Radboud University Nijmegen & MARIN

Quiescent Periods in Ship Motions

Report for Models Laboratory

Authors:
Casper Algera[†]
Dinand Blom
Emma Ex
Jort Jacobs
Steven Janssen
Jan Schobers
Jorian Weststrate

Supervisor Radboud University:
dr. A. Javan Peykar

Supervisors MARIN:
dr. ir. E.F.G. van Daalen
dr. ir. A. Koop

† Contact at cj.algera@gmail.com, this is the primary contact for this report.

June 27, 2025

Contents

1 Preface **2**

2 Preliminaries **3**
2.1 The six degrees of freedom of ship motion . . . 3
2.2 Data collection . . . 3
2.3 Definition of quiescent periods . . . 5
2.4 Training models and calculating performance . . . 7

3 Distribution of Quiescent Periods **10**
3.1 Exponential distribution . . . 10
3.2 Hypoexponential distribution . . . 11
3.3 Finding a better model . . . 12

4 Logistic Regression **14**
4.1 Our approach . . . 14
4.2 Results . . . 16

5 Random Forest **18**
5.1 Our approach . . . 18
5.2 Results . . . 18

6 Long Short-Term Memory (LSTM)-models for QP classification **20**
6.1 Our approach . . . 20
6.2 Results . . . 22

7 ARIMA **28**
7.1 Our approach . . . 28
7.2 Results . . . 28

8 Markov models **32**
8.1 Our approach . . . 32
8.2 Results . . . 35

9 Future work **36**
9.1 The distribution of quiescent periods . . . 36
9.2 AI models for QP classification . . . 36
9.3 Long Short-Term Memory networks for wave forecasting . . . 37
9.4 Neural Hierarchical Interpolation for Time Series Forecasting . . . 37

10 Conclusion **40**
10.1 Distribution of quiescent periods . . . 40
10.2 Short-term prediction of QPs . . . 40

11 Discussion **42**

A Appendices **44**
A.1 Correlation of variables for logistic regression . . . 44
A.2 Code . . . 45
A.3 Distribution of tasks . . . 45

1 Preface

This paper summarizes our research project for Models Laboratory, a 3rd year course given at Radboud University for bachelors students mathematics. Our project includes research into quiescent periods (QPs), a phenomenon seen at sea which are useful for ships traversing seas. Our client for this project is MARIN (Maritime Research Institute Netherlands), a Dutch company which specializes in sea and ship research. MARIN's problem is the following: In marine operations, both civilian and military, it is often needed for helicopters to land on a ship. In order to do this safely, the landing pad of the helicopter needs to stay as stationary as possible for about 30 seconds. This can be quite a challenge to achieve when large and frequent waves are colliding with the ship. Hence, it is useful to detect when the sea is quiet enough for a long period, such that a helicopter is able to land on the ship. These periods are called quiescent periods. MARIN tasked us to deeply research these QPs: is it possible to predict the next occurrence and the duration of a QP? This has been previously done in 2017 by a group of researchers at the SWI. This is a yearly conference where mathematicians work on problems from different industries. The article that followed from this conference is called Quiescent Periods for Helicopter Landing on Ships [5]. We will expand on the research that happened there.

MARIN gave us two explicit assignments to research:

- One conjecture in [5] is that the QPs follow a Poisson process, and MARIN asked us to verify this.
- We were asked if QPs can be predicted using logistic regression.

In addition to these questions, we also explored various other approaches.

We would like to thank all of our supervisors during this project. We would like to thank our supervisors at MARIN for supplying us with insights about maritime industry and for supplying is with new datasets whenever we needed them. We would also like to thank Ariyan for his contributions during this project.

2 Preliminaries

2.1 The six degrees of freedom of ship motion

When analysing QPs, it is important to mathematically characterize ship motions, so a QP can be defined based on these ship motions. Therefore, six degrees of freedom (DOF) in ship motion are introduced, and these are specified as:

- The heave z is the vertical motion, along the z-axis.
- The sway y is the lateral motion, along the y-axis.
- The surge x is the longitudinal motion, along the x-axis.
- The pitch θ is the rotation around the y-axis.
- The roll ϕ is the rotation around the x-axis.
- The yaw ψ is the rotation around the z-axis.

