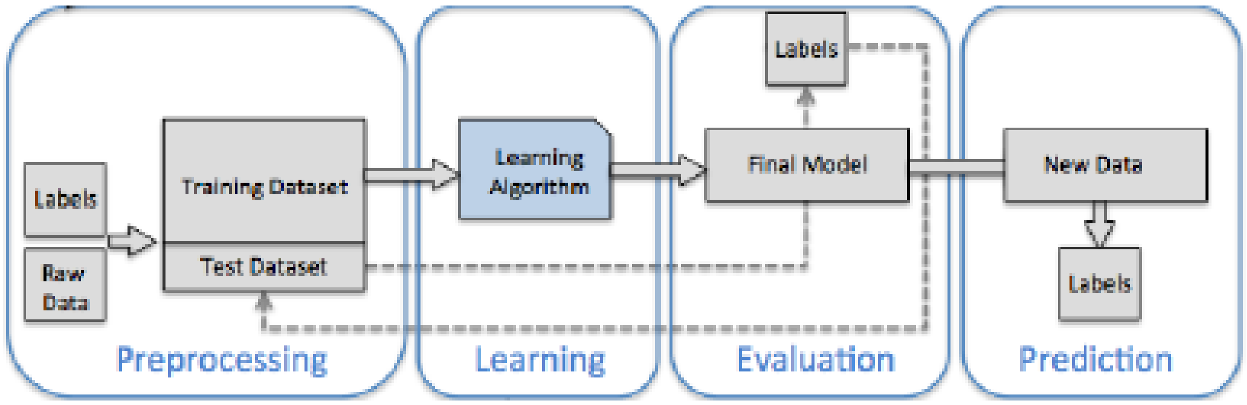


Figure 32: ANN process workflow

4.1 Pre-processing

Regarding the data pre-processing apart from feature selection technique utilized for inputs in the previous chapter (Pearson correlation coefficient of determination), the below specific sequence of steps is following:

- Shuffling the dataset.
- Splitting the data set into the training and validation sets
- Data Normalizing

Shuffling the dataset

`

Shuffling the data ensures that the model is trained on a diverse range of examples from throughout the model’s operation, allowing it to learn more effectively and generalize better to new data.

The split of the data

The data is typically split into training and validation sets, with percentage 80-20% respectively. This split percentage determines a fraction of the training data to be used as validation data. The model will set apart this fraction of the training data, will not train on it, and will evaluate the loss and any model metrics on this data at the end of each epoch. Next step is to separate target values(labels) from features. Labels are the values that model will be trained to predict.

Data normalization

By normalizing the input data, each feature is transformed to have similar ranges and distributions, which can help to reduce the impact of any one feature on the model's output. This can enable the model to learn the underlying patterns in the data more efficiently and improve its generalization capabilities, resulting in better performance on new, unseen data. The normalization used is simple mean and variance normalization. For each feature dimension, the layer first subtracts the mean of that dimension and then divides by the standard deviation of that dimension. Given a dataset X = ($\mathrm{x_i}$, $\mathrm{x_i}$, ..., $\mathrm{x_i}$), the normalized dataset $\mathrm{X_{norm}}$ = ($\mathrm{x_{1norm}}$, $\mathrm{x_{2norm}}$, ..., $\mathrm{x_{nnorm}}$) is defined as:

$$\mathrm{x_{i_{norm}}} = \frac{\mathrm{x_i} - mean(X)}{\mathrm{std(X)}}$$

where:

- $\mathrm{x_i}$ is the i-th element in X.
- mean(X) is the mean of X.
- std(X) is the standard deviation of X.

The formula subtracts the mean of the dataset from each element, which centers the dataset around 0. Then, each element is divided by the standard deviation of the dataset, which scales the dataset so that it has unit variance. This normalization technique is commonly used in machine learning to help algorithms converge faster during training and to prevent some features from dominating others due to differences in scale.

4.2 Training and evaluation

Regarding the learning and evaluation process the following user defined parameters have been taken into account for building ANN models.

- User defined network parameters:

1. Cost function
2. Validation
3. Performance metrics
4. Optimizer (Training Algorithm)
5. Hyperparameters:

`

- The number of layers
- The number of neurons in each layer
- The activation function of each layer
- Learning rate
- Batch size
- Number of epochs

Each step in the training process is important in ensuring that the model performs well and is able to generalize to new data. While the ultimate goal is to minimize the error function, it's not desirable to have a model that is overfitted to the training data and lacks generalization capabilities, even if it results in a low training error. Therefore, the significance and advantages of each step in the training procedure is demonstrated below:

Cost Function

Mean Squared error is mainly used as the error-loss function for regression problems, and this common practice is adopted herein as well. It measures the average squared difference between the predicted and actual values in a regression problem. The MSE cost function also penalizes large errors more heavily than small errors due to the squaring operation. The MSE is calculated using the following steps:

1. The error(residual) is calculated for every data point as $(y_i - d_i)$
2. the square value of residuals is calculated
3. The average of results from the above step are obtained:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - d_i)^2$$

Optimizer

When training a neural network, its weights are initially initialized randomly and then they are updated in each epoch in a manner such that they increase the overall accuracy of the network. In each epoch, the output of the training data is compared to actual data with the help of the loss function to calculate the error and then the weight is updated accordingly.

The group of backpropagation algorithms are used as optimizers. This family of algorithms can adjust the weights in respect to the steepest descent direction (negative of the gradient). This is the direction in which the cost function is decreasing most rapidly. In the case of the standard gradient descent algorithms the weights and biases are updated following this method, whereas the learning rate remains constant. An alternative is achieved by adjusting the learning rate of the back-propagation, in order to adapt the local curvature of the error surface. The concept is to increase learning rate when the error declines, as

`

long as the stability of the network is ensured. An aspect that may improve further their learning time is to take into account the momentum of the error function, aiming to avoid local minimum areas and converge faster to the global minimum.

A brief description of the algorithms that utilized for the ANN training is given below:

Stochastic gradient descent (SGD) is a popular optimizer for training ANNs, which uses gradient descent and error backpropagation. In the stochastic gradient descent (SGD) optimization algorithm, the gradients are calculated and the weights are updated for each training example (i.e., one example at a time). This can be computationally expensive, especially for large datasets. Therefore, batch and mini-batch methods were developed as more efficient alternatives.

Batch gradient descent involves computing the gradients for the entire training set, then updating the weights once per epoch. This is computationally more efficient than stochastic gradient descent, but may converge more slowly or get stuck in local minima.

Mini-batch gradient descent is a compromise between SGD and batch gradient descent, where the gradients are computed and the weights are updated for a small batch of randomly selected training examples at a time. When the batch size falls between 1 and the total number of training samples, it's referred to as a mini-batch method. This can reduce the computational burden of computing gradients for the entire training set, while still providing enough variance to avoid getting stuck in local minima. Mini-batch gradient descent is often used in practice because it can provide a good trade-off between efficiency and convergence.

Nesterov accelerated gradient (NAG) algorithm is an optimization algorithm for finding the minimum of a function. It is a variant of gradient descent that uses momentum to speed up the convergence to the minimum. The main idea behind NAG is to use an estimate of the future gradient to update the current parameters. In other words, instead of computing the gradient of the cost function at the current parameters, NAG computes the gradient at the "lookahead" position, which is an approximation of where the parameters will be after applying the momentum term to the previous update. This allows the algorithm to take larger steps in the direction of the minimum while still avoiding overshooting. NAG also has a mechanism for adapting the momentum parameter during training, which can help to further speed up convergence.

Adam is an adaptive learning rate optimization algorithm that is well-suited for networks training. The name "Adam" stands for Adaptive Moment Estimation. The algorithm uses a combination of gradient descent with momentum and root mean square propagation (RMSprop). The Adam optimizer maintains an exponentially decaying average of past gradients and squared gradients, and uses these to update the parameters of the model.

Nadam combines the adaptive learning rate of Adam with Nesterov momentum. The Nesterov momentum helps to reduce oscillations and overshooting in the optimization process by taking into account the future gradient information. This is achieved by using a modified version of the gradient that takes into account the momentum term.

`

The selection of training algorithm affects significantly the model's prediction and generalization capabilities. In this study a number of different algorithms were tested in order to conclude at the training procedure that would result in the best possible predictive characteristics. In order the effect of the training algorithm to be demonstrated two figures are constructed. The one figure example (on the left) illustrates the training progress of a neural network using different variants of stochastic gradient descent, including plain SGD, mini-batch SGD, SGD with momentum, and SGD with Nesterov and momentum, while the other hyperparameters remain constant. Similarly, different variants of Adaptive Moment Estimator (Adam). The other figure (on the right) provided below include examples of Adam optimizer with low, high, medium and adaptable learning rate. Similar examples are included for the SGD optimizer combined with momentum and Nesterov algorithms.

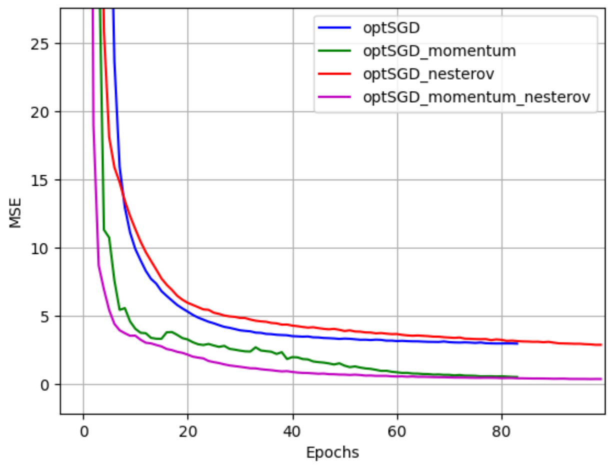


Figure 33: Adam optimizer and variants

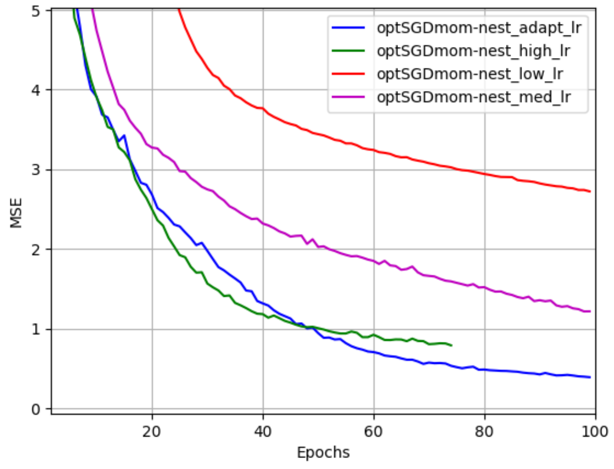


Figure 34: SGD optimizer combined with momentum and NAG for different learning rates

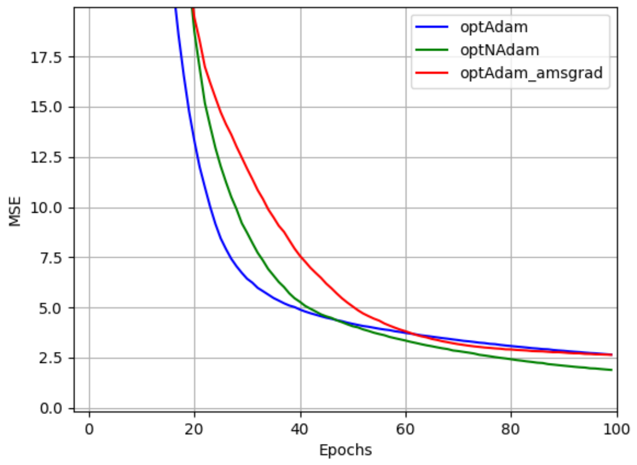


Figure 35: SGD optimizer and variants

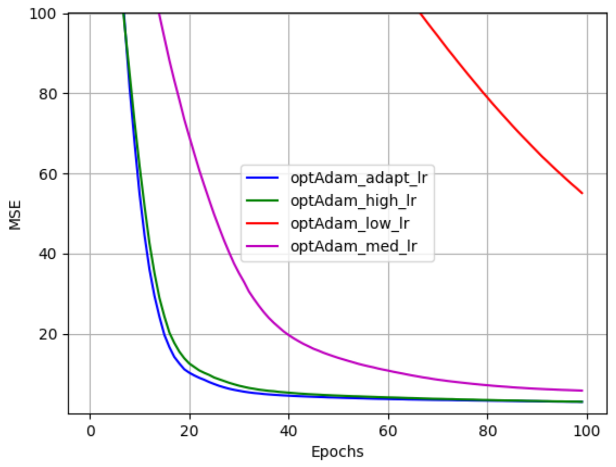


Figure 36: Adam optimizer for different learning rates

`

The figures on the left demonstrates the rapid convergence of the SGD optimizer with momentum and with Nesterov. This behaviour is due to the momentum and Nesterov updates, allowing the optimizer to move more efficiently towards the minimum of the loss function. To be more specific, momentum updates use a moving average of past gradients to update the parameters, while Nesterov updates use a slightly different approach that looks ahead in the direction of the momentum before updating the parameters. Both of these techniques have been shown to improve the convergence rate and stability of the optimization process. On the other hand, Adam optimizer and variants seem to need more epochs in order to complete their training process as according to the above figures, they have not achieved convergence yet. Right figure depicts the effect of varying learning rates on the convergence of the optimization algorithms. It is clear from the plots that the choice of learning rate has a significant impact on the speed and quality of convergence. It is essential to be cautious when choosing a high learning rate for an optimization algorithm as it can lead to unstable convergence or even divergence. When the learning rate is too high, the optimizer takes large steps that overshoot the minimum of the loss function and result in oscillations or even divergence. Therefore, it is recommended to start with a relatively low learning rate and gradually increase it while monitoring the convergence behaviour. Several techniques such as learning rate schedules or adaptive learning rates (above figure) can also be employed to control the learning rate during the training process and prevent instability. Adaptive learning rate technique found to be effective, as it allows the algorithm to converge faster and more reliably to the optimal solution. Last but not least, is should be mentioned herein that the fact of fast convergence is not coupled always with good predictions and a cross validation is always required when fitting an ANN model.

Hyperparameters

Fine-tuning an artificial neural network can be a challenging task, as there are no straightforward techniques to guide the process and manually searching all possible combinations for optimal values is inefficient. For the detection of the appropriate number of layers and neurons inside each layer was done by testing the training algorithms on a number of different layer architectures. In order this to be feasible the *number of neurons* for each layer was narrowed at the range of 1-20 and the *number of layers* was set to 1-2. These limitations were applied for the subject dataset since we are dealing with a small dataset and a more complex model with much more parameters were considered impractical for the specific dataset.

This trial approach was started by testing the training algorithms on all possible combinations of the neurons and layers, as restricted above experimenting with different kind of activation functions. The learning rate for the algorithms can be either constant or adjustable per epoch. During the initial evaluation, the algorithms used a constant learning rate for determining their training rate. Some ways to handle the limitations of activation functions, included in the most of the models, are listed below:

The process for obtaining the best performing *activation function*, implies on finding the most suitable family of activation functions (‘sigmoid’ and ‘tanh’, ‘ReLU’ and ‘ELU’ are categorized as family functions) and test a subset of them to ensure that they do not induce deviations in the model’s performance. The transfer functions used in the algorithm consisted of Exponential Linear Unit (‘ELU’) transfer function for the input layer to the first hidden layer. In case, second hidden layer exists the connection between the hidden layers was also chosen to be ELU activation function. Finally, the transfer function used for connecting the second (and last) hidden layer to the output layer was a linear transfer function, which is commonly used in Multilayer Perceptrons.

`

The *batch size* refers to the number of training samples that are fed through the neural network before adjusting the weights. With a batch size of 1, the model processes the training samples sequentially, adjusting the weights based on the error of each individual sample. When the batch size equals the total number of training samples, the model is trained using a batch method, adjusting the weights once per epoch. When the batch size falls between 1 and the total number of training samples, it's referred to as a mini-batch method. The appropriate batch size affects significantly the models' performance and a number of validation tests should be carried out to choose the tailor batch size for the case specific dataset and number of epochs.

The selection of the *number of epochs* in a neural network is closely related to the batch size used during training. Increasing the number of epochs means that the entire training dataset is passed through the model more times, but the batch size determines the number of weight updates that occur during each epoch. In other words, the total number of weight updates during training is a key factor in determining the effectiveness of the model. For example, if the number of epochs is held constant and the batch size is doubled, the number of weight updates is halved, resulting in a proportional decrease in the computational time required for the model's training. Therefore, when selecting the appropriate combination of batch size and number of epochs, it's essential to strike a balance between the number of weight updates and the computational time required to achieve an optimal level of model performance.

Tuning the hyperparameters of an optimization algorithm is a critical step in the training of deep learning models as it can significantly impact the convergence rate and final accuracy of the model. To demonstrate the importance of hyperparameter tuning, several figures have been provided below that showcase the effect of different hyperparameter settings on the convergence behaviour of specific optimization algorithms.

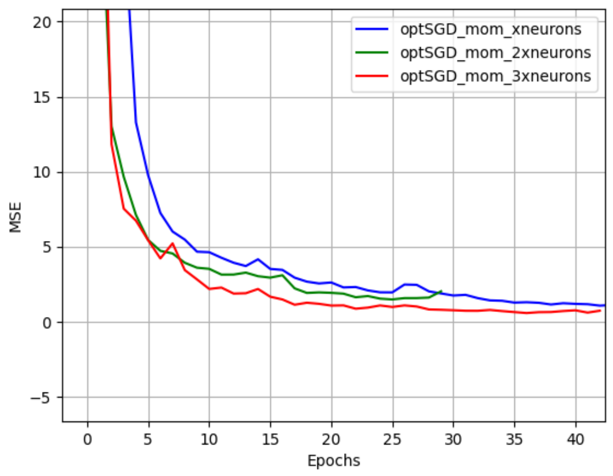


Figure 37: Combined SGD Optimizer with x, 2x and 3x number of neurons

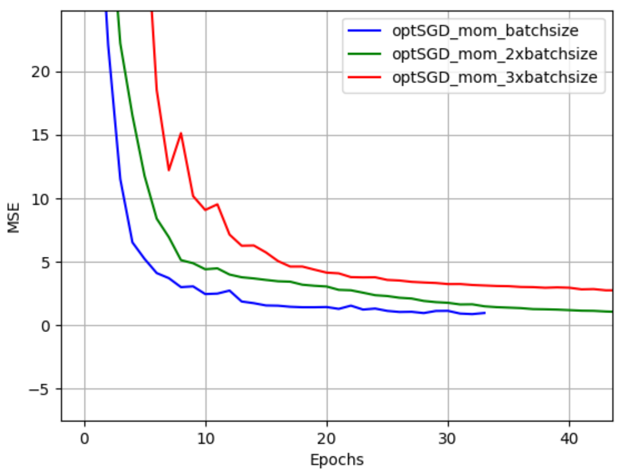


Figure 38: Combined SGD optimizer with different mini batches

`

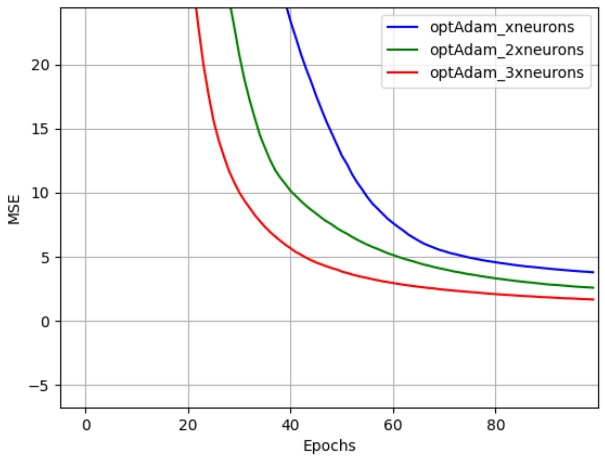


Figure 39: Adam Optimizer with x, 2x and 3x number of neurons

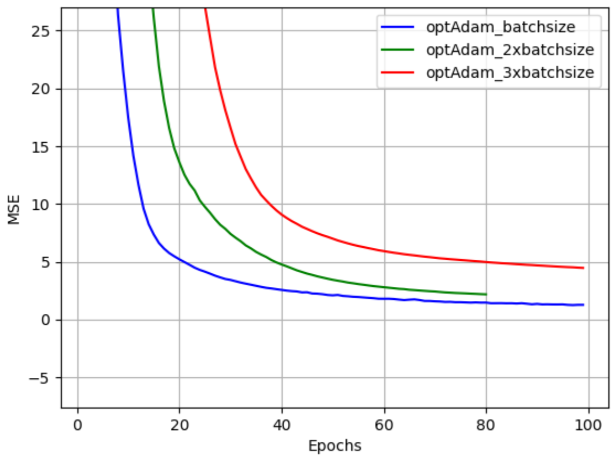


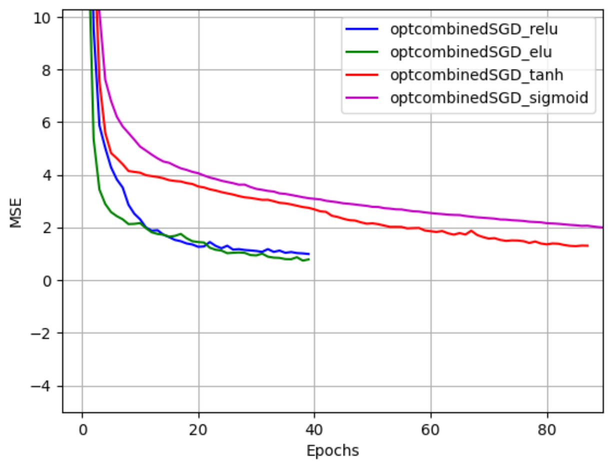
Figure 40: Adam optimizer with different mini batch methods

It can be observed from above figures that increasing the number of neurons can improve the performance of a model, particularly in cases where the task is complex and requires more representational power. At the above figures, the training process stops after five consecutive epochs with no improvement at the MSE. As the number of neurons increases, the model can capture more complex patterns and relationships in the data, leading to faster convergence and improved accuracy.

By comparing the two optimizer families seem that they are both Adam and SGD combined with momentum algorithm doing quite well in minimizing the MSE. It should be noted that the above MSE refers mean of all six mean square errors being tested and it's a good indicator for the performance of the model and for that reason it is enclosed herein.