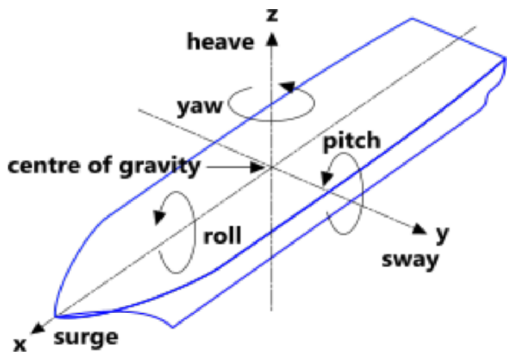

Figure 1: Illustration of the 6 DOF of ship motion. (Source: [5])

2.2 Data collection

For this project, MARIN provided us with ship motion data, generated utilising a program called MANWAV. We assume that the data is representative of actual ship motions and that the data is complete. The data has 162 columns with variables and 180000 rows with values. However, we are only interested in the time column and all the DOF around the ships centre of mass. The first dataset corresponds to an hour of simulated data, each next row corresponding to the simulated measurements of .2 seconds later than the previous. The second dataset corresponds to 5 hours of simulated data. The final datasets we worked with were 3 different sets of 10 hours of simulated data, where the difference between the last three datasets were set by their significant wave height H_s, the mean wave height of the highest third of the waves. The first of these corresponds to a calmer sea state with a significant wave height of $H_s = 3$ metres, the second to a sea state with a significant wave height of $H_s = 4$ metres and the third to the roughest sea state with a significant wave height $H_s = 5$ metres. Something that was different about the last three

datasets was that a technique called phase modulation was used. This technique introduced slight variations in the wave phases. Throughout this report with have used the dataset with a significant wave height of four metres.

2.2.1 Analysing consistency of MANWAV data

To properly determine the distribution of quiescent periods, we need sufficient data, which requires generation of long time series. With long time series comes the risk of abnormalities in the simulated data. MARIN asked us to check for repeating patterns and to check whether the long time series are still a faithful representation of wave motion.

Firstly, we were asked to find a method to detect if the data repeats itself. To do so we will try to find disjoint snippets of a certain length in the time series that have a minimal distance to one another. This problem has been studied extensively in the literature. In 2016, a group of researchers from the University of California and the University of New Mexico released a series of papers, that led to the development of efficient algorithms for motif detection, which were published in a Python package by Sean Law. The methods they developed are readily available to use in Python with the package STUMPY [10]. Doing this on our data with a long snippet length gives us a figure like Figure 2. The matrix profile at time t, which gives the minimal Euclidean distance between the snippet at time t and all other non-overlapping snippets, stays far away from zero. Indicating that, practically, the different time snippets are very different.

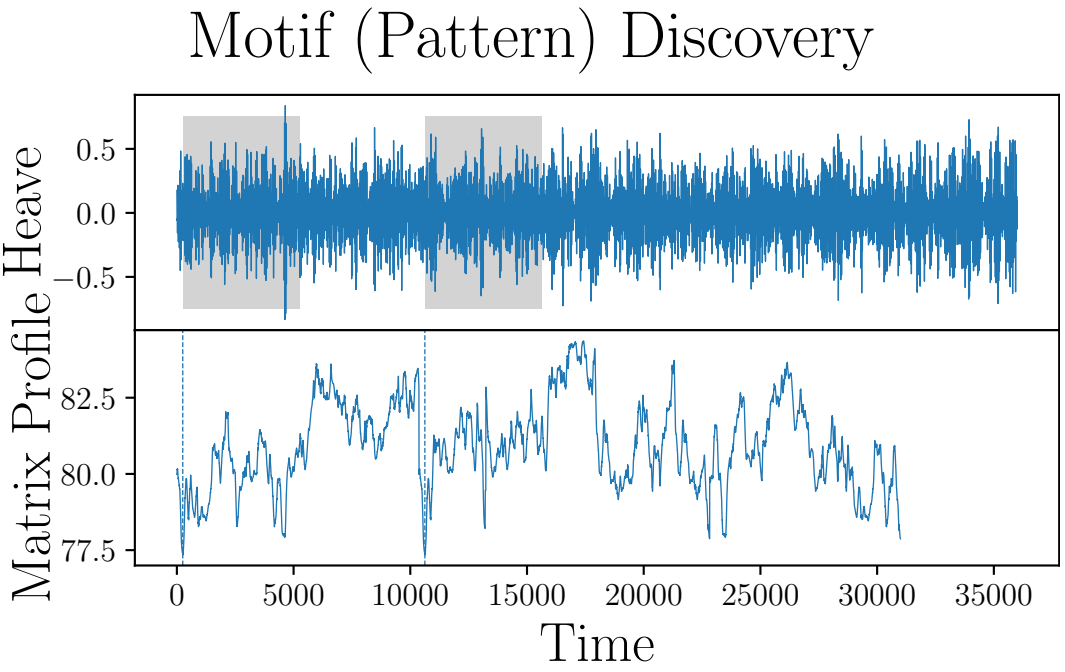

Figure 2: Motif discovery of length 5000 in heave

An advantage of the stumpy approach is that it is capable of finding more inconsistencies than just the one we were looking for. For example, large peaks in the matrix profile would indicate waves of abnormally high amplitude.

One of the solutions MARIN used to generate longer time series without modulation was to apply phase modulation when generating waves with MANWAV. One question that arose is whether or not this MANWAV still generates a faithful representation of real wave motion after this modulation. To verify this, we analysed the spectrogram of these waves in Figure 3.

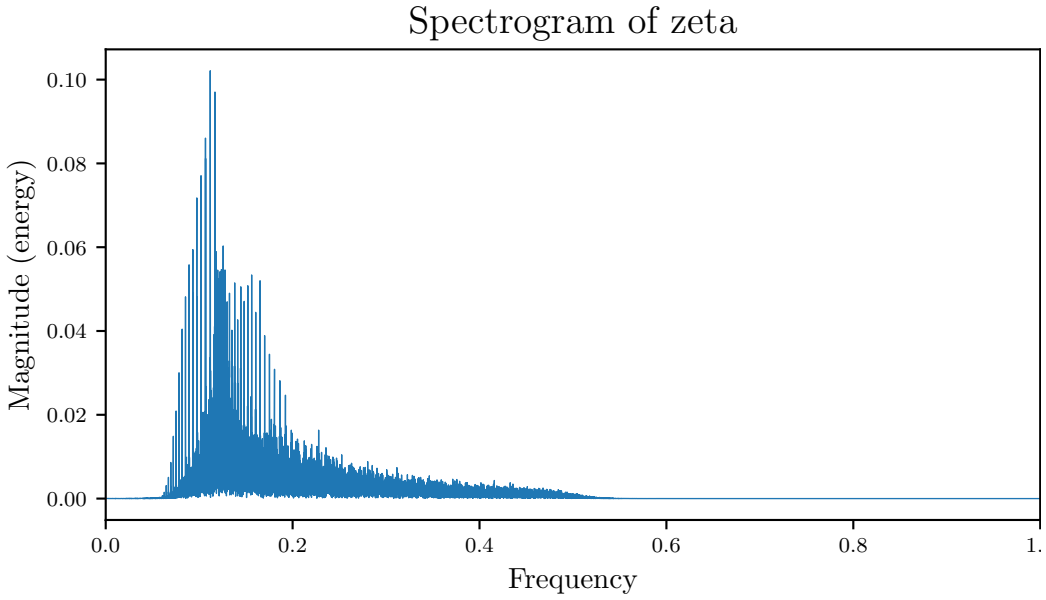

Figure 3: Spectrum of time series with phase modulation

We can see that the spectrogram in Figure 3 closely resembles the JONSWAP curve, which is a model for accurate ocean wave spectrograms. One can read more about the JONSWAP curve in [14]. As such, it is safe to assume that these waves are a good representation of actual wave data.