The choice *of activation function* also affects significantly the convergence speed and stability of the optimization process. For instance, the Rectified Linear Unit (ReLU) activation function and it's variants ('elu', 'LeakyRelu') have become popular due to its simplicity and effectiveness in deep learning, as it can help alleviate the vanishing gradient problem and improve the convergence speed. However, other activation functions such as sigmoid or tanh can also be effective in certain applications. It should be noted that activation functions depend also on the specific optimizer and should be tuned in order to identify the tailored for the specific case. In our case, two optimization algorithms, one instance from the SGD family and the other from Adam family, are being subject to different activation functions and the effect is depicted below:

`



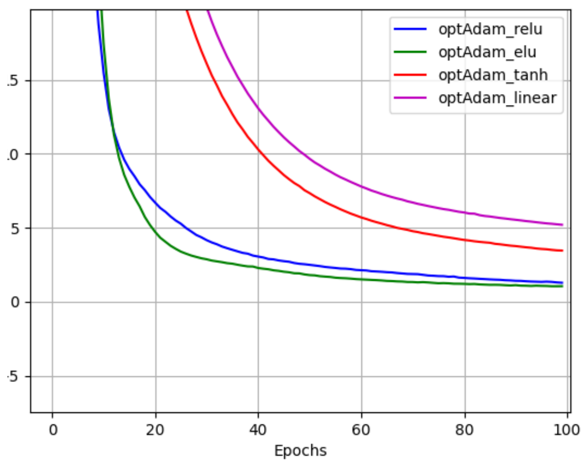


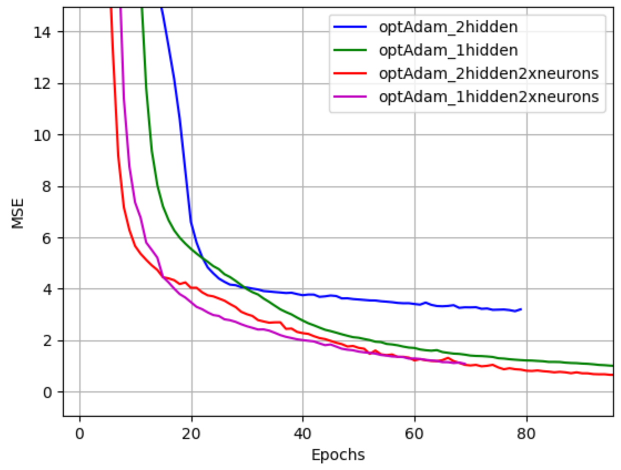
Figure 41: SGD optimizer combined with momentum and Nesterov experiencing different activation functions

Figure 42: Adam optimizer experiencing different activation functions

Both figures above indicate ReLu activation function and its variants as a good choice for this particular network architecture when using both Adam and combined SGD optimizers. This may be due to the fact that ReLu avoids the vanishing gradient problem and accelerates the convergence of the network during training. Tanh activation function as well as sigmoid seem to have similar effects while tanh is performing quite better for either sigmoid, softmax or linear activations. Also, it is worth noting that some of activation functions stopped the training process at a specific time-step(epoch). This happens due to the fact that early stopping technique has been implemented to the algorithm so that it has been configured to stop the training after five consecutive epochs without improvement in the performance metric (MSE). It should be clarified that some activation functions need their time to converge and may produce valuable results after some epochs.

The number of layers is playing also a crucial role in the training process, since each layer can learn and represent different levels of abstraction, allowing the network to extract more complex features from the input data. However, as the number of layers increases, the risk of overfitting the training data also increases, since the network may become too complex and specialized to the training data. Finding the optimal number of layers for a specific task and dataset is often an empirical process, and depends on the complexity of the data and the amount of training data available. In our case, the maximum number of hidden layers selected was two since it was considered impractical to build a more complex model for the subject dataset.

`



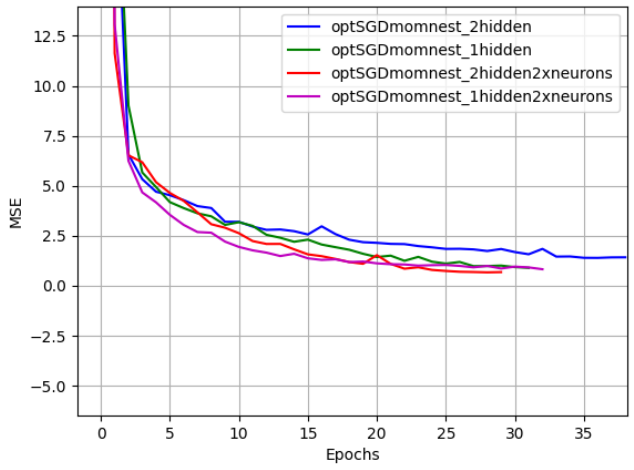


Figure 43: Adam optimizer tested at different hidden layers and neurons

Figure 44: SGD optimizer combined with momentum and Nesterov tested at different hidden layers and neurons

Two hidden layer networks seem to work quite well when increasing the number of neurons. The number of neurons in each hidden layer should be chosen carefully: Increasing the number of neurons can improve the performance of the network, but there is a point of diminishing returns. Too many neurons can lead to overfitting, which means the network becomes too specialized to the training data and performs poorly on new data. The above issue will be examined with cross validation techniques later on the study.

4.3 ANN final configuration and predictions

All the above figures represent samples of the possible combinations of hyperparameters with manual tunning. To reach the best performing algorithm the visualization of error history was extensively utilized to catch the effects of changes in hyperparameters. However, this can be tough and time-consuming process, especially when dealing with a large number of hyperparameters and possible combinations. To address this issue, Bayesian optimization techniques can be used to automate the process of hyperparameter tuning. The *hyperopt* package provides an implementation of Bayesian optimization that can be used with Keras models. In Bayesian optimization, the process of finding the best performing combination of hyperparameters is done iteratively. It is a probabilistic approach that uses previous evaluations of the objective function to guide the search for the best hyperparameters. TPE algorithm is used, which stands for Tree-structured Parzen Estimator, and it is a Bayesian optimization method that uses a probabilistic model to explore the hyperparameter space efficiently. It is particularly useful when the hyperparameter space is high-dimensional and difficult to explore using grid search or random search. By using a probabilistic model of the objective function, it is able to focus on promising regions of the hyperparameter space, leading to faster and more efficient optimization.

`

The goal of Bayesian optimization is to minimize the objective function with as few evaluations as possible. This makes it is an effective technique for optimizing expensive black-box functions with high-dimensional input spaces.

The process of Bayesian optimization can be divided into several steps:

1. *Define the search space*: The first step is to define a search space over the hyperparameters. This is typically done by specifying the type and range of each hyperparameter. The number of layers were set between 1-20 since a more complex model was considered impractical for this study. Also, the number of hidden layers were set to either one or two for similar purposes. Not only the number of neurons and hidden layers can be optimized but also the batch size, the activation function, the learning rate, even the optimizer.

2. *Define the objective function*: The objective function is the function that we want to optimize. It takes in the hyperparameters as input and returns a scalar value that we want to maximize or minimize. In this case, the goal is the minimization of the objective function (MSE).

3. *Initialize the model*: Bayesian optimization uses a probabilistic model to represent our belief about the distribution of the objective function. The model is initialized with prior beliefs over the hyperparameters and the objective function.

4. *Select the next set of hyperparameters*: The next set of hyperparameters to evaluate is selected by maximizing an acquisition function. The acquisition function quantifies how promising each point in the search space is based on the current probabilistic model. The acquisition function balances exploration (i.e., selecting hyperparameters that are uncertain) and exploitation (i.e., selecting hyperparameters that are likely to be optimal).

5. *Evaluate the objective function*: The objective function is evaluated at the selected set of hyperparameters.

6. *Update the model*: The data from the evaluation is used to update the probabilistic model over the hyperparameters and the objective function.

7. *Repeat*: Steps 4-6 are repeated until a stopping criterion is met (e.g., a maximum number of iterations is reached or the improvement in the objective function is below a threshold).

In our case however, before proceed to the construction of the ANN models we have to take into account the different behaviours and nature of heave and pitch in relation to other responses such as accelerations and added resistance. In this respect, two different ANN models were built for capturing more accurately the trend of the aforementioned responses. One ANN was built for approximating the RAO heave and RAO pitch and another for approximating the RAO of accelerations and added resistance. This separation was based on the findings of previous chapters where has been illustrated that the nature of these groups of responses tend to be similar. Bayesian optimization technique was used to identify the optimal hyperparameters. It involves iteratively testing different configurations of neural networks to determine the optimal number of neurons for a given task. The process typically begins by selecting a small number of neurons and training the network on a dataset. The performance of the network is then evaluated, and the number of neurons is gradually increased. This process is repeated until the error rate of the network stabilizes and stops improving, indicating that further increases in the number of neurons are unlikely to improve performance. The MSE was used to determine the best performing algorithms separately for both groups of responses, as follows:

`

One hidden layer networks						
Nadam			**Nadam**			
	MSE (RAOheave)	MSE (RAOpitch)	MSE (RAOaccb)	MSE (RAOaccm)	MSE (RAOaccs)	MSE (RAOadded)
Fn = 0.34	0.000596	0.001488	0.8799	0.2511	0.2661	0.3153
Fn=0.68	0.00095	0.00232	1.1031	0.3271	0.3943	0.3908
SGDmom_nest			**SGDmom_nest**			
Fn = 0.34	0.0002706	0.00090	0.8559	0.3156	0.4432	0.3363
Fn=0.68	0.00046	0.0013	0.9787	0.4951	0.6677	0.4123
SGDmom			**SGDmom**			
Fn = 0.34	0.0004133	0.000965	1.1523	0.3328	0.3214	0.4115
Fn=0.68	0.000385	0.00118	1.0589	0.4547	0.4920	0.5278

Table 9: Performance of chosen networks with one hidden layer

In order to obtain the optimal performance of the network in predicting accelerations and added resistance responses, two-layer networks were also tested. On the other hand, for heave and pitch responses, a single hidden layer was found to be pretty good, as these responses are simpler in nature and have been shown to be well-approximated using only one hidden layer.

Two hidden layer networks				
Adam				
Fn	MSE (RAOaccb)	MSE (RAOaccm)	MSE (RAOaccs)	MSE (RAOadded)
0.34	0.8965	0.4898	0.3459	0.3386
0.68	1.1459	0.5841	0.6203	0.4181
	SGDmom_nest			
0.34	1.059	0.3156	0.3839	0.3153
0.68	1.9573	0.4047	0.7008	0.3897
	SGDmom			
0.34	0.9059	0.6196	0.4879	0.3185
0.68	1.8907	0.8047	0.7988	0.4835

Table 10: Performance of chosen networks with two hidden layers for accelerations and added resistance

Overall, it is observed that one layer network achieves better performance for the subject dataset and this configuration is to be used for the following analysis.

`

This process of evaluation with MSE, can be visualized by obtaining the history of the error function during the training process of an artificial neural network (ANN), we can visualize valuable insights that can help us understand how the network is learning and improving over time. This information can be used to fine-tune the parameters of the network, such as the learning rate and the number of hidden layers, in order to improve its performance on the task at hand. The best performing algorithms are represented below by their 'history' objects.

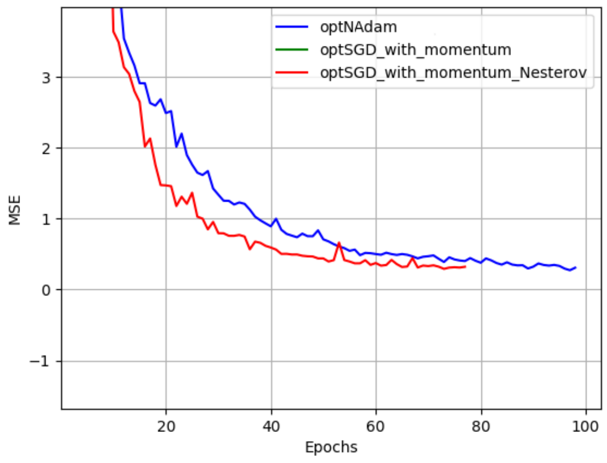


Figure 45: Cost function minimization through epochs (Fn=0.34) of RAO accelerations and added resistance

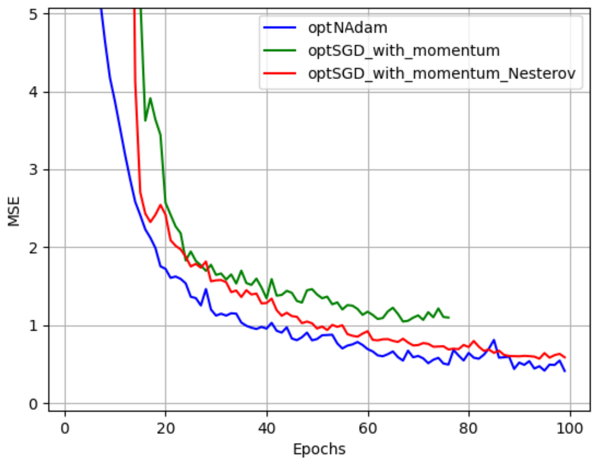


Figure 46: Cost function minimization through epochs (Fn=0.68) of RAO accelerations and added resistance

We can observe at the left Nadam (Adam with Nesterov algorithm) and SGD with Momentum and Nesterov to struggle to minimize the MSE and Adam finally wins the long race. Similarly, at the right-hand side Nadam captures the best MSE score at final epochs for accelerations and added resistance.
In order to minimize the mean squared error (MSE) for the heave and pitch responses, stochastic gradient descent (SGD) with momentum and Nesterov was employed. All algorithms seem to be effective in the for heave and pitch responses, in terms of minimizing the error function effectively. SGD with Momentum and Nesterov was chosen since it was configurated to have fewer parameters which makes it a simpler model.

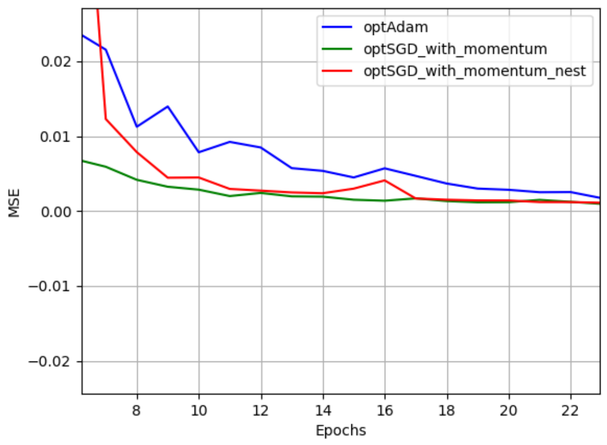


Figure 47: Cost function minimization through epochs (Fn=0.68) of RAO heave and pitch responses

`

To evaluate the performance of the models, cross-validation is typically used to split the dataset into training and testing sets. The model is then trained on the training set and tested on the testing set, and if the predictions match the actual labels quite well for both the training and testing sets, the model can be considered a good predictor.

In order to quantify the performance with regression metrics we plot the actual results from experiments against the predictions of the neural networks for both training and testing datasets. The plots below are used to evaluate the performance of the ANN models, separately for both different models, predicting the RAO heave and RAO pitch and RAO accelerations and added resistance respectively. The former refers to simpler model, while the latter needs a more complex configuration in order to be approximated accurately. For each ANN model, a comparison in regression metrics was made for training and testing sets in order to identify the prediction capabilities of the networks.

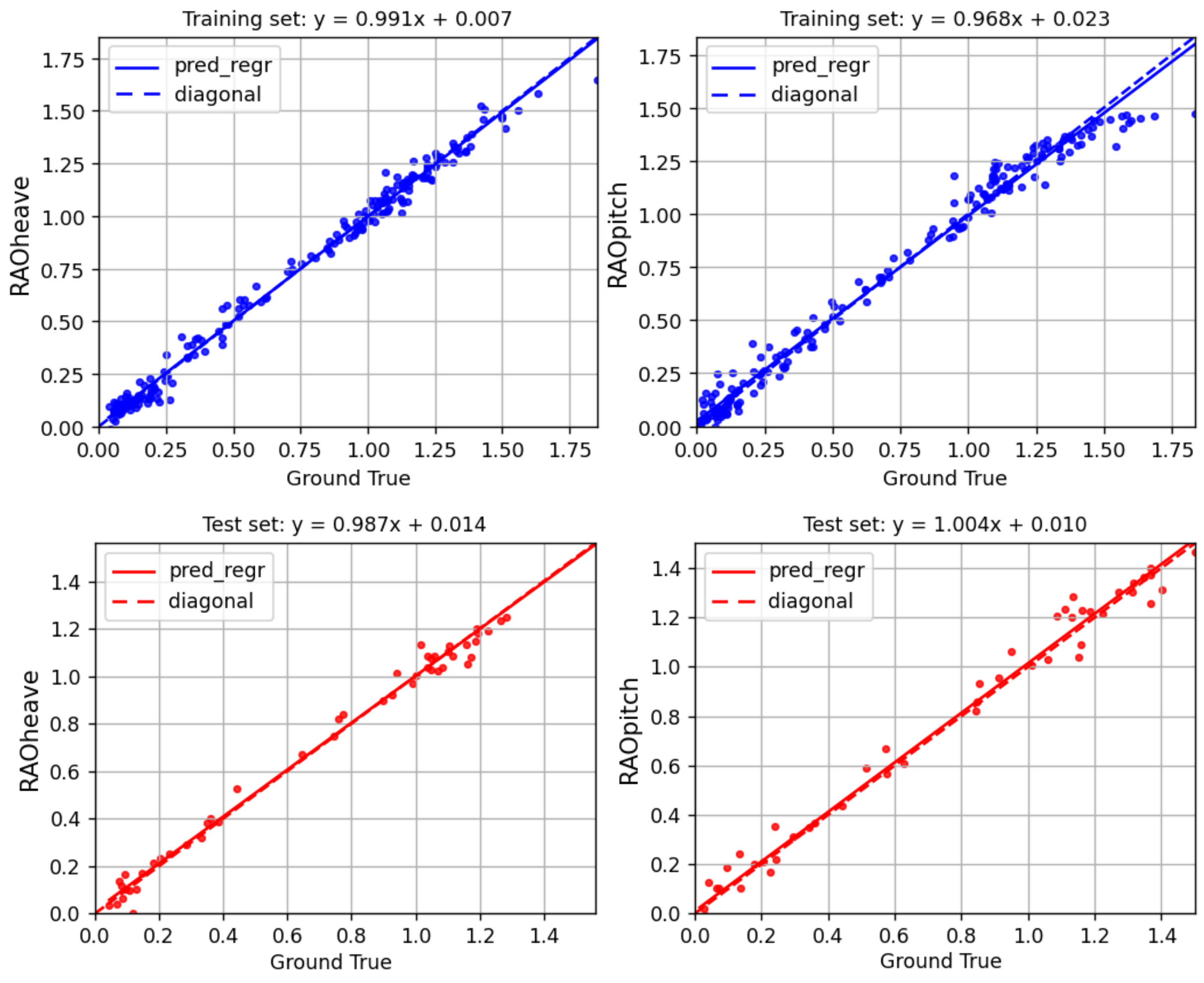


Figure 48: Performance of the model in training and testing sets (Fn=0.34) for heave and pitch responses

`

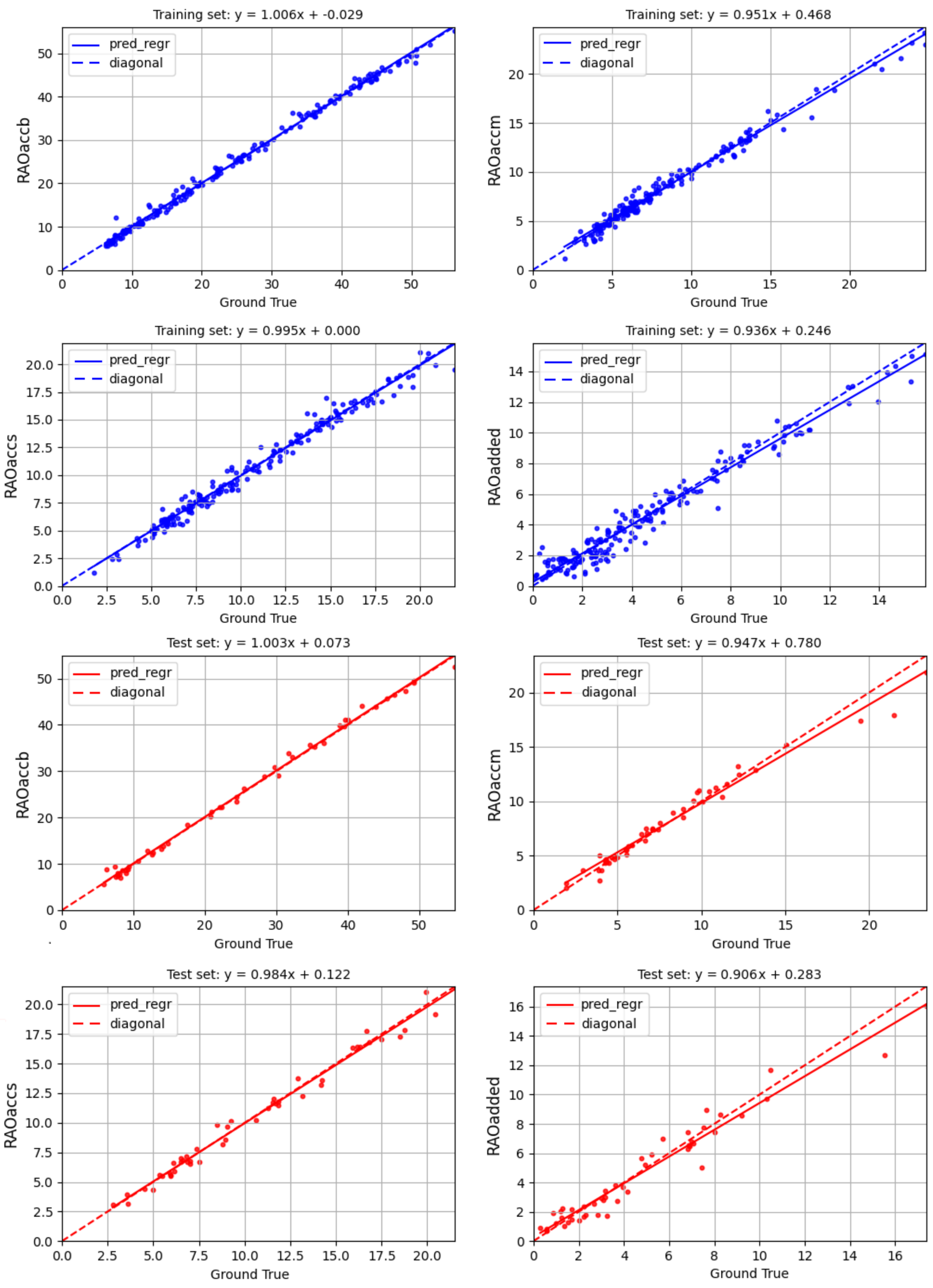


Figure 49: Performance of the model in training test and testing sets (Fn=0.34) for RAO accelerations and added resistance responses

`

The above figures show the performance of a neural network that was trained using the Nadam backpropagation algorithm with regression metrics. The networks' generalization properties are demonstrated by evaluating its performance over a testing set. The regression value, or slope of the curve, is an indicator of the relationship between the network's outputs and the target outputs, where a value of one corresponds to an exact linear relationship (i.e., $y = x$). A perfect prediction would result in a regression value of 1 and an intercept of 0, which would correspond to the diagonal line y=x on a scatterplot. The obtained regression metrics for heave, pitch, and accelerations were found to be very good, while the regression for added resistance was slightly worse. Similar regression metrics were obtained for Fn=0.68, but for convenience they were not presented.