2.3 Definition of quiescent periods

One of the challenges of this assignment is solidly defining the notion of a quiescent period mathematically. A quiescent period is a period of more than 30 seconds, where the helicopter has a chance to land on the ship. In particular, that means that the landing deck cannot move too much during this period. A large wave during a landing attempt could have disastrous consequences.

The simplest approach would be to put a simple restriction on each of the 6 DOF of the ship motion. This is what was used in [5]. In that paper the thresholds that were used were;

- peak-to-trough amplitude of heave less than 3 m;
- single roll amplitude less than 3°;
- time duration of at least 30 s.

So only two of the six DOF are restricted. This method does the job, but it has several disadvantages, namely;

1. different restrictions are applicable for different ships;
2. it is unclear what the impact is of changing the restrictions.

We found that the second point can be mitigated by calculating the permitted range of motion in the 3d space of the helipad under these restrictions. However, the first point remained an issue. This is why we explored different options for our project. In practice, quiescent periods are defined for different situations. These are outlined in Table 1 which was published in [9]. In all applications, we used the dataset with $H_s = 4$ metres. For this dataset, we applied the limits corresponding to heavy aircraft of category 3 and in daytime, unless otherwise specified.

Table 1: Operational limits

		Helideck Category								
Aircraft Category		1			2			3		
Movements		P/R	INC	H/R	P/R	INC	H/R	P/R	INC	H/R
Heavy	Day	±3	3.5	1.3	±2	2.5	1.0	±2	2.5	1.0
	Night	±3	3.5	1.0	±2	2.5	0.5	±1	1.5	0.5
Medium	Day	±4	4.5	1.3	±3	3.5	1.0	±3	3.5	1.0
	Night	±4	4.5	1.0	±2	2.5	0.5	±1.5	2.0	0.5

P/R = pitch and roll (deg)
INC = helideck inclination (deg)
H/R = heave rate (m/s)

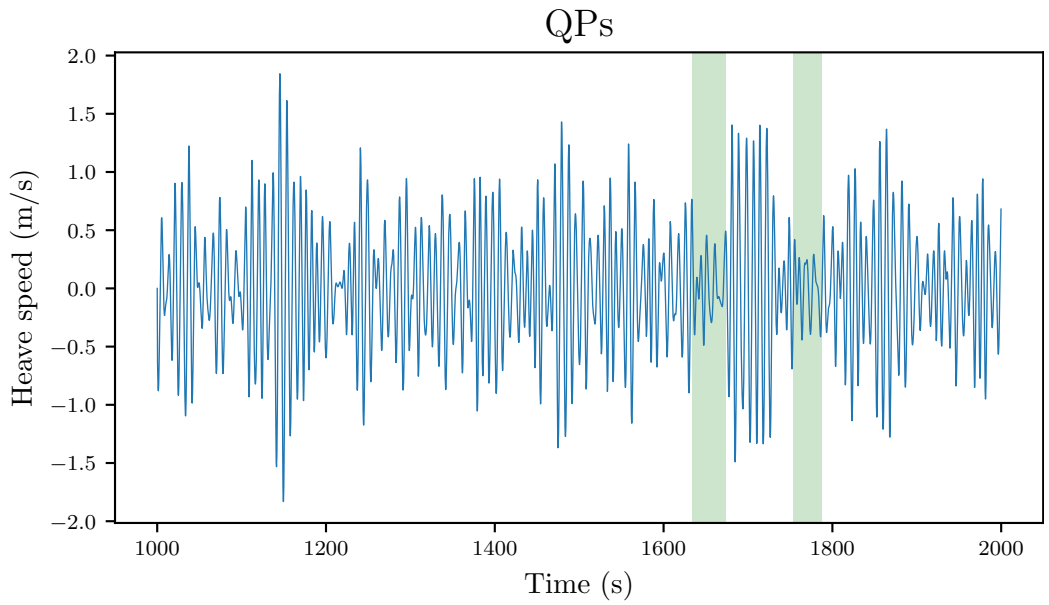

Figure 4: Detecting QP for a heavy ship at night on a helideck of category 3

To determine the quiescent periods based on the operations limits in Table 1, we calculated the motion of the helipad, based on the ship motion and the relative position of the helipad relative to the centre of mass of the ship. This was done by applying rotation and translation operations to some points on the helipad, given the position of the helideck. We have written a script in Python were we could detect the QPs in the data. As snippet plotting just the heave rate and the quiescent period can be seen in Figure 4.