It is worth noting that the Artificial Neural Network (ANN) model achieved similar levels of performance for both the training and testing datasets, which suggests that the model is reliable and stable. This means that the model is able to generalize well to new data and is not overfitting to the training data, which is an important characteristic of a well-performing model.

Below, the performance of the Series is examined for both speeds of Fn=0.34 and Fn = 0.68 and the results are illustrated.

ANN model performance for Fn=0.34

Since we have obtained the performance of the model separately for training and testing datasets with statistics, the overall performance of the model should be also examined. In this respect, we will plot the model predictions for both training and testing dataset along with actual experimental value for every Series model. This will allow us to visualize how well the model performs across the entire dataset.

The predictions of the accelerations, added resistance, heave and pitch responses for the full grid of experimental results for both Froude numbers are represented. Additionally, a 4th order polynomial was utilized to fit the datapoints of experimental results in order to get a quick approximation of RAOs of every response. From its nature, 4rth order polynomial has limitations in capturing the peaks and valleys, but it can provide a smooth representation of the datasets. On the other hand, Artificial Neural Networks (ANNs) are highly flexible and can accurately capture the underlying relationships in the data with any shape.

`

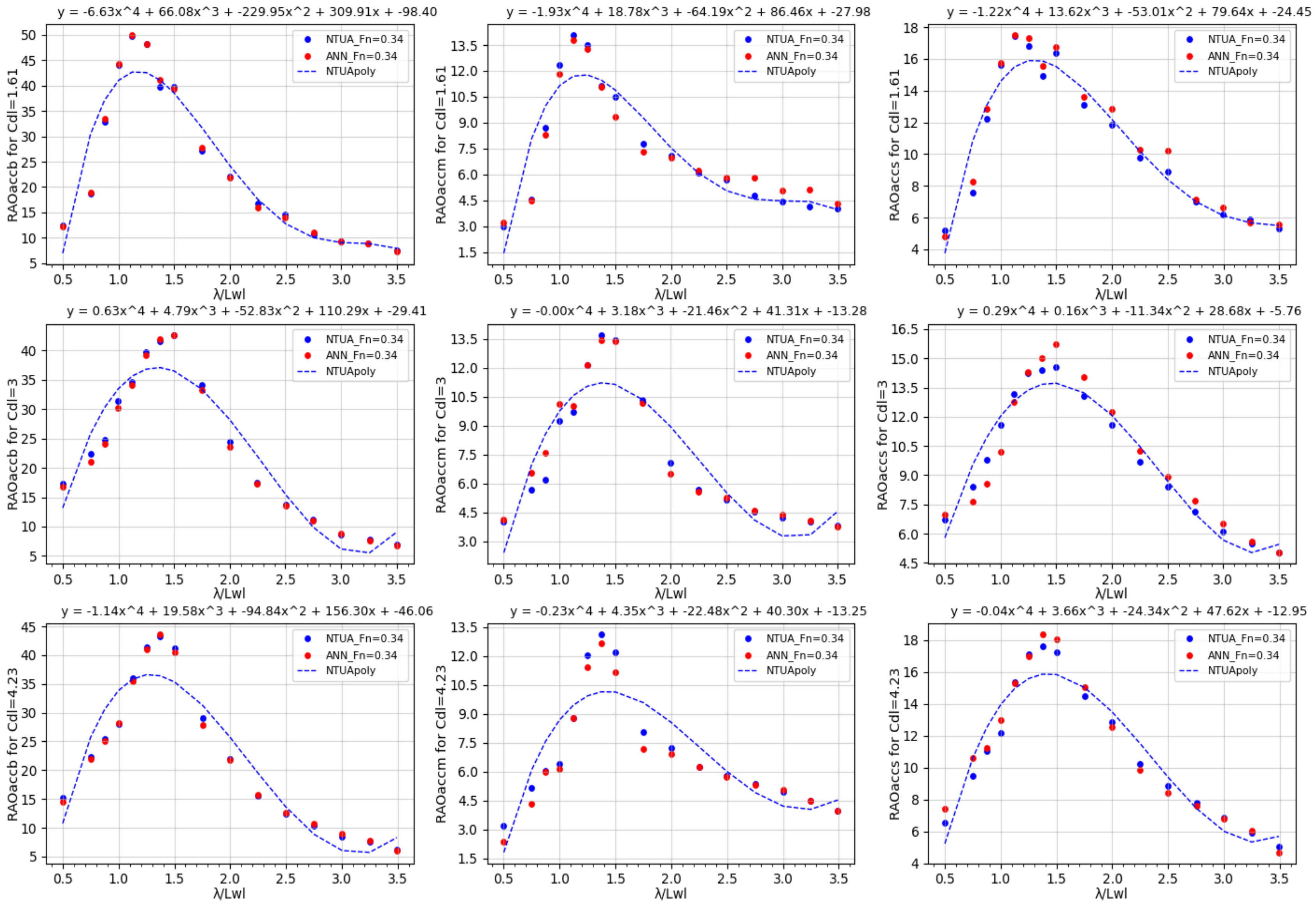


Figure 50: RAO *curves* for accelerations (L/B=4.00, Fn = 0.34)

`

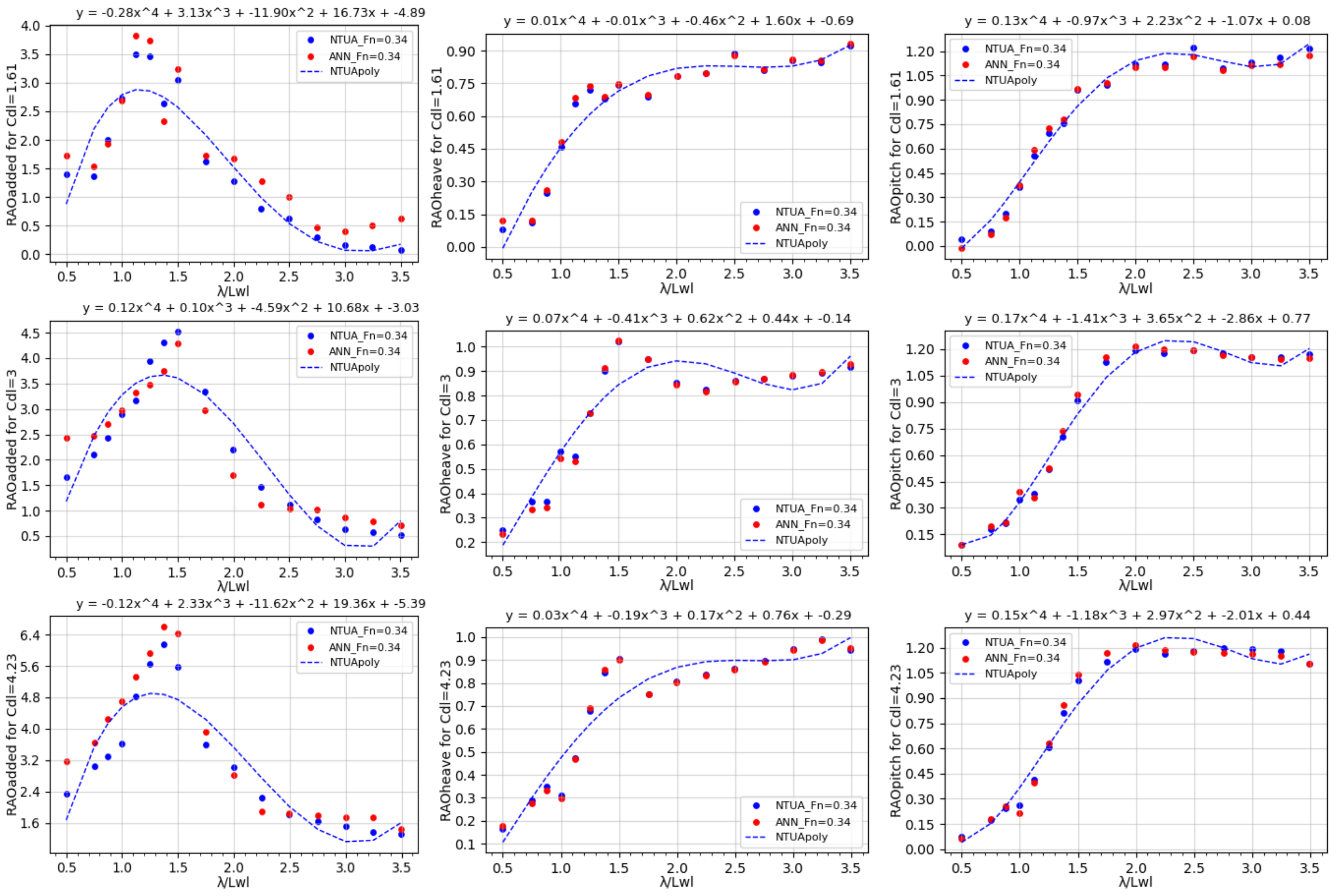


Figure 51: RAO *curves* for heave, pitch and added resistance (L/B=4.00, Fn = 0.34)

`

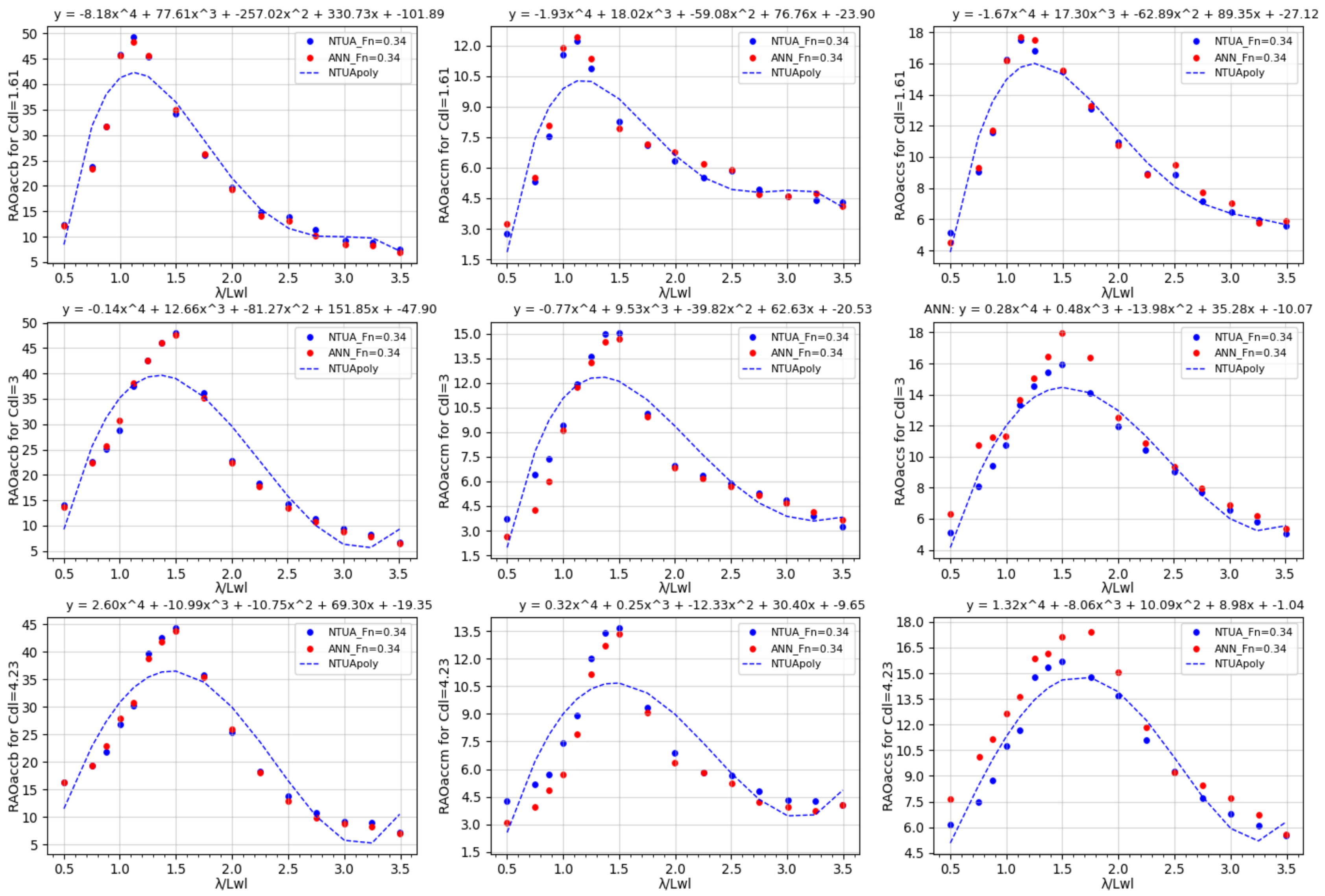


Figure 52: RAO *curves* for accelerations (L/B=4.75, Fn = 0.34)

`

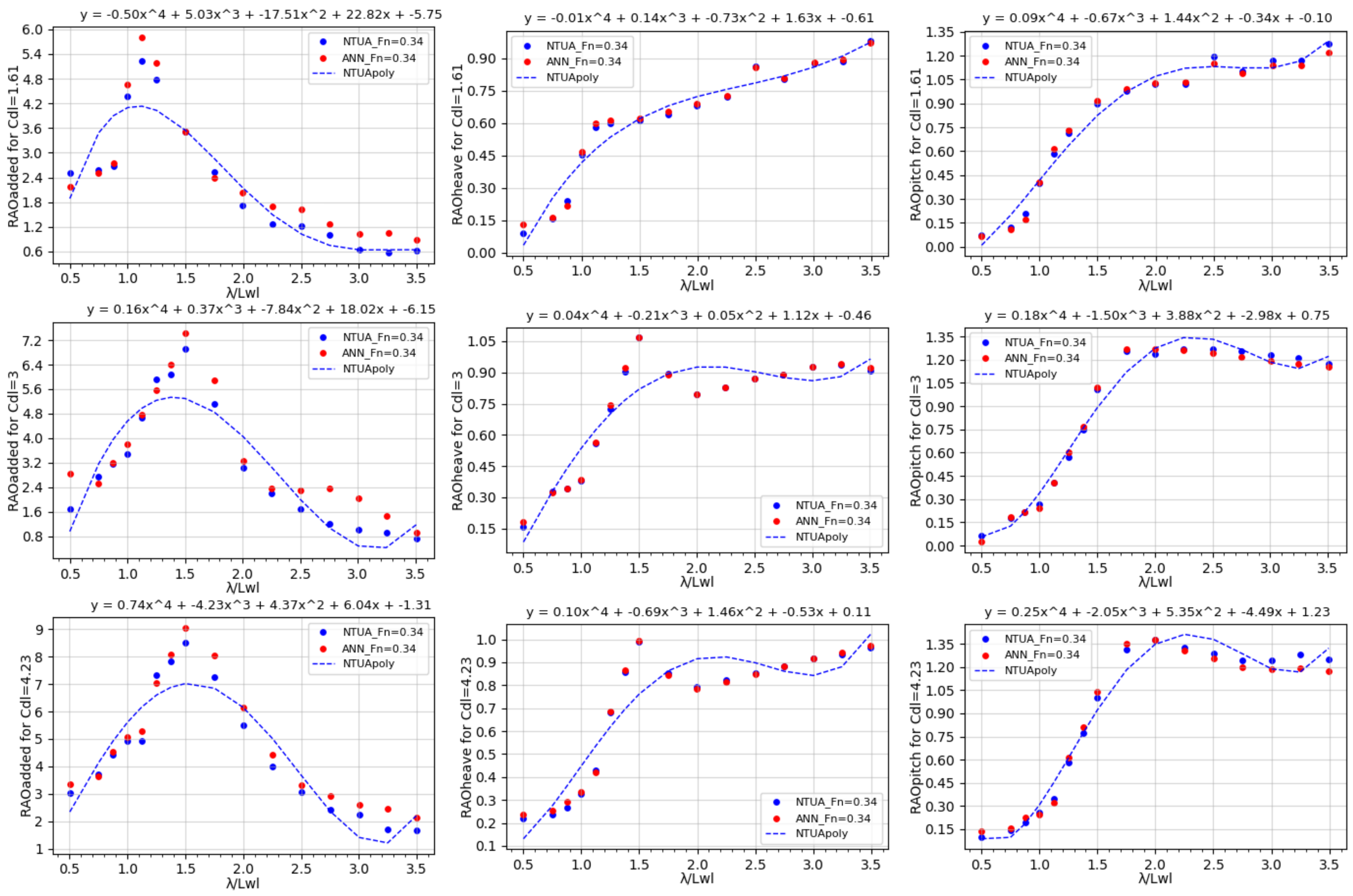


Figure 53: RAO *curves* for heave, pitch and added resistance (L/B=4.75, Fn = 0.34)

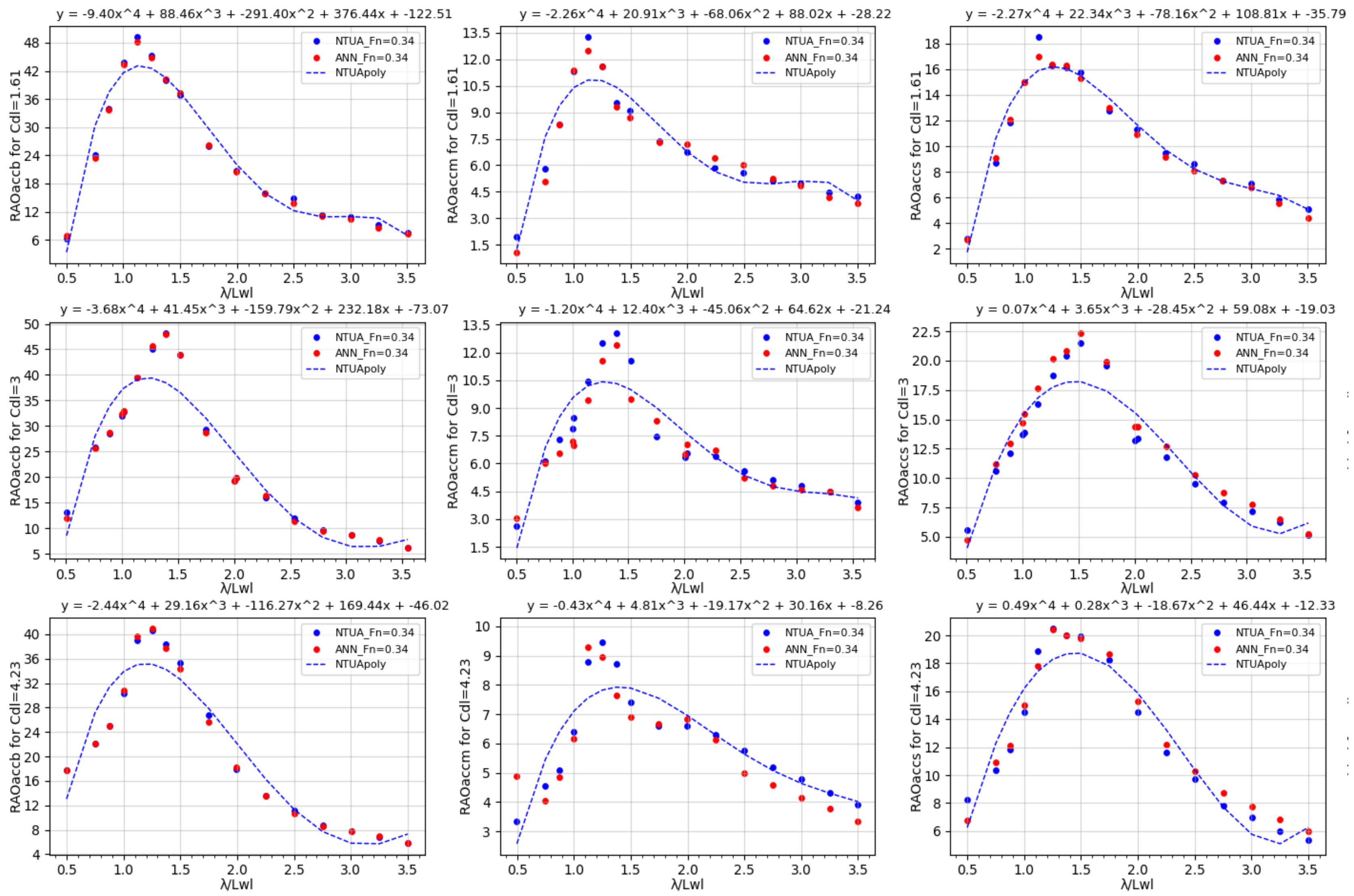


Figure 54: RAO *curves* for accelerations (L/B=5.50, Fn = 0.34)

`

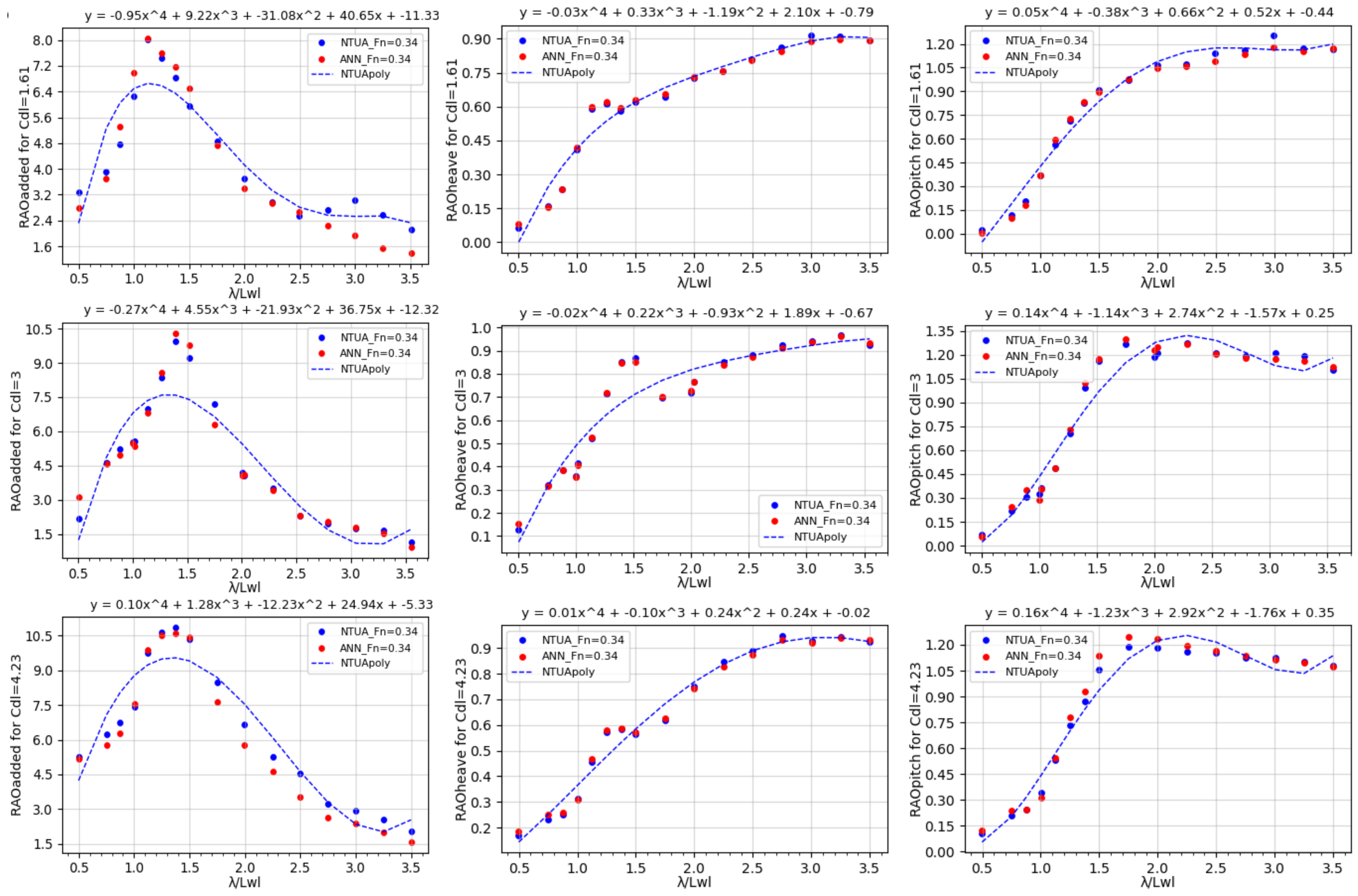


Figure 55: RAO *curves* for heave, pitch and added resistance (L/B=5.50, Fn = 0.34)

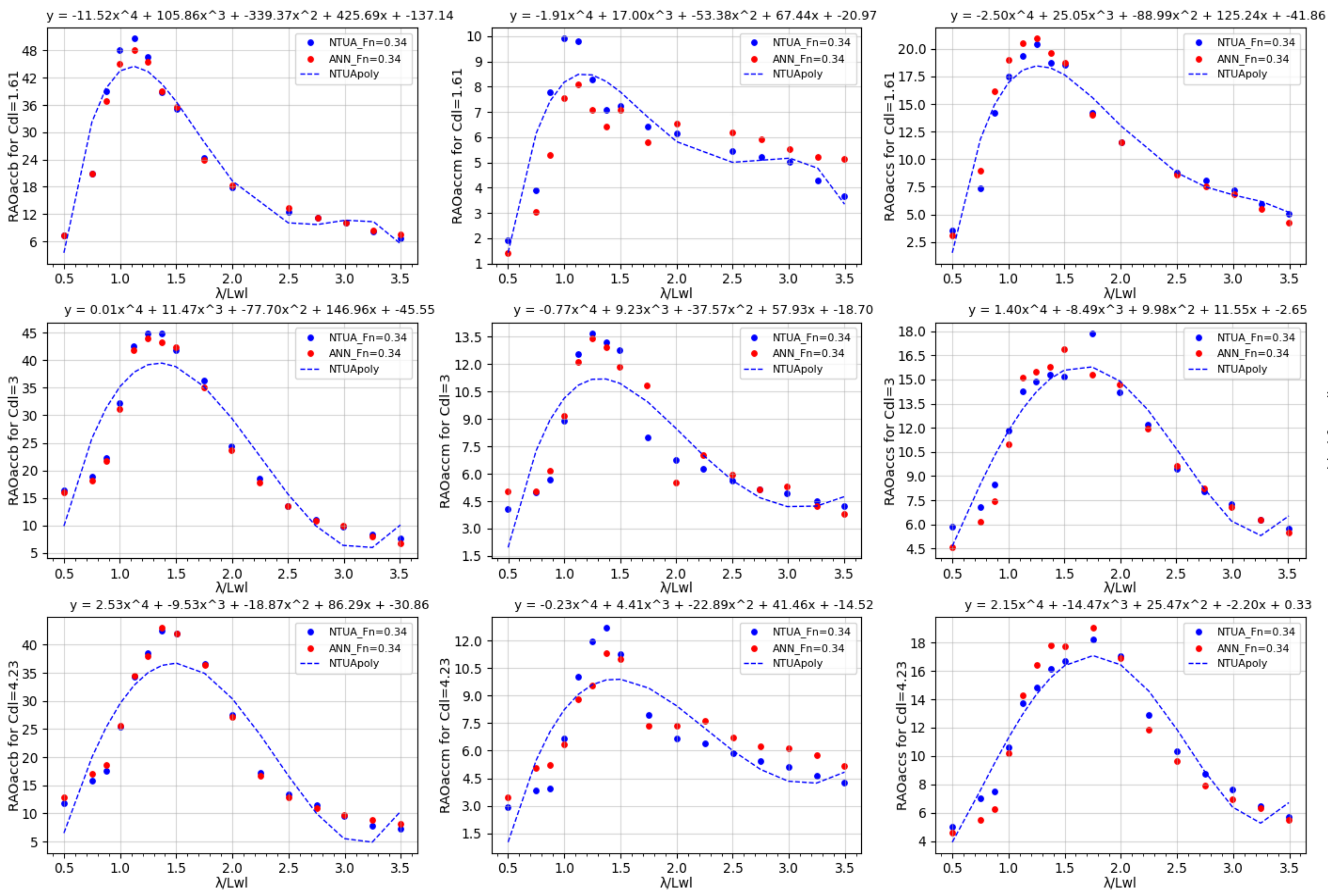


Figure 56: RAO *curves* for accelerations (L/B=6.25, Fn = 0.34)

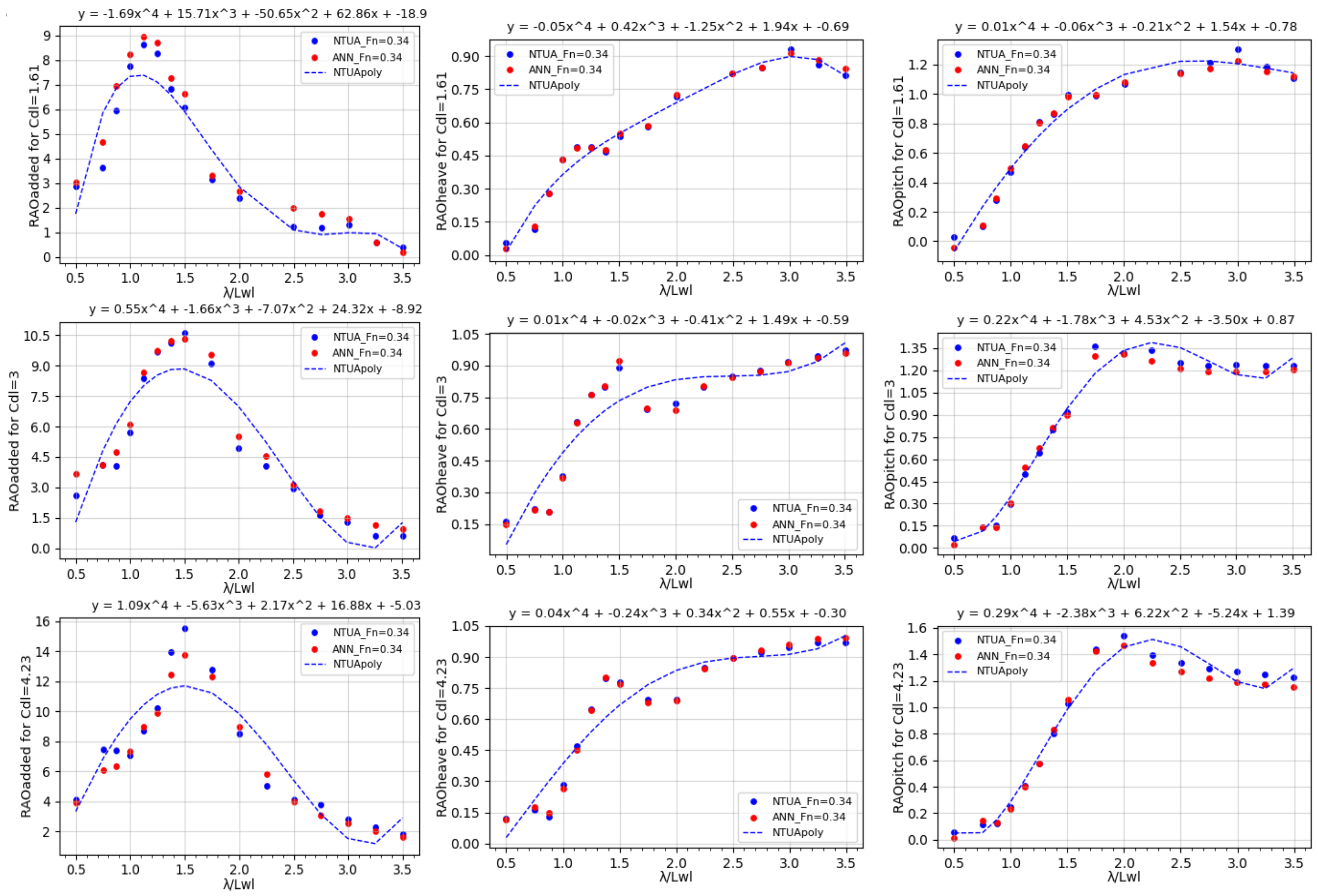


Figure 57: RAO *curves* for heave, pitch and added resistance (L/B=6.25, Fn = 0.34)

`

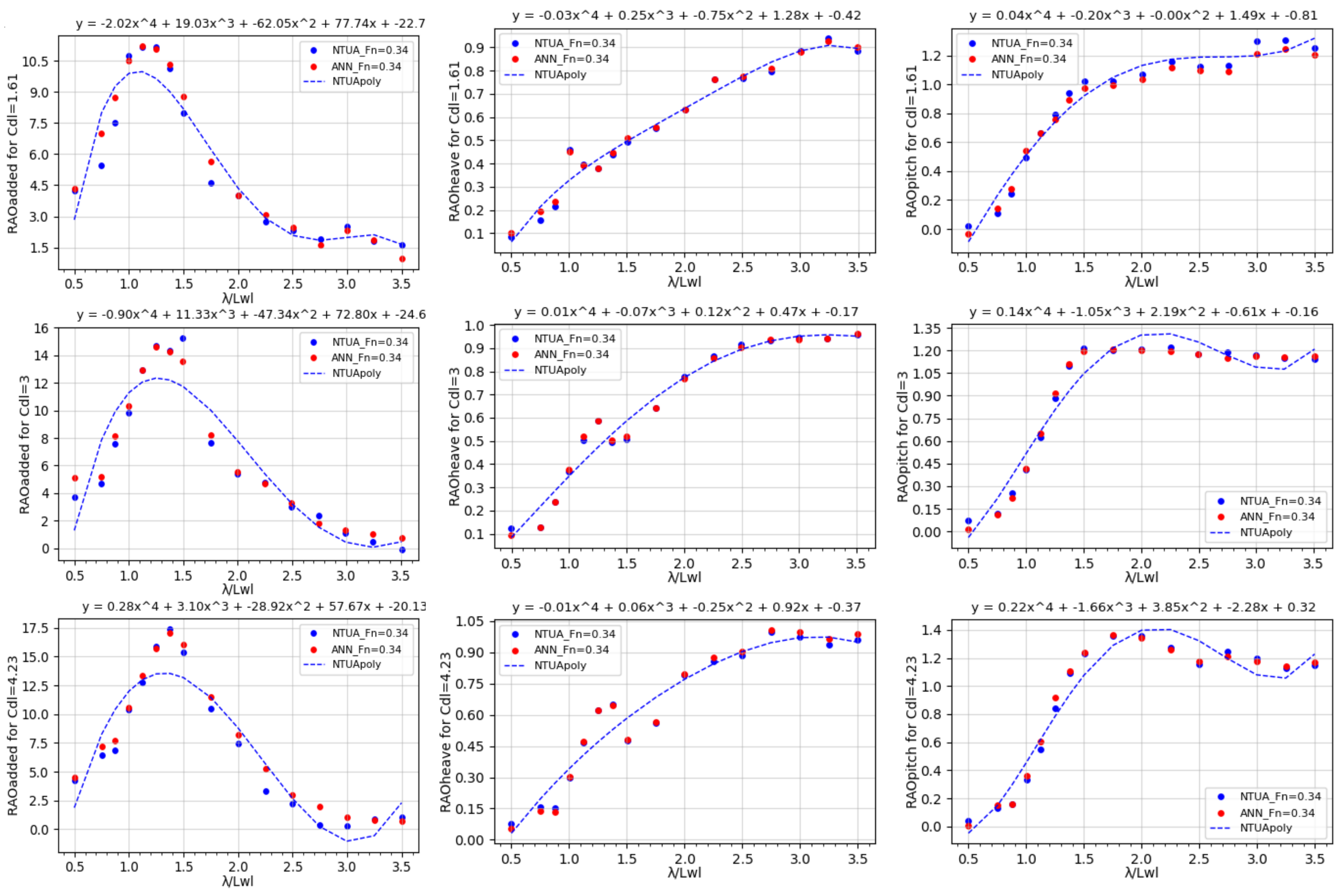


Figure 58: RAO *curves* for heave, pitch and added resistance (L/B=7.00, Fn = 0.34)

`

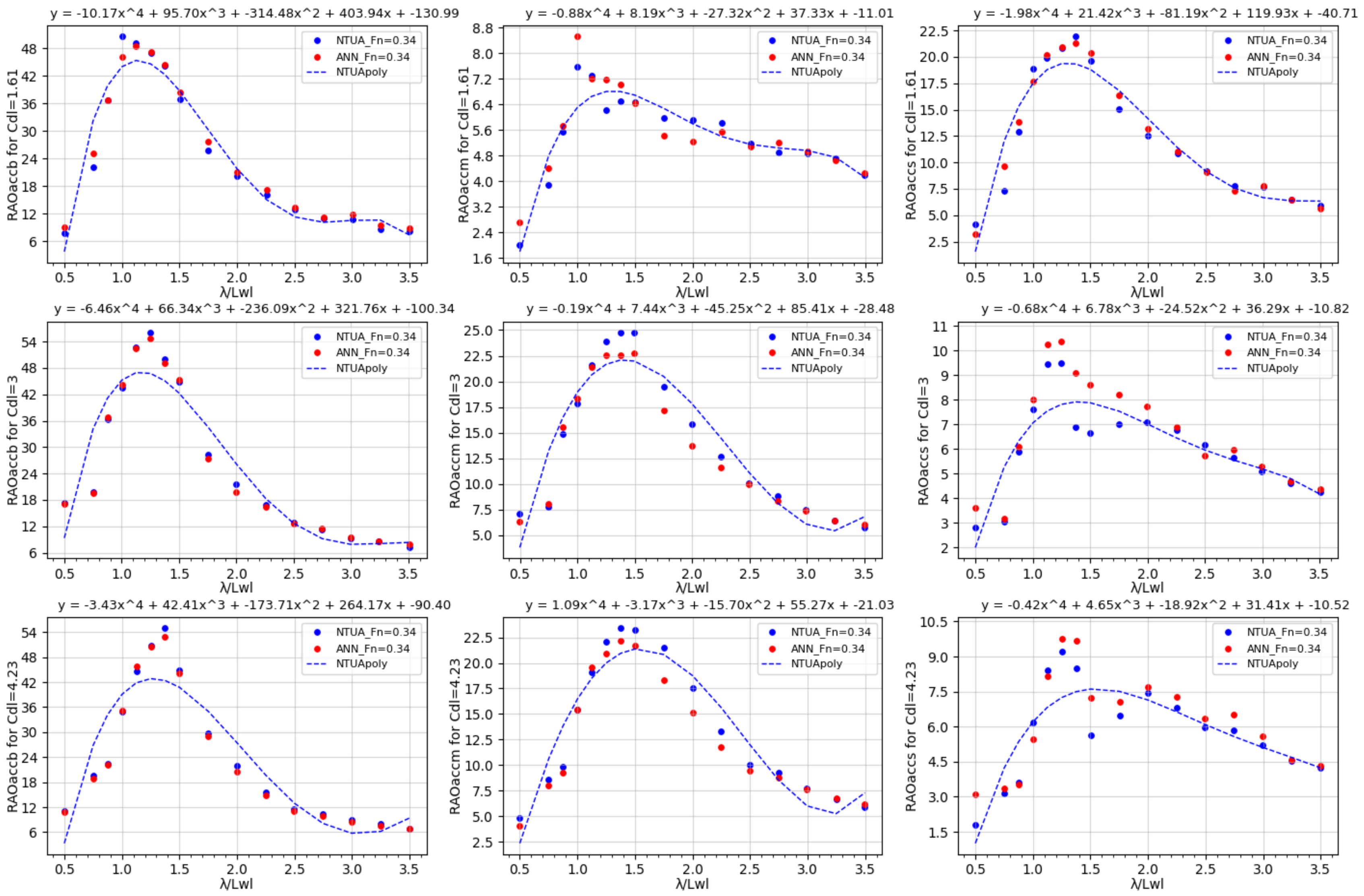


Figure 59: RAO *curves* for accelerations (L/B=7.00, Fn = 0.34)

`

Based on the grid of figures presented, it is evident that a simple Artificial Neural Network (ANN) can accurately predict the heave and pitch responses. Furthermore, it should be noted that for these responses, a 4th order polynomial function fits the points well and captures the underlying trend accurately. With regards to the Response Amplitude Operators (RAOs) for accelerations, the ANN can predict them with high accuracy and low variance, especially for bow and mid accelerations. As for the added resistance, the ANN can provide a good approximation, but it does not achieve such high level of accuracy compared to other responses. That is reasonable if someone considers the nature of added resistance which is a second order variable with respect to wave height and is more sensitive to changes in the input parameters compared to the first-order variables which correspond to the rest of the responses. Also, as mentioned previously, the 4th order polynomial function is a cursory approximation method that does not capture the peaks and valleys of the data with high accuracy. It is although considered satisfactory for a quick overview of the responses.

ANN model performance for Fn=0.68

This chapter presents the experimental results for a Fn=0.68, along with the predictions of an Artificial Neural Network (ANN) for both the training and testing sets, and a 4th order polynomial used to roughly approximate the RAO curves. The complete grid of experimental results for all the responses is presented. It should be noted that the last data point for accelerations and added resistance was omitted from the polynomial approximation for this particular Fn, as it was observed to alter the curvature at the end of the curve resulting in deviation from the expected pattern observed in the rest of the data set. Therefore, it was considered inconsistent and was excluded from the polynomial approximation.