Comparing the different definitions for a QP is a bit of an apples and oranges comparison. However, as expected, on the same data, with suitable choice of the restriction, the two definitions gave non-overlapping QPs. This means that there were time intervals in the data were using one definition there is a QP, but not using the other definition and vice versa. In short, the new QP detection gives markedly different results to the previous one.

2.4 Training models and calculating performance

When training models, there has to be a way to determine whether model A is better than model B. This is done using loss functions. Such a function converts the predictions of the model to a number. To generate this number, the loss function compares the predictions with the actual data. There are numerous different loss functions with different ways of converting the predictions of the model to a number. The resulting number represents the cost of the model. When training the model, the model with the lowest loss value is chosen as the best.

Since we are working with predictive models, we provide some general evaluative tools for these models. We use the definitions below in evaluating the performance of a couple of our models. Evaluating happens after training the models with the loss functions. The evaluative tools we

are going to present are to visualize the performance for the human eye. However these functions could also be used as (part of) loss functions. Usually, the whole dataset is split up in at least two parts. A train set and a test set. There is no hard rule for the ratio between these two, usually the train set is about 80% of the data, and the test set about 20% of the data. We will use these ratios when training and testing our models, as well. The idea of splitting the data is that you train your model on the train set, and, to get an estimate of the error, one tests on the test set. One should not simply test on the train set, since the model will likely perform really well on this dataset, since it was trained exactly on this data. Often, a third distinction is made, dividing the train set again into actual train set and the test set. In this setting, the model is continuously being trained on the train set, then validated on the test set. This constant validating has as a goal to determine the best parameters for the model. Then, the best model is selected and tested on the test set.

Now, we address how we actually determine the performance of a model. A common way to determine the performance of a classification model is using the accuracy, which is the percentage of correct classification, so:

$$\text{Acc} := \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{2.1}$$

where TP are the amount of data points in the test set that are positive and were correctly classified by the model as positive, and FP points in the test set that are negative and were incorrectly classified by the model as positive and the same is for TN and FN. In the case of data with imbalanced classes the accuracy could be misleading. Consider a case with a data set of 1000 points of which 900 points have class 0 and the other one 100 have class 1. If we split this dataset in a 80/20 training set and test set we have a test set of 200 points of which approximately 180 have class 0. A model that classifies everything as class 0 would have an accuracy of 90% which is really high, but one could not consider this a very "good" model. This is where recall and precision come in. Recall or True Positive Rate is the fraction of positives that are detected by the model, so numerically it is defined by:

$$\text{Recall} := \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

Precision is the fraction of correctly classified positives by the model of all actual positives, so:

$$\text{Precision} := \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.3}$$

If we go back to the example of the imbalanced classes, we can see that model that classifies everything as 0, thus negative, has a recall and precision of 0 percent, because it cannot classify anything as positive. Recall and precision are commonly combined into the F_β-score, which is defined as:

$$F_\beta := \frac{(\beta^2 + 1) \cdot \text{Recall} \cdot \text{Precision}}{\beta^2 \cdot \text{Precision} + \text{Recall}} = \frac{(\beta^2 + 1)\text{TP}}{(\beta^2 + 1)\text{TP} + \beta^2 \cdot \text{FN} + \text{FP}} \tag{2.4}$$

Where β is a positive real number which is chosen such that recall is considered β times as important as precision. In this paper we primarily use the F_1-score

The last performance metric we want to highlight is the False Positive Rate, which is the ratio of incorrectly classified positives of all negatives.

$$\text{FPR} := \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.5}$$

The FPR is especially important for detecting QP as false positives can be extremely dangerous for the helicopter landing. To make calculations in the following subsection easier, we have to introduce the Reverse False Positive Rate which is defined as:

$$\text{RFPR} := 1 - \text{FPR}. \tag{2.6}$$

This is done to have a maximum score of 1.0, when there are no false positives.