`

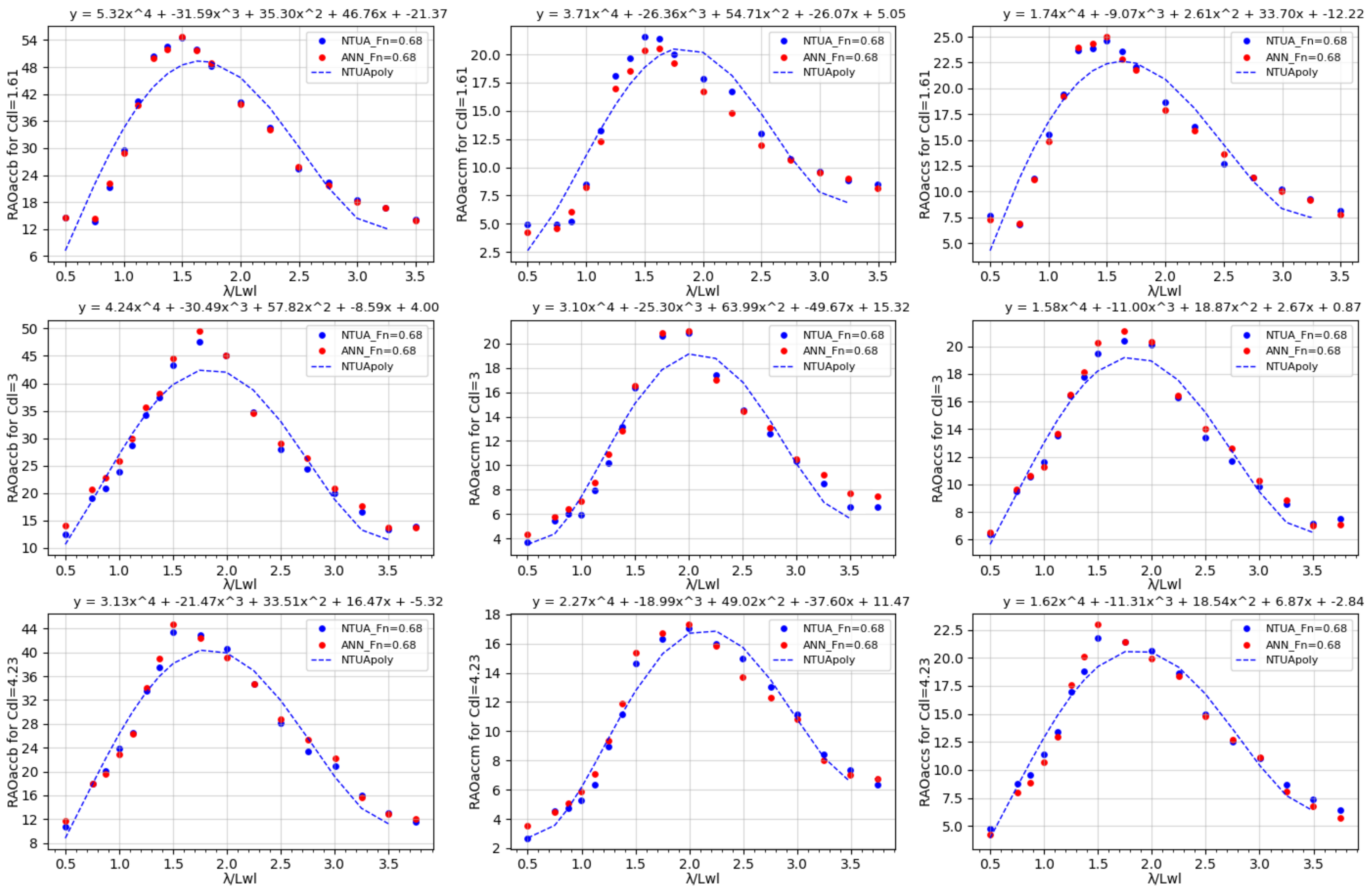


Figure 60: RAO *curves* for accelerations (L/B=4.00, Fn = 0.68)

`

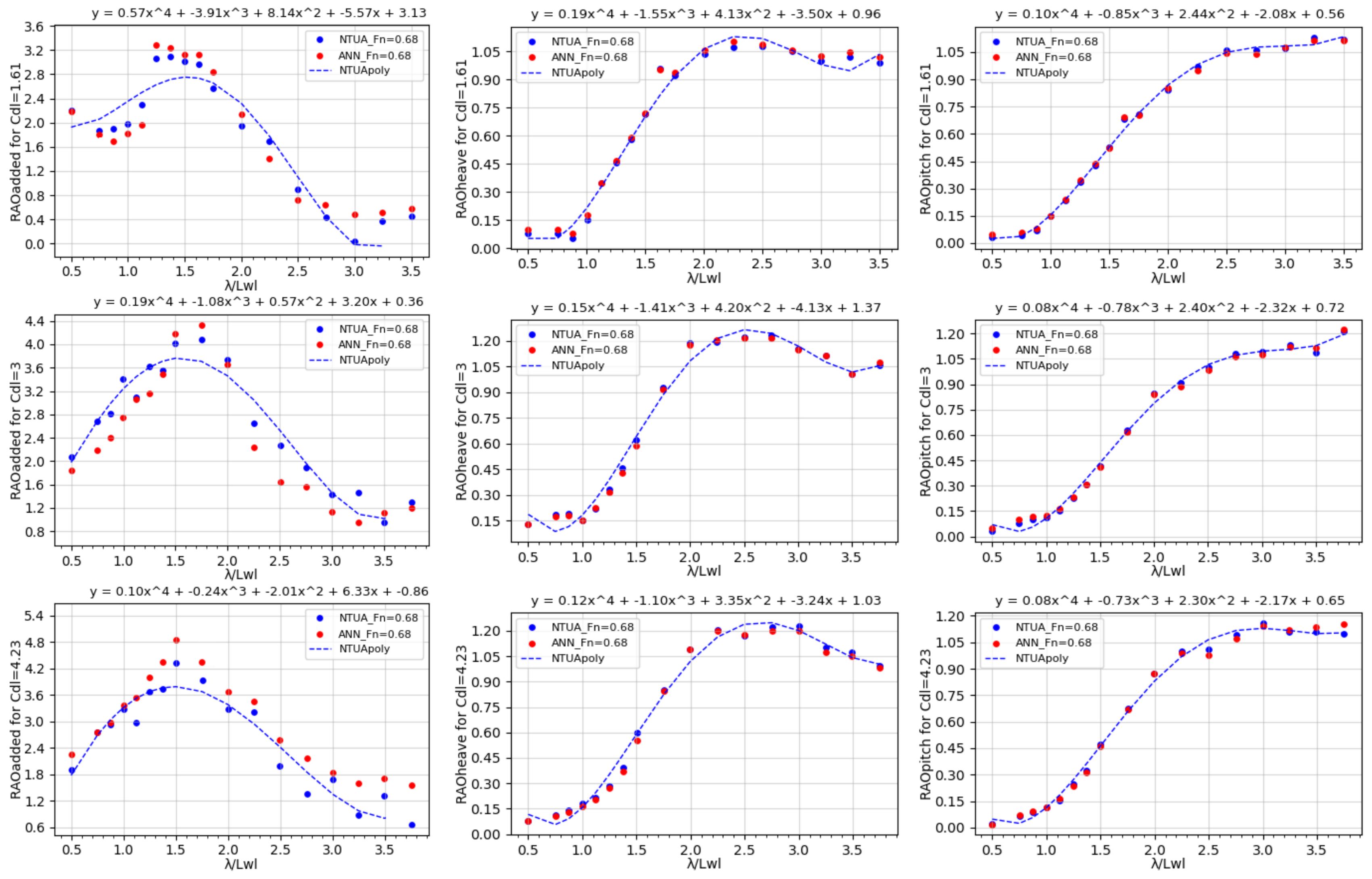


Figure 61: RAO *curves* for heave, pitch and added resistance (L/B=4.00, Fn=0.68)

`

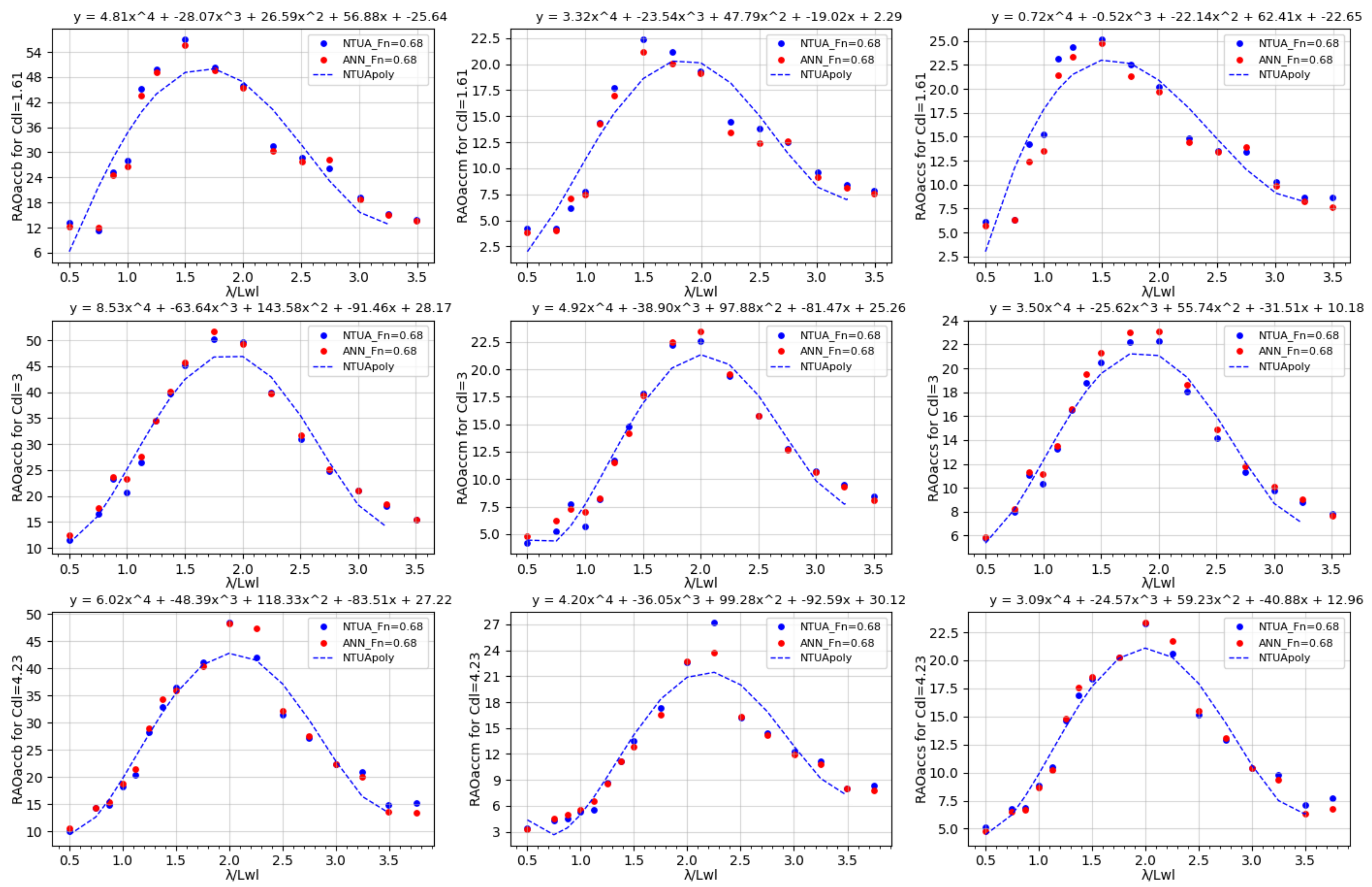


Figure 62: RAO *curves* for accelerations (L/B=4.75, Fn = 0.68)

`

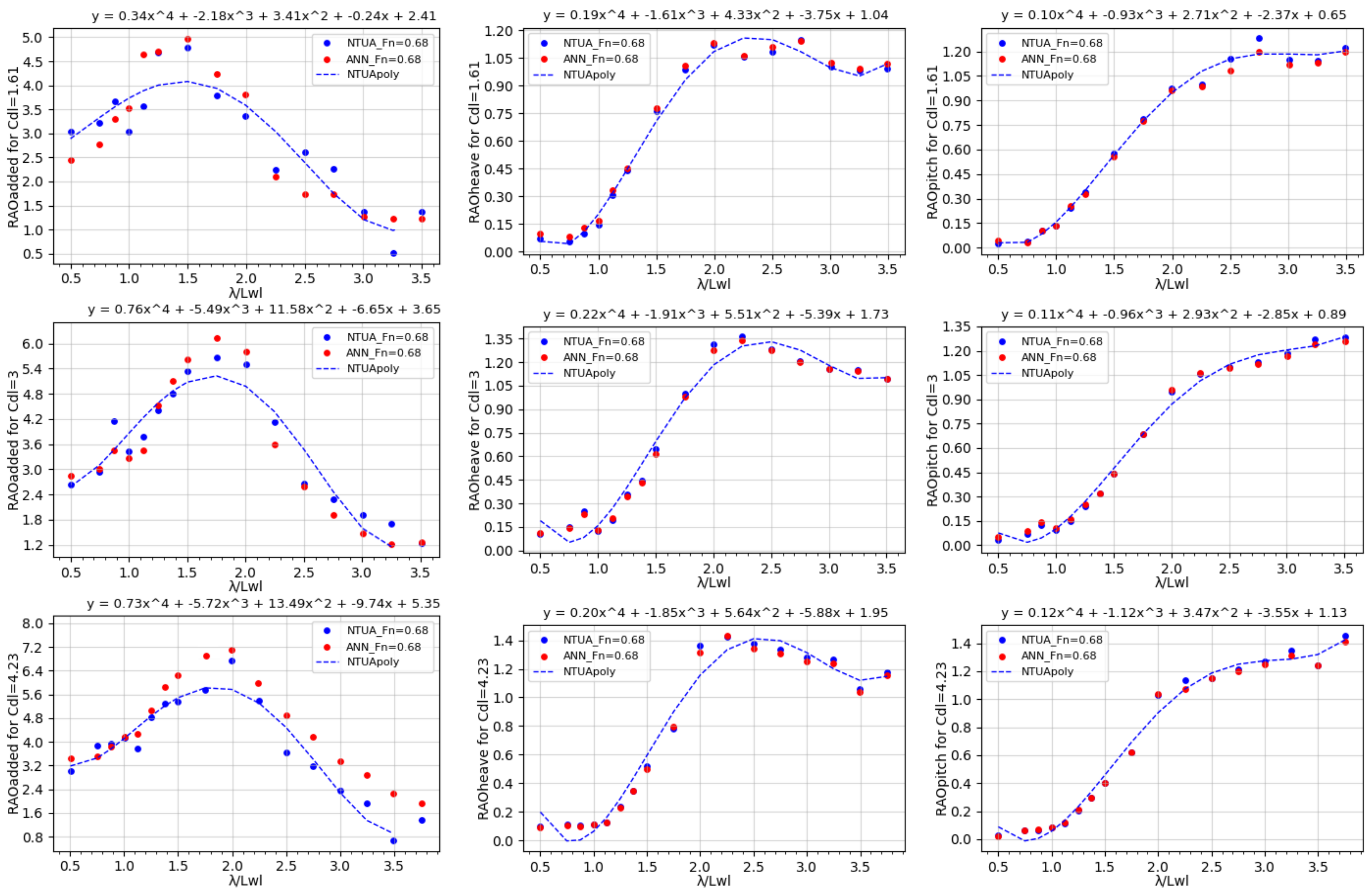


Figure 63: RAO *curves* for heave, pitch and added resistance (L/B=4.75, Fn = 0.68)

`

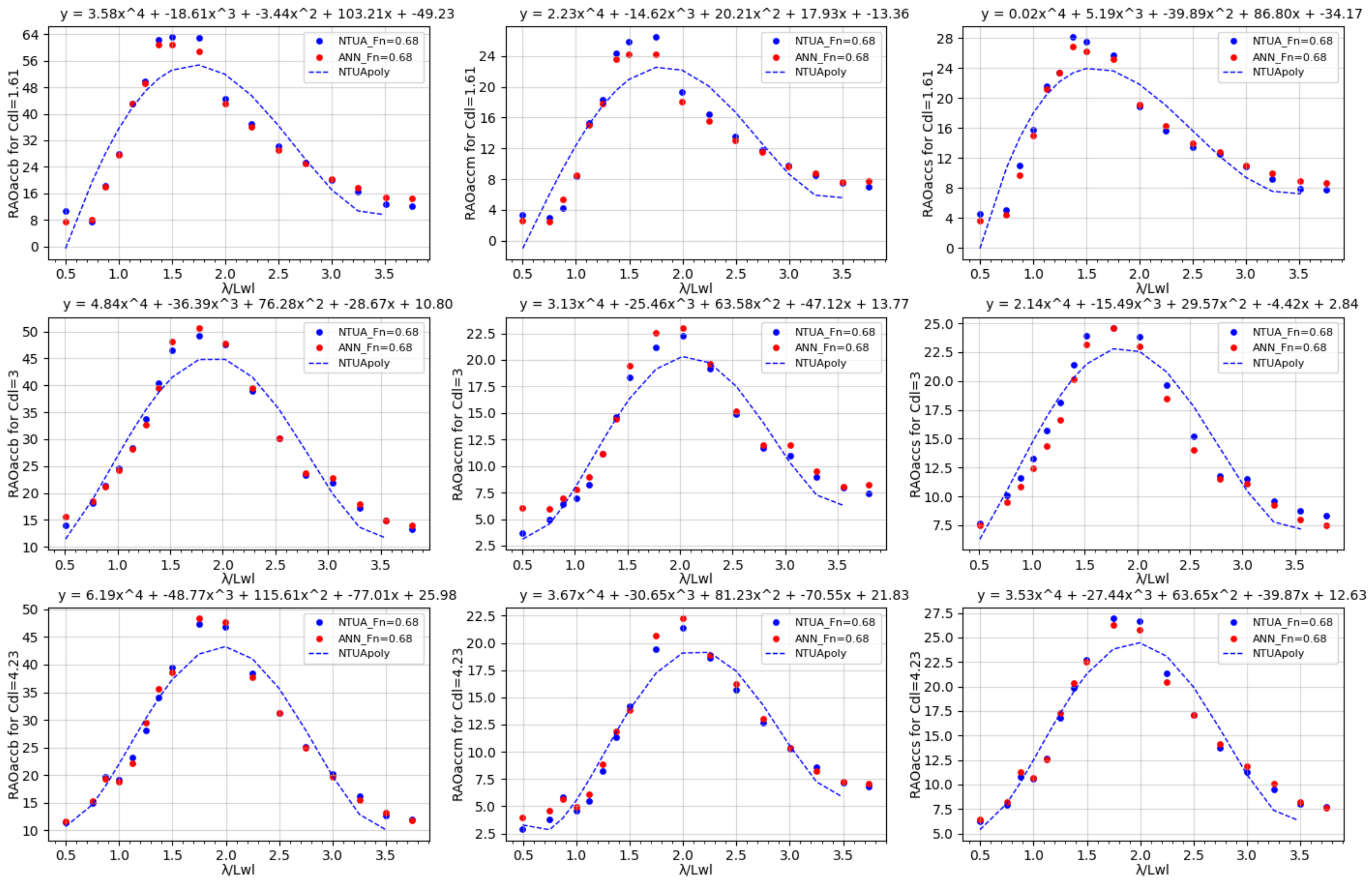


Figure 64: RAO *curves* for accelerations (L/B=5.50, Fn = 0.68)

`

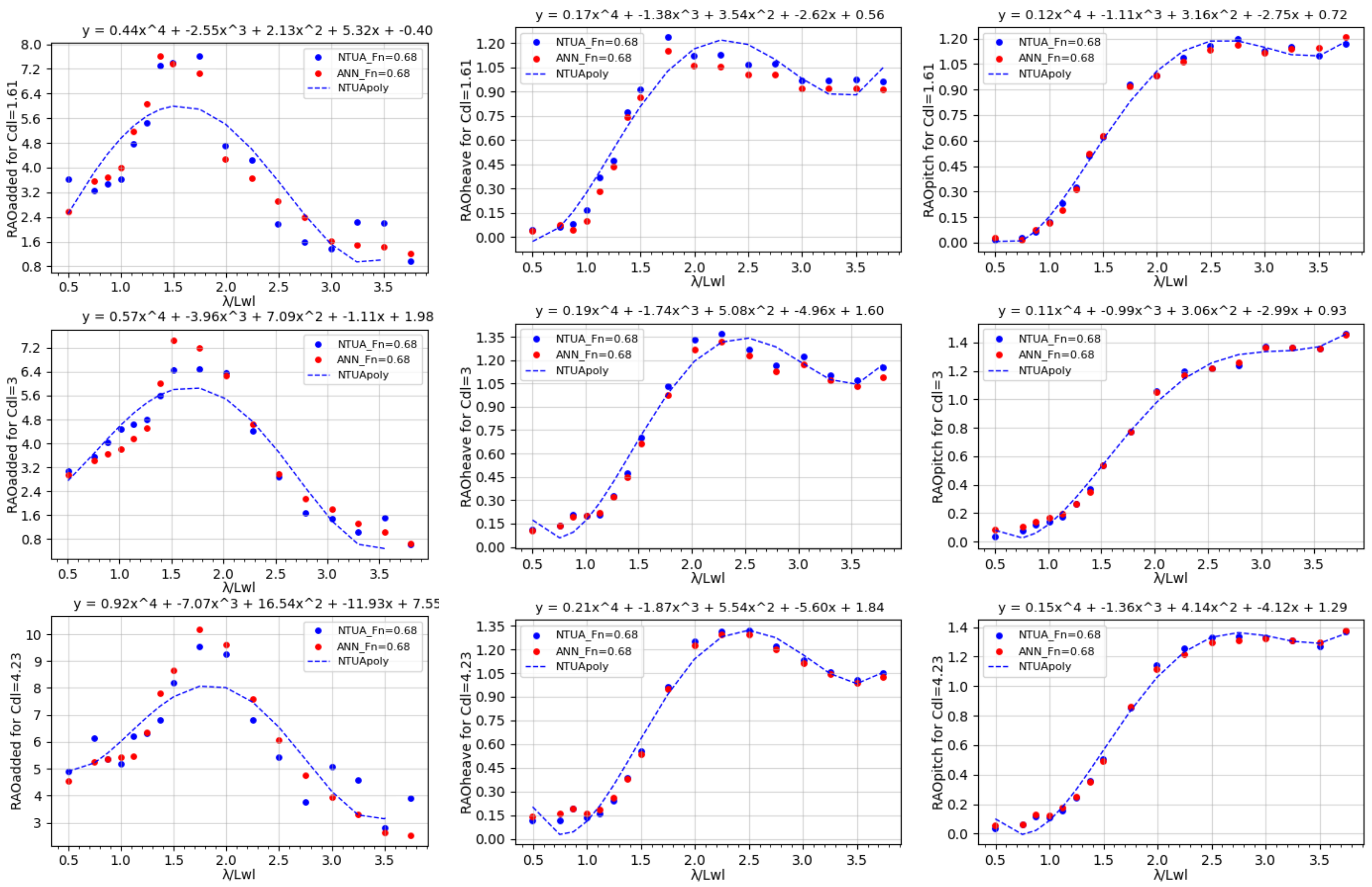


Figure 65: RAO *curves* for heave, pitch and added resistance (L/B=5.50, Fn = 0.68)