3 Distribution of Quiescent Periods

We want to investigate which distribution the quiescent periods follow. Two random variables are of particular interest to us:

- The duration of the QP, in seconds. Since every QP has a duration equal to or greater than 30, we normalize this by subtracting 30 from them.
- The time between two QPs, in seconds.

3.1 Exponential distribution

The conjecture in [5] is that the durations of QPs follow a Poisson process. The Poisson process is characterised by events that happen in an instant, and the times between the events are exponentially distributed. However, since QPs happen in 30 seconds or more, we do not think that QPs follow a classical Poisson process. Instead, we conjecture that the normalized QP durations and the times between QPs are exponentially distributed.

A random variable X is said to be exponentially distributed with rate $\lambda > 0$ if the cumulative distribution function of X is given by

$$F(x) = \mathbb{P}[X \leq x] = 1 - e^{-\lambda x}.$$

We denote this by $X \sim \text{Exp}(\lambda)$. Alternatively, the parameter can also be named the scale β, related by $\beta = \frac{1}{\lambda}$.

In order to test if our random variables are exponentially distributed, we used the Kolmogorov-Smirnov test (KS-test), which we will explain here.

Let $X_1, X_2, ..., X_n$ be independent and identically distributed random variables. Then the empirical distribution of these variables is given by

$$F_n(x) = \frac{\#\{X_i : X_i \leq x\}}{n} = \frac{1}{n} \sum_{i=1}^{n} \chi_{[-\infty,x]}(X_i),$$

where $\chi_{[-\infty,x]}$ is the characteristic function on $[-\infty, x]$. Roughly speaking, F_n is for large n an approximation of the actual distribution of the random variables. Then, given a cumulative distribution function F, the Kolmogorov-Smirnov test statistic is given by

$$D_n = \sup_x |F_n(x) - F(x)|.$$

If D_n is small, it will follow that F_n is roughly the distribution function of F. In this case, we run the KS-test with the null hypothesis that the normalized QP durations and the times between QPs are exponentially distributed.

We used Python to perform the test on the datasets of MARIN mentioned in Subsection 2.2. The result is the following: the p-values of the tests done across the datasets were not often very high (in most cases, beneath 0.30), but they do not stay too low (they are almost always above 0.05). So we have no reason to reject our null hypothesis, but the statistical evidence that the durations and times between QPs are indeed exponentially distributed is not very high.

Figure 5 is a normalized histogram of the data and QP criteria mentioned in Subsection 2.3. This gives an illustration that our null hypothesis is reasonable. For both random variables, we see that the empirical distribution of the data is not significantly different from the exponential distribution, which has the sample mean as its scale. One gets similar results with other datasets and QP criteria, as long as enough QPs are generated by the restrictions the chosen QP criteria have.