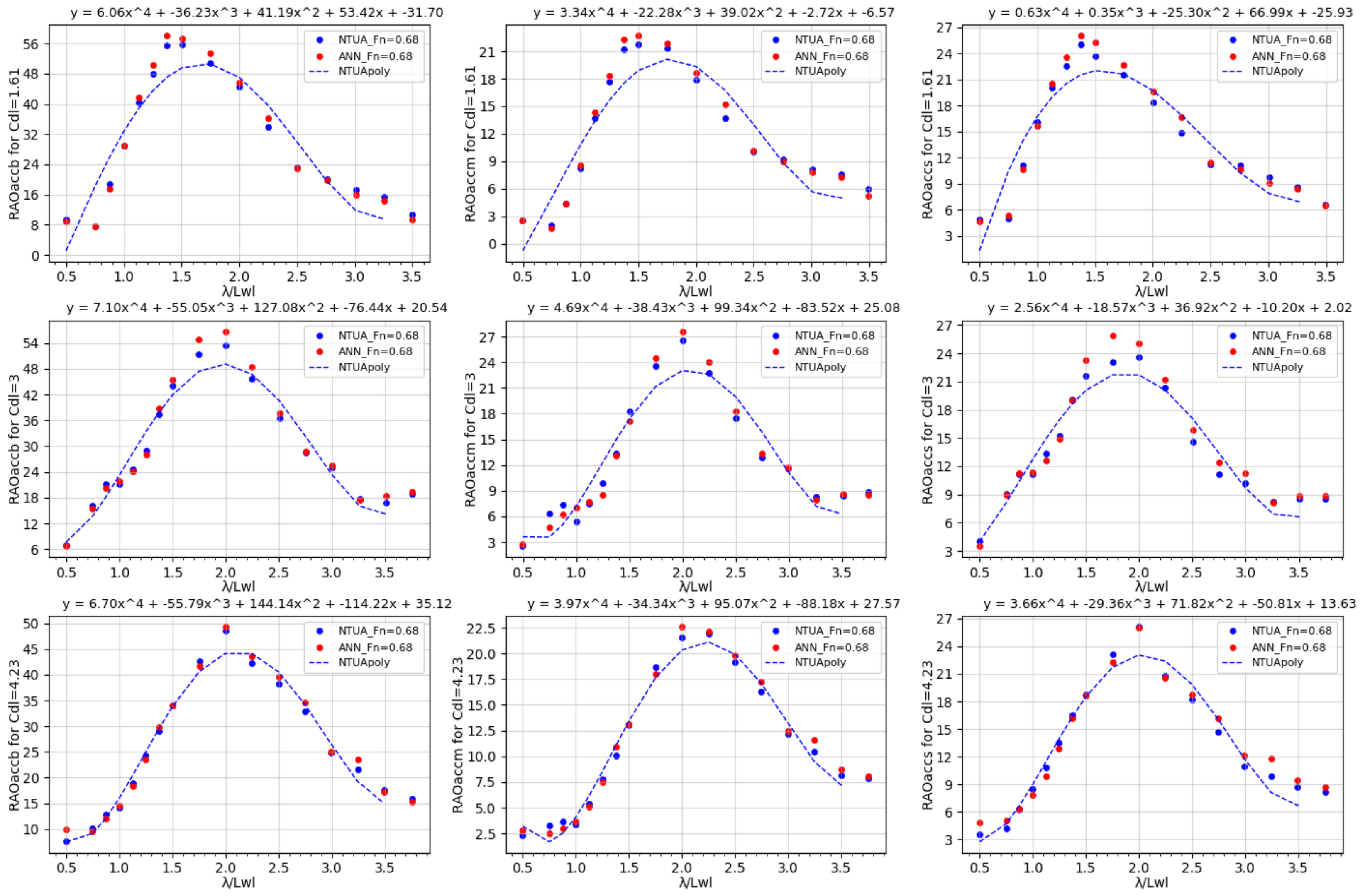


Figure 66: RAO *curves* for accelerations and added resistance (L/B=6.25, Fn = 0.68)

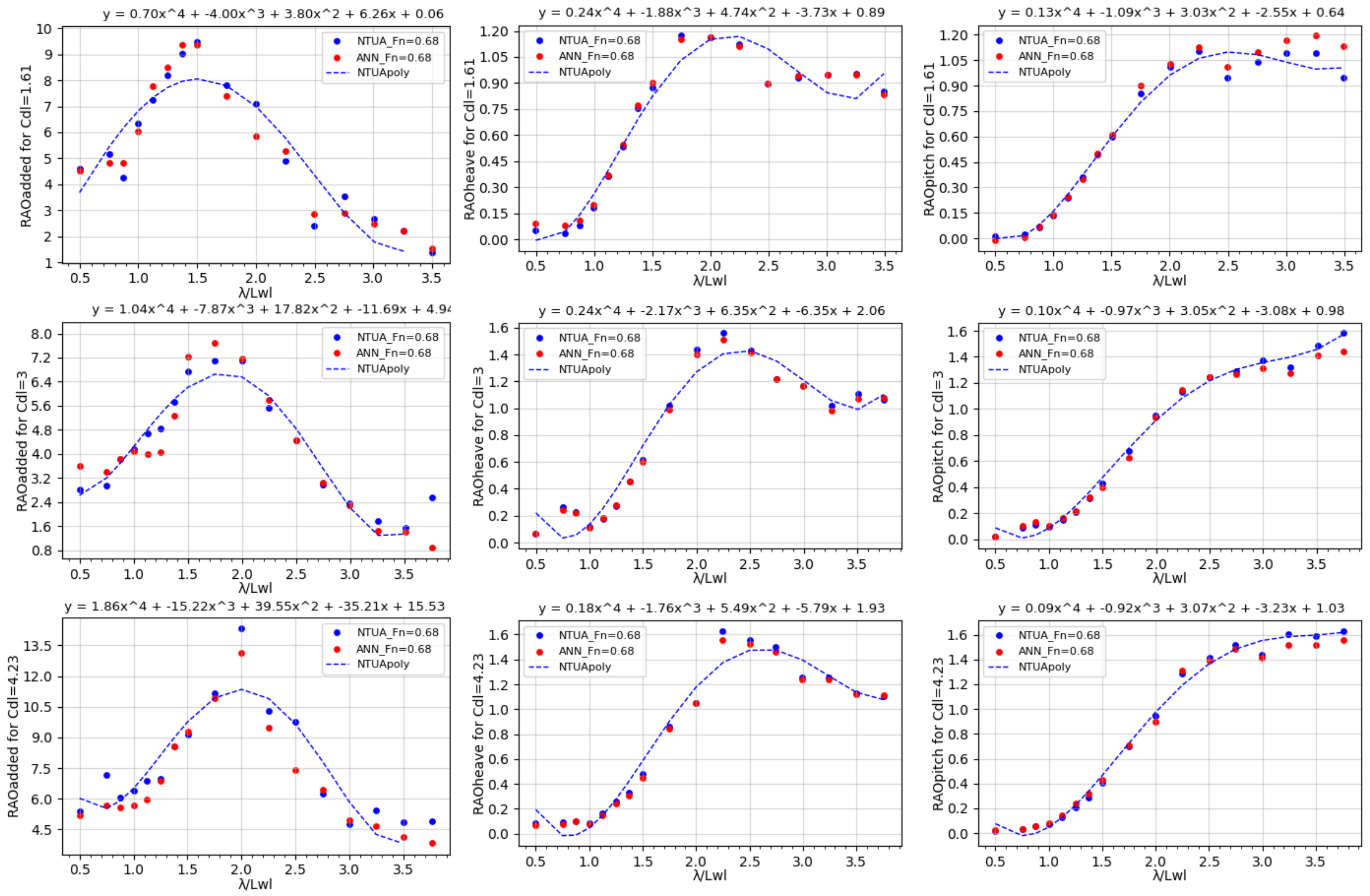


Figure 67: RAO *curves* for heave, pitch and added resistance (L/B=6.25, Fn = 0.68)

`

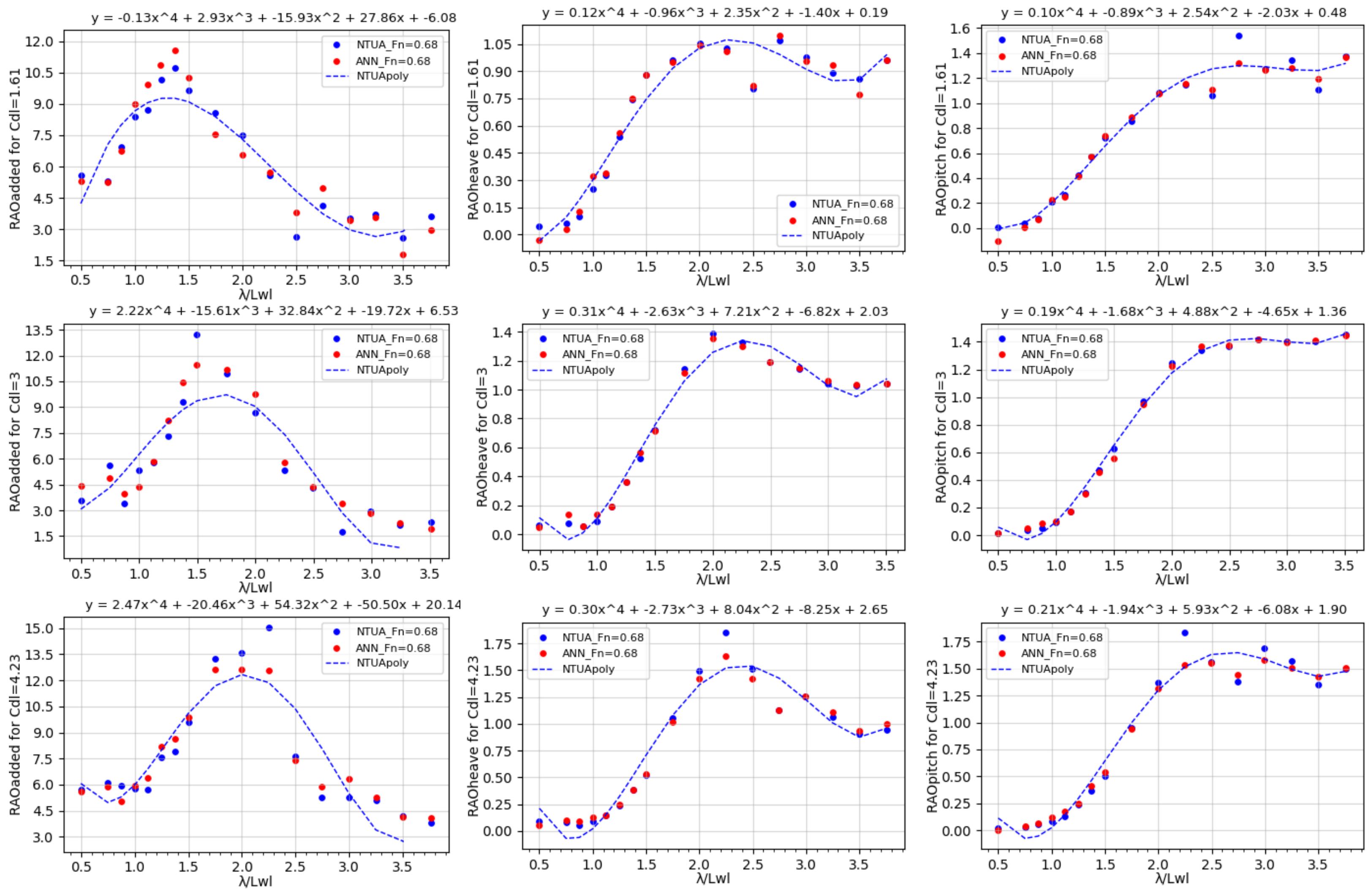


Figure 68: RAO *curves* for heave and pitch responses (L/B=7.00, Fn = 0.68)

`

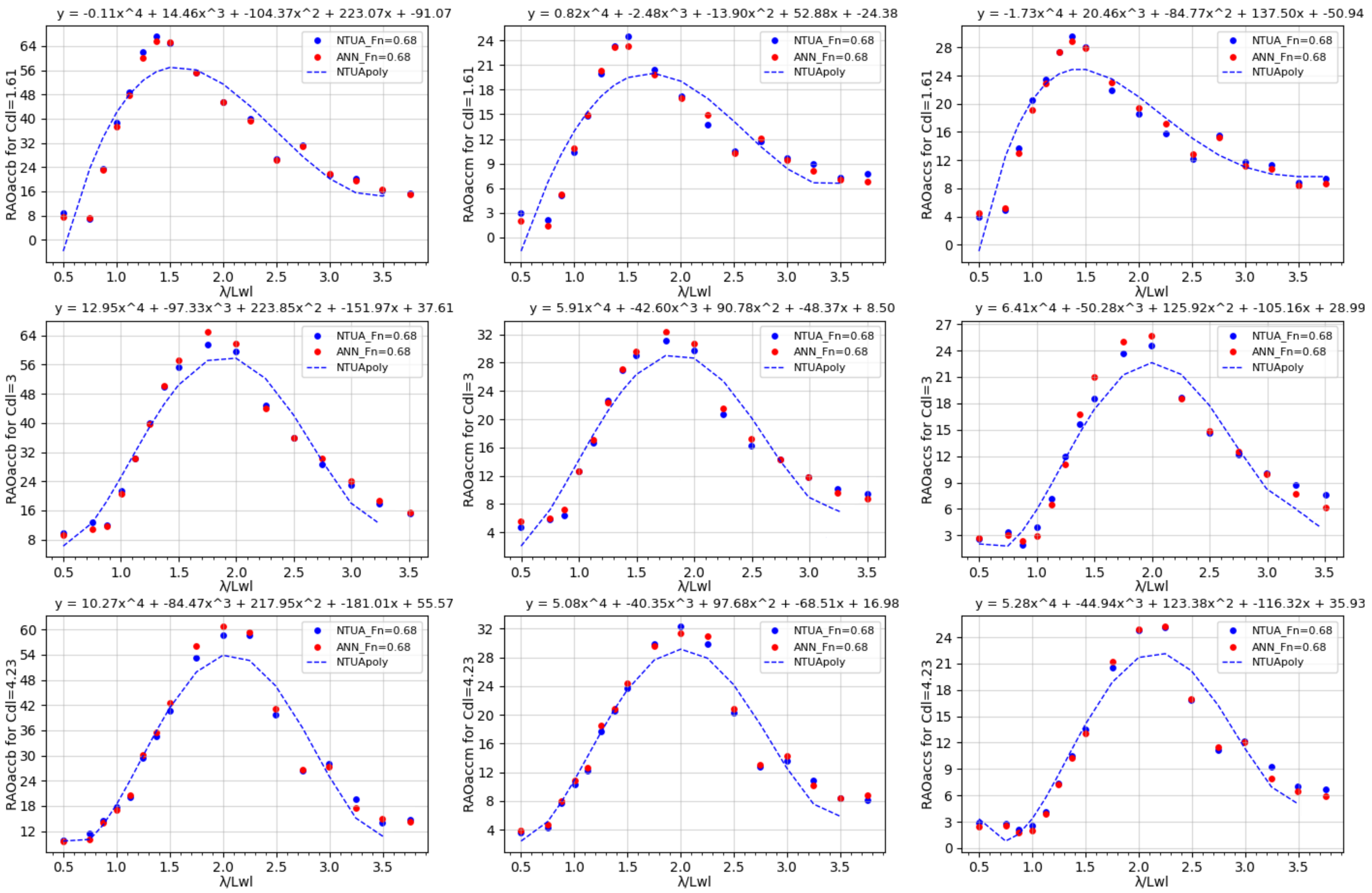


Figure 69: RAO *curves* for accelerations (L/B=7.00, Fn = 0.68)

`

The behaviour of the ANN model for Fn=0.68 is similar to Fn=0.34. The increased variance in added resistance compared to other responses, continues to persist in this higher Fn, and is slightly more noticeable in this range. The nature of the data points, especially for Fn=0.68 make the results to become more dependent on the training and testing set used, and the prediction results of the model can vary in range of approximately 15%. It is recommended more datapoints to be generated in order to make a more accurate representation.

However, taking into account the small dataset used for generating the results, its second order nature of added resistance with respect to wave heigh which makes it more sensitive to changes in the input variables, and also the performance the best fit curve at the testing datasets (as it is illustrated in the next chapter), we can conclude that the performance of ANN is considered satisfactory, at least for the scope of this study. An additional argument that supports the reliable prediction capabilities of the ANN is that the tank tests used to generate the vessel responses have a consistent margin of error as it can also be confirm for repeatability experiments utilized for this thesis. The results vary when obtaining the RAOs for these responses for the same non-dimensional λ/L ratio. Specifically, the results obtained from these tests typically fall within a similar error range, suggesting a consistent level of accuracy in the testing methodology.

4.4 Comparison with best fit curves

The final stage of evaluating an artificial neural network (ANN) model is to test its performance on various testing sets. This is important because the model needs to be able to generalize well and make accurate predictions on data that it has not been trained on. To achieve this, the testing sets are usually chosen randomly from a larger dataset that has been partitioned into training, validation, and testing sets. Although the sample presented below only includes two test sets, it's important to emphasize that the evaluation process involved testing the performance of the ANN model on multiple testing sets.

This final stage was utilized for comparison with the best fit curves. In order to aid in the understanding of whether or not the ANN models are capable of producing satisfactory predictions, apart from statistical measures that have been represented in previous chapters, the results derived from the ANN model are presented alongside the best fit curves. The use of best fit curves can provide a useful reference point for evaluating the performance of ANN models. These best fit curves were generated using high-order polynomials, lower-order polynomials, and 2nd order piecewise interpolation, and they provide a useful visual representation of the underlying relationships between the variables being studied. Comparing the predictions of the ANN models to the best fit curves can reveal any discrepancies or inconsistencies in the results. This information can be used to fine-tune the models and improve their performance. The most difficult responses to be predicted refers to accelerations and added resistance and thus the heave and pitch responses have been omitted for the below figure.

The performance of the ANN model can be evaluated by comparing its predictions to the best fit curves for RAO accelerations and RAO added resistance in the testing datasets. It should be mentioned that borders were chosen to be 10% of the maximum actual value of target parameters. Any values that fall outside of these borders are not included in the predictions of the models shown in the diagram, so it is possible that the accuracy of the models may vary for these values.

`

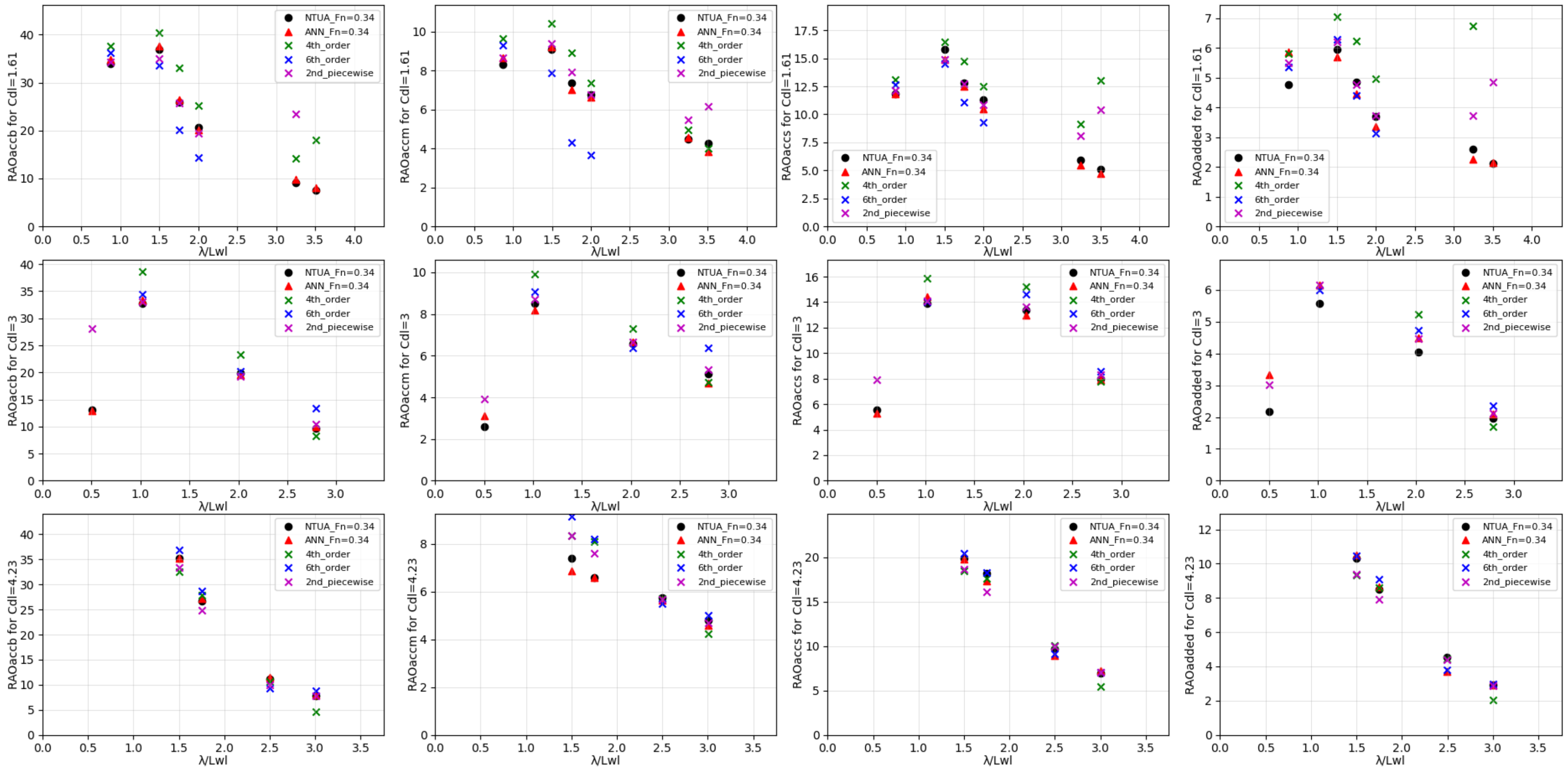


Figure 70: A typical example of predictions of ANN and best fit curves at testing datasets for the parent hull form for Fn=0.34

`

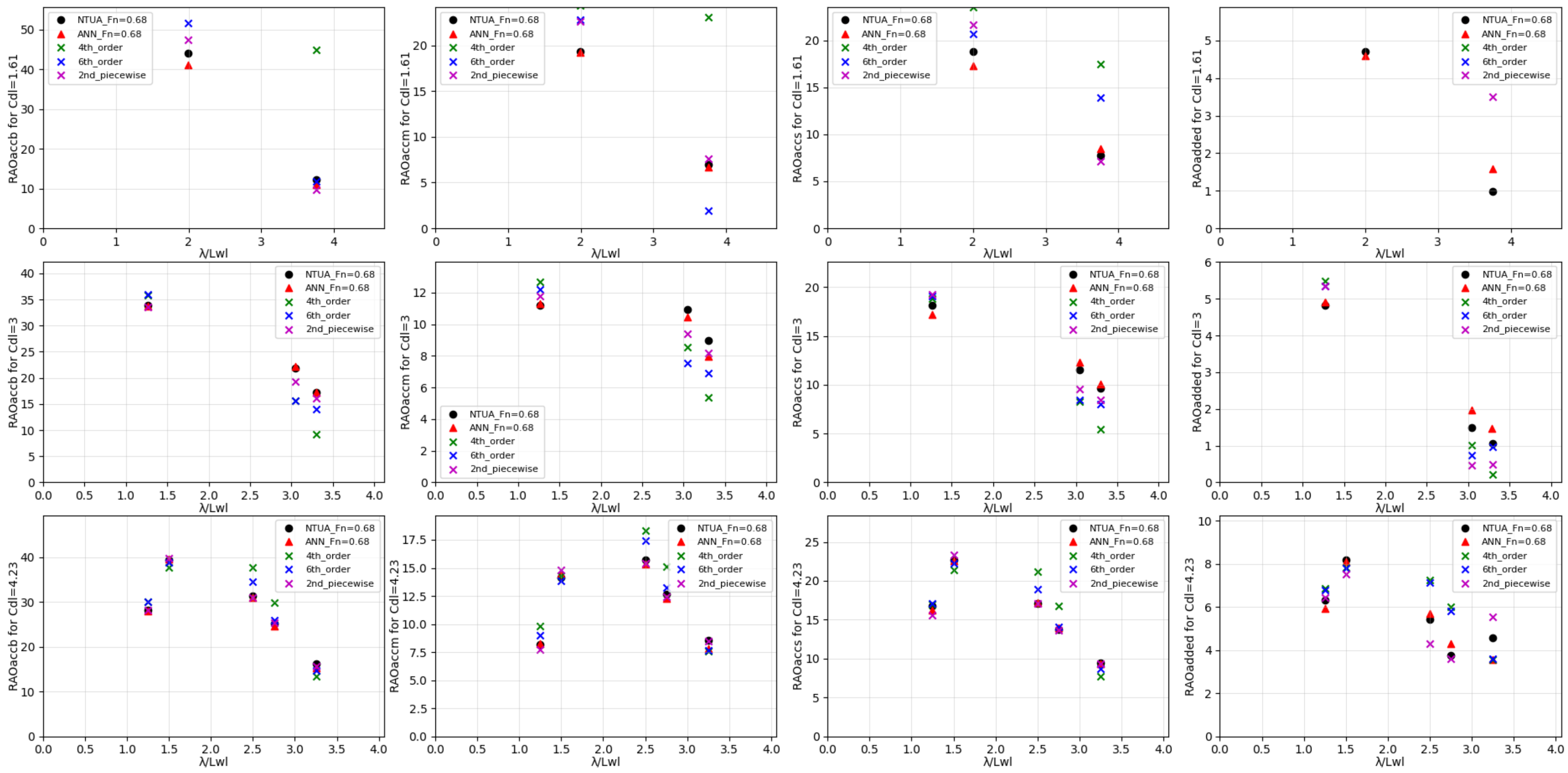


Figure 71: A typical example of predictions of ANN and best fit curves at testing datasets for the parent hull form for Fn=0.68

`

5. Conclusions and recommendations

The purpose of this work was to investigate, and develop an ANN model capable of predicting the RAO responses for the NTUA Semi-Planing Series in head seas and regular waves. On the way through, many configurations and optimization algorithms have been employed in order to find the most suitable model. The study also compared the prediction capabilities of various models, including high and low order polynomials and piece-wise interpolation models, with an ANN model for heave, pitch, accelerations, and added resistance for this type of hull forms. It was found that the performance of the ANN model was satisfactory for predicting those responses with good level of accuracy with the mean absolute percentage error of all the responses excluding adding resistance to be below 10%.

It is evident of the above representation of results that the error in predicting the added resistance is quite bigger than other responses. Specifically, the mean absolute percentage error of the added resistance after using many different training and testing sets, corresponds roughly to 15% for Fn=0.34 and Fn=0.68. However, considering the sensitivity of added resistance to the change of input parameters since it corresponds to second order variable and the fact that the tank tests used to generate the model responses have a consistent error range in repeatability tests, we can conclude that the ANN model is capable of approximating accurately those responses.

The 4th order polynomial showed relatively stable performance, although not with high accuracy. Higher-order polynomials were more prone to overfitting and therefore less reliable predictors. To mention also that the 2nd order piecewise function showed relatively good performance. The finding of this study suggests that the ANN model is the best predictor among the models tested.

Overall, the model is considered reliable and can serve as a library for obtaining response amplitude operators for heave, pitch, accelerations, and added resistance of such hull forms within the range of λ/L between 0.5 and 3.5, eliminating the need for model tests or computational fluid dynamics (CFD) simulations. This is significant since there are only a few available results in the literature pertaining to semi-planing hull forms. The response amplitude operators (RAOs) calculated for regular waves can be utilized for integrating over a range of frequency within a known wave spectrum to obtain corresponding responses. To ensure efficiency and accuracy in approximation with regular waves, it is important to use low wave heights, which helps to maintain this linearity. In cases where there are very large wave heights, the concept of linearity breaks down, and the regular waves in that range become irrelevant.

When dealing with higher wave heights, irregular waves are used for an accurate representation of vessel responses. In this context, it may be valuable to explore the creation of a neural network model that accounts for different wave conditions and encountering wave angles. This could help to identify how the hull form responds to different sea conditions and provide insights into how to optimize the design for different operating environments. Finally, may gathering more data for similar hull forms could be a step towards improving the accuracy of the predictions, as the ANN model requires a large amount of data in order to reach its full potential. This may involve conducting additional experiments or simulations to generate more data points. By gathering data of various similar hull forms and loading conditions, it would be possible to develop a more robust model capable not only to predict more stable results for added resistance, but also improves the accuracy and decrease the uncertainty for the full model.

`

6. Bibliography

https://web.archive.org/web/20181006235506/http://didattica.cs.unicam.it/lib/exe/fetch.php?media=didattica:magistrale:kebi:ay_1718:ke-11_neural_networks.pdf

https://dokumen.pub/python-machine-learning-9781783555130-1783555130-s-7419445.html

https://keras.io/api/layers/activations/

http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf

neuronalenetze-en-zeta2-2col-dkrieselcom.pdf

http://mycourses.ntua.gr/courses/NAVAL1078/document/S._Raptis_-_Computational_Intelligence_Notes.pdf

http://mycourses.ntua.gr/courses/NAVAL1078/document/%D4%C6%C1%D6%C5%D3%D4%C1%D3_%D5%F0%EF%EB%EF%E3%E9%F3%F4%E9%EA%DE_%CD%EF%E7%EC%EF%F3%FD%ED%E7-_%CA%C5%D6._4%2C5%2C6.pdf

http://mycourses.ntua.gr/courses/NAVAL1075/document/01_%C4%F5%ED%E1%EC%E9%EA%DE_%D0%EB%EF%DF%EF%F5_%26_%C5%F1%E3%E1%F3%F4%DE%F1%E9%EF_-_%D3%E7%EC%E5%E9%FE%F3%E5%E9%F2/01_Athanassoulis_Belibassakis_Dyn_Ploioy_2012-2013.pdf

https://dspace.lib.ntua.gr/xmlui/bitstream/handle/123456789/48435/Senteris%20Alexandros_Thesis.pdf?sequence=1

http://mycourses.ntua.gr/courses/NAVAL1075/document/02_%C4%F5%ED%E1%EC%E9%EA%DE_%D0%EB%EF%DF%EF%F5_%26_%C5%F1%E3%E1%F3%F4%DE%F1%E9%EF_-_%C4%E9%E1%F6%DC%ED%E5%E9%E5%F2/%C5%ED%FC%F4%E7%F4%E1_4-_%C1%F0%EF%EA%F1%DF%F3%E5%E9%F2_%F0%EB%EF%DF%EF%F5_%F3%E5_%E8%E1%EB%DC%F3%F3%E9%EF%F5%F2_%EA%F5%EC%E1%F4%E9%F3%EC%EF%FD%F2_-Seakeeping-/01_Sea_Waves_Random_Phase_Model.pdf

https://link.springer.com/article/10.1007/s00773-011-0151-0

https://www.academia.edu/81518638/Dynamic_Performance_of_the_NTUA_Double_Chine_Series_Hull_Forms_in_Random_Waves?auto=download&email_work_card=download-paper

(PDF) Resistance and Seakeeping Characteristics of a Systematic Series in the Pre-planing Condition (Part I) (researchgate.net)

(PDF) Dynamic Performance of the NTUA Double-Chine Series Hull Forms in Regular Waves (researchgate.net)

`

(PDF) Model tests for the hydrodynamic behavior of a semi-planing hull form in calm and rough waters (researchgate.net)

Appendix A

A.1 Executive summary of Python's libraries

Python3 software was used for data manipulation and processing as well as for ANN implementation which is a very powerful tool for data analysis. The integrated libraries, a gift from python developers, represent a collection of modules and objects in which a wide range of methods and attributes can be applied.

Pandas is a powerful library that is used for data analysis and manipulation. It provides various data structures, the most important of which are listed below:

- *Series*: One-dimensional data structure that is immutable in size and can hold data of a single type. It is similar to a column. A series can have a label or index, which can be used to identify and access the data

- *DataFrames*: Two-dimensional data structure that is mutable in size. It is similar to a table and consists of rows and columns. The rows are typically labelled with an index, and the columns can have labels and contain data of different types.

- *Dictionaries*: Data structure that stores key-value pairs. The keys in a dictionary must be unique, and the values can be of any data type. The keys are used to access the corresponding values, and the values can be modified or retrieved using the keys.

Pandas also includes a variety of built-in functions for data cleaning, filtering, and transformation, as well as tools for handling missing data such as *pandas.read_excel/csv()* methods.

SciPy is an open-source library in Python for scientific computing. It provides functionality for working with arrays, numerical optimization, signal processing, linear algebra, and more. The library provides a high-level interface for a wide range of mathematical and engineering tasks. *Interpolate* module is extensively used for this project. This module provides functions for various types of interpolation, including 1-dimensional and multi-dimensional interpolation. Univariate spline() is a curve fitting technique used to interpolate or smooth a set of data points by constructing a mathematical function that passes through the given data. This technique involves dividing the data into smaller intervals and fitting polynomial functions within those intervals (piece-wise) to approximate the original data.

Scikit-learn is a machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It is built on top of SciPy and includes a wide range of algorithms for tasks such as classification, regression, clustering, dimensionality reduction, and model selection and training. In this

`

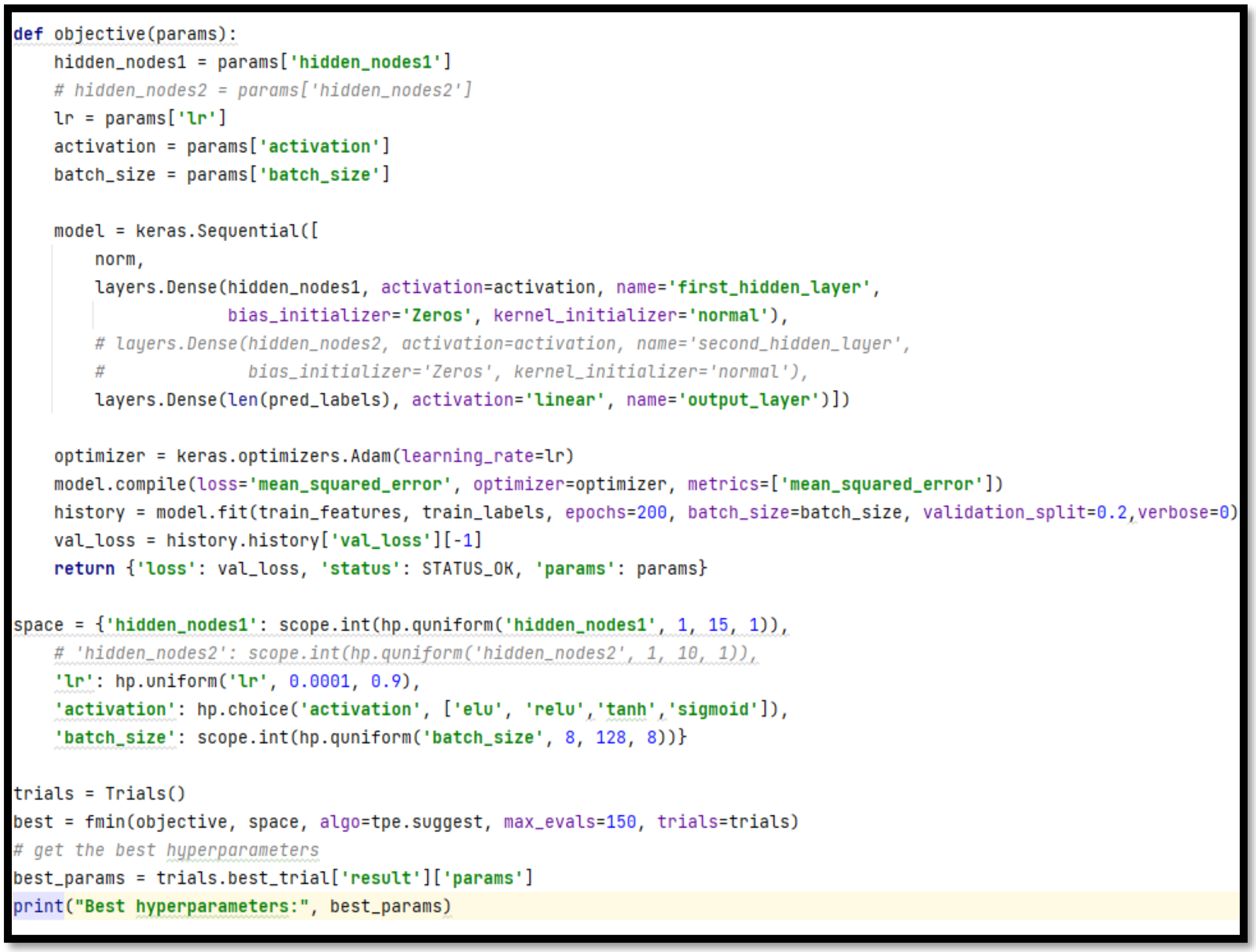
particular thesis it was utilized for Polynomial Interpolation. Especially, the *PolynomialFeatures* module is used to generate new features consisting of all polynomial combinations of the existing features. This means that given a set of features, *PolynomialFeatures()* can create additional features that are the powers of the original features up to a specified degree. These new features can then be used to fit a polynomial function to the data points. Then the module *LinearRegression()* is used to computes the coefficients of the polynomial function that best fits the data points by minimizing the sum of squared errors between the actual and predicted values of *Polynomial Features*.

Matplotlib is a plotting library which provides an object-oriented application programming interface (API) for creating static and interactive visualizations in Python. The method *fig,ax = subplots()* was extensively used to create multiple plots in the same figure. The *fig* object allows for manipulation of the overall figure properties, while the *ax* object allows for manipulation of the individual plot axes.

Seaborn is a data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn is particularly well-suited for visualizing complex datasets with multiple variables, and offers a variety of built-in plotting functions and customization options. *Pairplot()* function was used for creating parametric plots that visualizes the pairwise relationships between variables in a dataset. It's a useful tool for exploratory data analysis and identifying patterns in data. The 'hue' parameter in Pairplot allows for color-coding of the plots based on a categorical variable. This can be helpful for visualizing the relationships between variables while also considering their grouping based on the categorical variable.

Keras is a high-level Neural Networks API that runs on top of TensorFlow and provides all necessary mathematical algorithms for building, training, optimization and validating procedures. It supports a variety of popular architectures and layers. The Sequential model in Keras provides a simple and intuitive way to build neural network models with a linear stack of layers. A Dense layer is a type of neural network layer in which each neuron in the layer is connected to every neuron in the previous layer. It is the most common type of layer in deep learning models, and is used for a wide range of tasks such as classification, regression, and sequence modelling. A Sequential model can be created by adding layers one by one and compile it with an optimizer, loss function, and metrics. *Hyperopt* library of keras was used for optimizing the ANN by searching through hyperparameter space efficiently. In hyperplot, the search space can be defined using various distribution functions. It uses trials class to keep track of the optimization history, including the hyperparameters and loss values. A sample of ANN model in keras can be found below:

`



```python
def objective(params):
    hidden_nodes1 = params['hidden_nodes1']
    # hidden_nodes2 = params['hidden_nodes2']
    lr = params['lr']
    activation = params['activation']
    batch_size = params['batch_size']

    model = keras.Sequential([
        norm,
        layers.Dense(hidden_nodes1, activation=activation, name='first_hidden_layer',
                     bias_initializer='Zeros', kernel_initializer='normal'),
        # layers.Dense(hidden_nodes2, activation=activation, name='second_hidden_layer',
        #              bias_initializer='Zeros', kernel_initializer='normal'),
        layers.Dense(len(pred_labels), activation='linear', name='output_layer')])

    optimizer = keras.optimizers.Adam(learning_rate=lr)
    model.compile(loss='mean_squared_error', optimizer=optimizer, metrics=['mean_squared_error'])
    history = model.fit(train_features, train_labels, epochs=200, batch_size=batch_size, validation_split=0.2,verbose=0)
    val_loss = history.history['val_loss'][-1]
    return {'loss': val_loss, 'status': STATUS_OK, 'params': params}

space = {'hidden_nodes1': scope.int(hp.quniform('hidden_nodes1', 1, 15, 1)),
    # 'hidden_nodes2': scope.int(hp.quniform('hidden_nodes2', 1, 10, 1)),
    'lr': hp.uniform('lr', 0.0001, 0.9),
    'activation': hp.choice('activation', ['elu', 'relu','tanh','sigmoid']),
    'batch_size': scope.int(hp.quniform('batch_size', 8, 128, 8))}

trials = Trials()
best = fmin(objective, space, algo=tpe.suggest, max_evals=150, trials=trials)
# get the best hyperparameters
best_params = trials.best_trial['result']['params']
print("Best hyperparameters:", best_params)
```

Figure 72: Sample of an ANN model with keras

The objective function takes a dictionary of hyperparameters as input, builds a neural network model using those hyperparameters, trains it on a training dataset (train_features and train_labels), and returns the validation loss of the trained model.

The space dictionary defines the search space for the hyperparameters. Each key in the dictionary corresponds to a hyperparameter, and the value defines the search range and type of the hyperparameter. For example, hidden_nodes1 is an integer hyperparameter that can take values between 1 and 15 in steps of 1, while lr is a continuous hyperparameter that can take any value between 0.0001 and 0.9.

The Trials object is created to keep track of the hyperparameter optimization process.

The fmin function from the hyperopt library is used to minimize the objective function over the search space. The algo parameter specifies the search algorithm to use, in this case Tree-structured Parzen Estimator (TPE). The max_evals parameter determines the maximum number of iterations to run the optimization process. The trials parameter is the Trials object created earlier to track the optimization process.

`

After the optimization process is complete, the best hyperparameters found during the search are extracted from the Trials object and stored in the best_params variable.

Finally, the best hyperparameters are printed to the console.