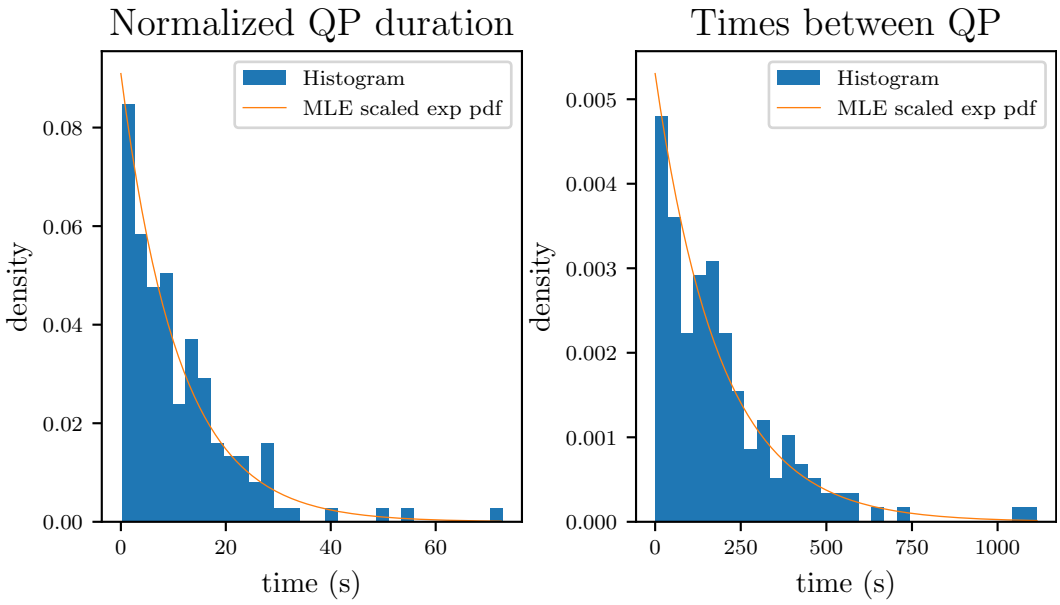

Figure 5: Histogram of QP durations and times between QPs, compared with the exponential distribution.

3.2 Hypoexponential distribution

We can conclude that the exponential distribution could be a good model to predict QPs. To model this further, we introduce hypoexponential distributions:

A random variable X is hypoexponentially distributed with two variables λ, μ, if the PDF and CDF of X is given by respectively

$$f(x) = \frac{\lambda\mu}{\lambda - \mu}\left(e^{-\lambda x} - e^{-\mu x}\right);$$

$$F(x) = 1 - \frac{\mu}{\mu - \lambda}e^{-\lambda x} - \frac{\lambda}{\lambda - \mu}e^{-\mu x}.$$

We notate this with $X \sim \text{Hypexp}(\lambda, \mu)$. One can show that if $X \sim \text{Exp}(\lambda)$, $Y \sim \text{Exp}(\mu)$ and X is independent of Y, then $X + Y \sim \text{Hypexp}(\lambda, \mu)$.

We determined the correlation coefficient of the normalized QP duration and the times between QPs, and in general it was almost always under 0.1. So we can with some certainty say that both

variables are independent. So when $X \sim \text{Exp}(\lambda)$ and $Y \sim \text{Exp}(\mu)$ represent an approximation of normalized QP duration and times between QPs, respectively where λ and μ are experimentally deduced as the reciprocal of the sample mean, we can use the hypoexponential distribution to compute the approximate probability that a QP has happened in t seconds:

$$\begin{aligned}\mathbb{P}\big[\text{a QP has happened in } t \text{ seconds}\big] &\approx \mathbb{P}\big[X + Y + 30 \leq t\big] = \mathbb{P}\big[X + Y \leq t - 30\big] \\ &= 1 - \frac{\mu}{\mu - \lambda} e^{-\lambda(t-30)} - \frac{\lambda}{\lambda - \mu} e^{-\mu(t-30)}.\end{aligned} \tag{3.1}$$

Computing the approximate probability that one of more QPs happen in a certain time frame requires the theory of hypoexponential distributions with more than two variables.

Figure 6 gives a normalized histogram of $X + Y + 30$, under the same circumstances as Figure 5, and the plot of

$$f(x) = \frac{\lambda\mu}{\lambda - \mu}\big(e^{-\lambda(x-30)} - e^{-\mu(x-30)}\big),$$

which is the PDF of the theoretical distribution given in (3.1). Again, they do not seem to differ too much.