A2. ANN coefficients for RAO accelerations and added resistance (Fn=0.34)

Table 11:ANN model's coefficients for RAO accelerations and added resistance (optimizer: Nadam, inputs:16, hidden_neurons:20, batch_size:16, lr:0.035)

Input Layer	**WEIGHTS (16,)** [0.62256247 0.29682294 0.08590208 0.14449531 3.3283837 2.7435267 2.1458158 1.9855535 1.8644297 0.65084195 0.8478933 5.5195312 1.9870209 2.803473 0.48050517 1.7545781 2.9940107]
	BIAS (16,) [0.03776717 0.0338604 0.00197593 0.00457937 0.06511195 1.476902 0.8167769 0.8285403 0.826138 0.0762987 0.19100718 1.1216888 1.0801232 0.5586313 0.01452228 0.04849622 1.1496272]
Hidden layer	**WEIGHTS (20,17)** [[-1.01808846e+00 2.33035162e-01 -2.09167433e+00 -4.67635542e-01 -1.12973740e-02 4.51382482e-04 -3.59710395e-01 1.22706406e-01 -3.46822143e-02 -4.06687021e-01 -2.36781979e+00 -6.58386111e-01 -5.99517882e-01 -6.56965375e-01 -1.79140553e-01 -7.25868404e-01 -6.37334108e-01 -1.81007221e-01 -1.43319368e+00 -4.83160347e-01] [-1.70504823e-02 1.64400518e-01 -3.59024793e-01 1.42553186e+00 -1.80710584e-01 5.44824488e-02 -1.20394778e+00 1.07677031e+00 2.85837743e-02 -2.03615856e+00 4.12957281e-01 -1.95245087e-01 6.94112837e-01 -2.24142045e-01 1.01954484e+00 1.39474344e+00 7.98733354e-01 1.29748654e+00 9.57058549e-01 8.50404024e-01] [-3.11417371e-01 2.28422821e-01 -3.97236757e-02 6.96534276e-01 -3.64566952e-01 1.75936949e-02 -2.30910107e-01 -1.68530300e-01 -4.43688184e-02 -3.07213128e-01 1.26419321e-01 1.58953714e+00 7.26147354e-01 2.58745492e-01 8.08302641e-01 5.01880310e-02 5.31710625e-01 5.76150358e-01 -1.42603970e+00 2.32485697e-01] [4.22049791e-01 -5.33652641e-02 -2.95320094e-01 5.19585535e-02 -9.78351653e-01 -6.65634274e-02 4.11544181e-02 1.09306112e-01 1.18673984e-02 1.43676281e+00 -2.07188159e-01 -1.32762003e+00 1.79895818e-01 -1.42563313e-01 3.71652693e-01 -1.14029832e-01 3.61753523e-01 -1.92052618e-01 8.09160471e-01 5.33053696e-01] [1.28383292e-02 2.13752002e-01 -3.34083661e-02 1.63447663e-01 -4.44830991e-02 -3.73928063e-02 -9.38239872e-01 -8.92741323e-01 2.93199141e-02 -8.57665911e-02 -6.69206738e-01 1.37781858e-01 -2.02910423e-01 2.62704790e-01 -3.46529037e-01 -3.53926718e-01 -4.72389519e-01 -5.26117742e-01 -2.59359866e-01 -7.47575521e-01] [8.88664365e-01 -1.24198817e-01 1.73346922e-01 8.79143536e-01 -3.53530556e-01 -6.85854303e-03 5.04284501e-01 6.57343268e-01 6.13737386e-03 -5.71757555e-01 6.06592357e-01 -1.66834265e-01 9.57838058e-01 -9.14150774e-01 2.55460799e-01 5.73682308e-01 5.86221553e-02 4.87065256e-01 -2.85783589e-01 1.24603081e+00] [5.50542951e-01 -5.56767695e-02 -3.16212267e-01 -2.60184377e-01 -1.48951840e-02 5.71014434e-02 -7.05085695e-01 -3.63885462e-01 -1.58315413e-02 6.09986708e-02 8.85755047e-02 -1.10317871e-01 4.14416045e-01 1.71895787e-01 -1.50845319e-01 -5.36406696e-01 -5.45964539e-02 -5.45455039e-01 3.85223776e-01 -1.98914319e-01] [3.13410133e-01 1.94693044e-01 1.44920158e+00 1.96091518e-01 -1.08683713e-01 1.11147258e-02 -6.48213208e-01 -2.91090906e-01 -4.77936901e-02 -9.46275070e-02 2.36639887e-01 8.37714493e-01 4.44906831e-01 2.36934319e-01 4.12278056e-01 -1.98957026e-01 2.05098659e-01 -3.97552371e-01 -6.47470653e-01 5.04487574e-01] [-3.07846338e-01 -9.30180371e-01 -5.81413209e-01 -9.26353037e-01 9.39808547e-01 3.07632480e-02 8.72023925e-02 -9.67613995e-01

	3.79617028e-02 -5.45677304e-01 -3.96210462e-01 3.67291659e-01 3.69861543e-01 -2.34212786e-01 -7.47203231e-01 -1.04835975e+00 -7.54725873e-01 -8.10621917e-01 6.22289479e-01 -1.01866031e+00] [2.99605250e-01 3.21772434e-02 -3.93151671e-01 -2.69318610e-01 -8.52886885e-02 -3.94613929e-02 -3.67248267e-01 -2.43962437e-01 4.04013181e-03 4.38985795e-01 -1.28655266e-02 2.73850650e-01 -6.35007545e-02 5.51177502e-01 2.01246776e-02 -4.35919732e-01 1.63368374e-01 -3.15314382e-01 3.29194754e-01 -2.11457118e-01] [3.51540625e-01 -3.45671028e-01 6.96001709e-01 -5.50027192e-01 2.69294709e-01 -5.93845211e-02 -5.68203740e-02 -2.34518409e-01 -4.17225845e-02 2.04982534e-02 3.42703104e-01 2.44941518e-01 -4.64292839e-02 4.39950287e-01 -4.01036650e-01 -3.65940213e-01 -3.39402795e-01 -7.45448053e-01 -4.45187330e-01 9.07183439e-02] [1.28192019e+00 -2.00346000e-02 -1.74617660e+00 1.43250927e-01 4.66340892e-02 -3.43935639e-02 -5.23199588e-02 1.02131951e+00 2.20929296e-03 4.95975971e-01 5.13173163e-01 -2.33958554e+00 -2.24338865e+00 -1.21688887e-01 -3.43482286e-01 1.07808039e-01 -2.45850578e-01 2.64060665e-02 1.98978436e+00 8.72508526e-01] [-4.22085255e-01 -2.89317250e-01 -4.39368308e-01 -2.76683033e-01 1.00297995e-01 4.50521521e-02 1.06902742e+00 2.62407422e-01 -3.50827463e-02 4.73047972e-01 -8.73874843e-01 -7.25111812e-02 6.59433186e-01 -5.86097360e-01 1.65104687e-01 -2.89800823e-01 -2.85217911e-01 -6.24519661e-02 5.40575147e-01 -5.30245423e-01] [-2.20387697e-01 -6.26887321e-01 9.48791146e-01 8.69805664e-02 -5.32371640e-01 6.88212439e-02 2.32180253e-01 3.54251236e-01 3.22152227e-02 -3.81756186e-01 7.85916746e-01 8.88220191e-01 4.98586088e-01 -7.83651590e-01 -6.81905448e-02 3.07686448e-01 1.18044905e-01 3.26306671e-01 -9.00749147e-01 -3.08877468e-01] [-5.84547400e-01 -3.27710301e-01 7.48654485e-01 1.47486880e-01 -3.64708304e-01 -9.04250250e-04 1.39148045e+00 6.99807346e-01 4.76568229e-02 1.94025293e-01 4.72903103e-01 3.31008494e-01 3.54085207e-01 -1.64676696e-01 -1.42372668e-01 6.46045446e-01 1.78651690e-01 2.40102276e-01 -4.03266907e-01 3.10588093e-03] [-4.55276132e-01 -4.97850716e-01 3.26971352e-01 1.31630853e-01 -2.15248540e-01 -2.95217857e-02 4.83618677e-01 5.77750742e-01 -3.72366533e-02 1.56782001e-01 8.52416605e-02 3.57279003e-01 3.77803445e-01 -6.23526931e-01 1.05324358e-01 1.31985351e-01 3.35727423e-01 3.26973259e-01 1.09543167e-01 -2.38617063e-01] [5.06774604e-01 8.85650158e-01 -8.86112571e-01 -2.17163935e-01 8.18968788e-02 -2.61702538e-02 -1.35805443e-01 -4.45747584e-01 4.49213423e-02 -4.17352058e-02 -5.88887036e-01 -1.18549161e-01 -7.36166120e-01 1.37882960e+00 -4.29572552e-01 -3.72522712e-01 -4.12094265e-01 -4.72081959e-01 -1.69794098e-01 3.24147761e-01]]
	BIAS (20,) [0.944985 1.2362249 1.5935267 0.9919175 -0.6272939 -0.23976956 -0.26233917 0.8574633 -0.21611468 0.31076023 2.0434659 1.642888 0.5127198 1.2769005 1.3540316 1.0132515 1.6343058 0.8091653 0.01607968 0.96688336]
Output layer	**WEIGHTS (4,20)** [[0.71578467 0.6047299 -0.00499988 0.65065485] [0.939259 0.19958808 0.4471704 0.46372008] [1.2609822 -0.41260085 1.6147916 -0.2698291] [1.1075486 -0.07727033 0.4612086 0.30337447] [-1.6032615 -1.0021654 -0.11699753 -0.14422634] [-0.08080845 0.50405663 -0.20024632 0.40676162] [-1.970527 -1.0889789 -0.05631466 -0.22081614] [0.8094743 0.9659899 -0.63601345 0.28432006] [0.01747553 0.23572035 0.13315141 0.46292067] [-1.1332687 1.776734 0.88956916 -0.54351944] [1.8182377 0.1460223 0.6350799 0.69124454] [-0.13877463 1.9903052 -2.233096 0.27219552] [-0.04619284 -1.0730829 0.9559342 -0.24633524] [0.73172635 -0.796439 1.4352458 -0.08044] [1.0773305 0.37195006 0.51403195 -0.6400854] [1.3557489 0.5477104 0.47931746 0.70547694] [1.0340424 0.5322712 0.46203455 -0.16718037] [1.4126979 -0.03570934 0.5205944 -0.26975414] [0.8065183 0.82227254 -0.7277692 -0.45387974] [0.8913451 0.50366914 0.19179587 1.1733906]]
	BIAS (4,) [0.8918625 0.7082447 0.8212668 0.4385635]

	BIAS (18,) [1.8804214 1.541594 0.02831864 0.86169493 0.79202527 -0.00309563 -2.0917907 0.77442825 0.6014528 0.72152156 -1.9170315 -0.18745787 0.86710197 1.1081414 0.97171724 1.3256344 1.0477724 1.2123548]
Output layer	**WEIGHTS (4,18)** [[1.2602606 0.39307007 0.67444754 0.0046742] [1.0910406 0.91094863 0.5664749 0.5196367] [0.60795337 1.083153 0.2397921 0.27405602] [1.0221375 -0.19495444 0.83407617 0.05510397] [1.0895122 0.38338122 0.47652844 -0.10404018] [0.9625271 -0.08445769 -0.29449716 0.17927241] [1.06631 1.3156431 -0.9694705 -0.7771187] [1.3356769 0.43756217 0.5123394 0.9668349] [-0.15622151 1.6123075 -1.2022281 -0.01967052] [1.100416 1.0666938 -0.5367843 -0.07778013] [-1.9868752 -0.74177337 -0.48291537 -0.16255577] [0.5099548 0.34622318 0.88951904 -0.1320936] [0.9010512 -0.84672385 1.1978266 -0.38012993] [1.2170677 -0.02398186 -0.33751863 -0.31085458] [1.1180247 0.45543092 0.84786296 0.04875967] [1.0527034 0.7199191 0.27921 0.2958843] [0.97134477 0.14543436 1.4558579 -0.09750448] [1.1718782 0.48852015 0.6868209 0.60139245]]
	BIAS (4,) [0.9779138 0.73985386 0.9115663 1.3193431]

A3. ANN coefficients for RAO accelerations and added resistance (Fn=0.68)

Table 12:ANN model's coefficients for RAO accelerations and added resistance (optimizer: Nadam, inputs:16, hidden_neurons:18, batch_size:32, lr:0.094)

Input Layer	**WEIGHTS (16,)** 0.6166049 0.3779225 0.151047 0.189631 3.4438028 6.7591724 2.4631262 1.4391725 1.9057757 0.7279169 0.7107176 5.55625 5.931079 2.8308396 0.48223326 2.99105]
	BIAS (16,) [3.7843145e-02 3.6186103e-02 7.7717863e-03 8.6571146e-03 1.4061789e-01 8.7733145e+00 2.5468292e+00 6.9193780e-01 9.2604035e-01 2.2512019e-01 2.7858862e-01 1.1415234e+00 7.8391204e+00 5.8934999e-01 1.5482384e-02 1.1724964e+00]

Hidden layer	**WEIGHTS (18,16)** [[-1.3617705 -1.3458458 -2.145031 -0.79479015 -0.5539892 -0.9850121 -1.2263374 -0.2596401 -1.0600888 -1.7532105 -2.1747391 -0.55914265 -2.597239 -0.55137455 -2.3656118 -1.7183303 -0.84393513 -1.5134598] [0.35888335 0.70088625 0.09314877 -0.18888682 0.43398187 1.0180863 0.9501011 1.4733189 -0.27017444 2.166617 -2.0447025 -0.6260666 1.3608143 2.006951 0.9981873 1.2148957 0.75681204 0.84337974] [0.19492385 0.2573939 -0.07864635 -0.06365991 0.24115673 -0.60628325 -1.1477773 -0.35171762 2.1444418 -0.4039466 -0.27948156 -0.09708434 -0.3610335 -0.20586699 0.44127944 0.2593653 0.7427258 0.02423783] [0.39407247 0.2329722 0.34480312 0.00779036 0.08294812 -0.90870523 0.07578766 1.1189418 -1.572741 0.1452583 0.22798426 -0.46722266 -0.96031237 0.2961728 0.50936294 0.50688934 1.9769082 0.3828996] [-0.3344791 0.0302862 -0.9572306 0.47556335 -1.0214993 0.3492469 0.24855687 -0.23787604 0.07537663 -0.993356 -1.3683058 0.6336485 -0.5428179 -0.36890292 -0.7365786 -0.6933031 -0.01757032 -1.3594629] [-1.6580076 0.64484596 0.06582579 -0.28064457 -1.8669612 0.35745522 -0.8514549 1.0242718 -0.26935846 0.39320037 0.59779894 -1.0644042 -0.09286435 0.7328842 0.4298125 0.35665208 0.6572461 0.4958046] [0.01397251 0.36978617 -0.19313543 0.17936674 -0.02423091 0.847896 0.10698835 -1.1654141 0.08255671 -0.03343971 -0.43092564 0.37167034 0.43158686 -0.2888874 0.14901036 0.28657803 -1.3636101 0.15940045] [0.42110494 0.17386901 0.38226286 0.64192015 -0.60778654 0.5860467 0.18697919 -1.3814751 0.31915683 -2.1219985 0.5546528 1.0092851 0.6287212 -0.8817091 -0.3832402 -0.3162774 -0.5562183 -0.14038809] [-2.356948 -0.38787544 -0.7414526 -0.53848326 -0.80977166 -1.1626456 -1.0043248 -0.90887475 -0.10685331 -0.35539946 -0.32394052 -0.7347626 -0.90428853 -0.7266227 -0.9124985 -1.2396033 -0.5558907 -1.3663667] [0.1505592 0.60869193 0.24574837 0.36051348 0.09636551 0.8473465 -0.03573273 -0.5214279 0.18276428 0.406771 -0.583653 0.2523638 0.1541094 -0.15465158 0.85703725 0.6920459 -0.728681 0.8434177] [0.14335735 0.16652128 -0.5036309 0.3945925 -0.04236839 0.6023431 -0.14436075 -0.2902741 0.21108666 -0.21112457 -1.0774478 0.30360165 0.3250042 -0.5077938 0.28230622 0.11009566 -0.19286418 -0.0935223] [0.19837537 1.0590528 1.4401251 0.30955487 -1.1081176 0.5167236 2.6740408 1.1179812 -1.3473201 0.42572844 -0.4919574 0.21369086 -0.7159042 -0.6150627 0.18813282 0.5281562 -0.54396343 -0.5265536] [-1.6697141 -0.03360011 -0.30246794 -0.526126 -1.705994 -0.21034566 0.5664226 -0.5713105 -0.27745232 0.2855231 1.0626874 -0.66521037 -0.16982521 0.34009734 -0.1285533 -0.41687053 -0.8247876 -0.42988035] [0.26922438 0.6946212 -0.6734936 -0.68020654 0.23677306 0.5854924 -0.7472295 0.1750754 1.0556523 -0.187886 0.8359897 0.8622061 1.3938761 0.508359 0.939347 0.1913872 1.569678 1.1329042] [-0.16706961 -0.09034313 0.06902049 -0.37572855 0.9555319 -0.2408902 -0.7905918 -0.17052224 0.42746595 0.5008087 1.217873 0.14614633 0.8689367 0.3142088 0.7817946 -0.01126756 0.15014468 1.2760973] [0.7088312 0.48430574 1.8039975 2.1584885 -0.7226943 -0.35167754 -0.6262165 0.27182698 -0.42344743 -0.24857177 0.41425392 0.1738483 -0.57201344 0.02710992 -0.33683285 0.5770609 -0.0476303 -0.6466031]]
	BIAS (18,) [1.8804214 1.541594 0.02831864 0.86169493 0.79202527 -0.00309563 -2.0917907 0.77442825 0.6014528 0.72152156 -1.9170315 -0.18745787 0.86710197 1.1081414 0.97171724 1.3256344 1.0477724 1.2123548]
Output layer	**WEIGHTS (4,18)** [[1.2602606 0.39307007 0.67444754 0.0046742] [1.0910406 0.91094863 0.5664749 0.5196367] [0.60795337 1.083153 0.2397921 0.27405602] [1.0221375 -0.19495444 0.83407617 0.05510397] [1.0895122 0.38338122 0.47652844 -0.10404018] [0.9625271 -0.08445769 -0.29449716 0.17927241] [1.06631 1.3156431 -0.9694705 -0.7771187] [1.3356769 0.43756217 0.5123394 0.9668349] [-0.15622151 1.6123075 -1.2022281 -0.01967052] [1.100416 1.0666938 -0.5367843 -0.07778013] [-1.9868752 -0.74177337 -0.48291537 -0.16255577] [0.5099548 0.34622318 0.88951904 -0.1320936] [0.9010512 -0.84672385 1.1978266 -0.38012993] [1.2170677 -0.02398186 -0.33751863 -0.31085458] [1.1180247 0.45543092 0.84786296 0.04875967] [1.0527034 0.7199191 0.27921 0.2958843] [0.97134477 0.14543436 1.4558579 -0.09750448] [1.1718782 0.48852015 0.6868209 0.60139245]]

`

	BIAS (4,) [0.9779138 0.73985386 0.9115663 1.3193431]

A4. ANN coefficients for RAO heave and pitch for both Fn=0.34 and Fn=0.68

Table 13: ANN model's coefficients for RAO heave and pitch
(Optimizer: SGD with Momentum and Nesterov, inputs:19, hidden_neurons:9, batch_size:64, lr:0.02)

Input Layer	**WEIGHTS (19,)** [0.62256247 0.29682294 0.08590208 0.14449531 3.3283837 2.7435267 2.1458158 1.9855535 1.8644297 23.650444 7.8338113 10.3973465 4.4787283 5.5195312 1.9870209 2.803473 0.48050523 1.7545781 2.9940107]
	BIAS (19,) [3.7767168e-02 3.3860400e-02 1.9759270e-03 4.5793694e-03 6.5111950e-02 1.4769020e+00 8.1677687e-01 8.2854033e-01 8.2613802e-01 1.9028891e+02 1.8651718e+01 2.0616280e+01 1.2157830e+01 1.1216888e+00 1.0801232e+00 5.5863130e-01 1.4522284e-02 4.8496217e-02 1.1496272e+00]
Hidden layer	**WEIGHTS (9,19)** [[-4.01746444e-02 1.01093352e-01 -3.18577960e-02 -9.49883461e-02 1.67257190e-02 -4.69035879e-02 3.21033671e-02 -1.92310967e-04 -1.55207319e-02] [-4.25831368e-03 9.38304216e-02 -1.88630968e-02 -1.28664644e-02 -1.38328964e-04 7.11861346e-03 2.84672193e-02 -6.98130950e-02 2.12953333e-02] [1.42097631e-02 -3.56625244e-02 -4.78763841e-02 -3.24590839e-02 6.87740743e-03 1.46214687e-03 -5.65449111e-02 1.39060151e-03 4.07549329e-02] [8.71946663e-03 3.55633087e-02 3.45142968e-02 -4.21207696e-02 3.63262780e-02 4.61886497e-03 -4.52431627e-02 -4.60057938e-03 -1.45853711e-02] [1.28906816e-01 4.03887071e-02 -1.24036754e-02 -1.08267717e-01 -1.00235306e-02 -1.74092613e-02 1.03701837e-01 -6.75177425e-02 3.52910049e-02] [7.08515719e-02 -2.20995247e-02 -5.39766811e-02 -3.09422221e-02 3.36553566e-02 5.91418296e-02 -7.41366670e-02 -6.22114316e-02 -6.34335577e-02] [8.57804939e-02 2.47187205e-02 -1.25728667e-01 1.36542231e-01 -2.28391677e-01 6.48049936e-02 -3.88609129e-04 9.63690951e-02 1.02455504e-01] [-1.95722328e-03 -1.35538399e-01 7.61304572e-02 1.37617499e-01 -7.31527954e-02 1.40149131e-01 4.44276556e-02 2.43150163e-02 5.38688786e-02] [2.43177507e-02 -6.03134632e-02 7.35178441e-02 1.03903934e-01 -1.16475575e-01 1.75085086e-02 -5.67585528e-02 2.37175077e-02 -1.89777911e-02] [5.79900742e-02 3.95220593e-02 -1.36659192e-02 -1.28079988e-02 8.54428113e-02 4.65486906e-02 -2.30043977e-02 8.61312822e-02 -8.33226088e-03] [2.18278226e-02 8.06807131e-02 5.34624122e-02 5.22138961e-02 -2.01575011e-02 4.92480583e-02 -2.84191367e-04 3.03703174e-02 1.79569982e-02] [-1.87713001e-02 4.72531617e-02 1.16109997e-02 -1.26182858e-03 -2.21048426e-02 7.36138970e-02 -1.58760771e-02 3.81030217e-02 -4.89458814e-02] [-1.15460657e-01 1.38520822e-02 1.65320220e-04 5.92559204e-03 5.53115681e-02 2.41743531e-02 2.53746510e-02 -8.54371395e-03 7.74098411e-02] [-4.83703651e-02 2.37047654e-02 1.59384403e-02 2.49589365e-02 -6.52640313e-02 -5.53264096e-02 2.38345955e-02 5.69230691e-02 2.52672415e-02] [-7.62091056e-02 7.79217333e-02 3.87254581e-02 1.38340117e-02 -2.66808681e-02 1.13205157e-01 2.97184102e-02 -8.03487934e-03 2.49360017e-02]

`

	[-3.33237052e-02 9.63995606e-02 4.05435488e-02 4.40350808e-02 7.30084628e-02 7.57392049e-02 5.00668772e-03 -7.58183002e-02 -1.47712622e-02] [1.73038319e-02 -2.54510939e-02 3.63003314e-02 8.83553550e-02 -5.56655228e-03 -3.06457989e-02 -6.26173690e-02 -4.79081683e-02 -2.70311311e-02] [6.05927855e-02 3.43293995e-02 6.88425973e-02 -6.90047583e-03 -1.52261555e-03 4.61610183e-02 -9.19030979e-02 -2.46934053e-02 3.86474691e-02] [-1.15109734e-01 -6.01875799e-05 -6.49853200e-02 2.36059520e-02 9.41636600e-03 -8.16580430e-02 9.31957960e-02 4.11607772e-02 -3.85009195e-03]]
	BIAS (9,) [-0.013453 -0.17110762 -0.05173545 0.35176358 -0.30122194 0.19454896 0.07215822 0.08793311 -0.03403595]
Output layer	**WEIGHTS (2,9)** [[0.01425115 -0.0105062] [0.08957645 -0.57810324] [-0.5534149 0.66307396] [0.29359332 0.47057867] [-0.41305748 -0.22452106] [0.01934183 0.5550909] [-0.33677825 0.70901424] [0.05833336 0.20903519] [0.14098872 -0.18645616]]
	BIAS (2,) [0.47389916 0.39326108]