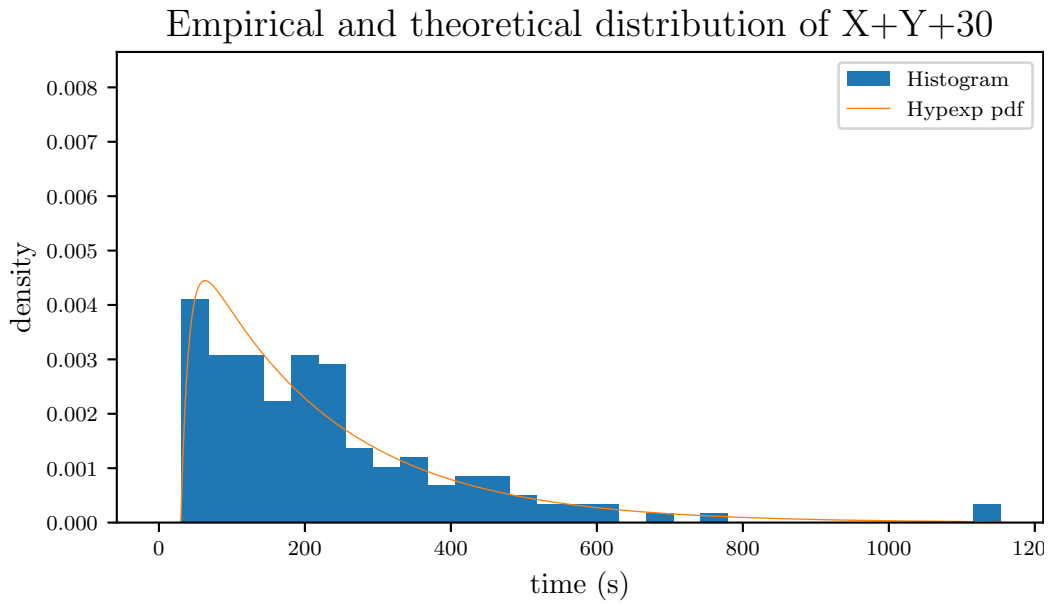

Figure 6: Histogram of a QP cycle and the theoretical distribution of it

One drawback of modelling QPs in this way is that the parameters need to be estimated experimentally, which requires a lot of auxiliary data about the sea the ship is traversing.

3.3 Finding a better model

Whilst statistics is an incredibly useful tool, it is fundamentally not possible to prove that our data follows a certain distribution using solely statistics. We have shown that, statistically, we

cannot reject our null hypothesis. But the statistical evidence that our null hypothesis is indeed true is not very high. This means that there probably exists a better distribution that models the distribution of the QPs. However, this model may be more complicated.

One famous quote from statistician George Box is that *all models are wrong* [2, p. 424], which means that no model can exactly fit the data. Using the exponential distribution to model the QPs may be wrong, but it may still be useful, as the error between the data and the model is not very high.

4 Logistic Regression

Logistic regression is a probabilistic classifier that predicts binary classes (0,1) based on a given input. A probabilistic classifier is a model that takes some input and returns the probability that this data point belongs to a specific class. The input to the logistic regression model can consist of multiple variables. The model uses a function with parameters a_i that are trained using labelled data. The following function is the classifier function.

$$P(z) = \frac{e^z}{1 - e^z} \tag{4.1}$$

where p is the probability that the input belongs to class 1, and z is a linear combination of the input variables:

$$z = a_0 + \sum_{i=1}^{n} a_i x_i \tag{4.2}$$

where $x_1, ..., x_n$ are the input variables, and $a_1, ..., a_n$ the coefficients and a_0 the offset. When training the model, the coefficients a_i and offset a_0 are fitted.

Since the output of the model is binary while $P(z)$ returns a value between 0 and 1, a threshold value has to be assigned. When the predicted value $P(z)$ is greater than this threshold value, the prediction is class 1. When $P(z)$ is lower, the prediction is class 0. This threshold value is 0.5 by default, but it can be adjusted to the particular context or for fitting.

4.1 Our approach

The goal of this application is to quickly predict the appearance of quiescent periods in the very near future, where the very near future implies a maximum of a few seconds.

The definition used for a quiescent period is for aircraft category 3 with heavy movements at day, as stated in Table 1. So, for at least 30 s, the pitch and roll should be less than $\pm 2°$, the helideck inclination less than 2.5°, and the heave rate less than 1.0 meter per second. As stated in the previous section, logistic regression has a binary output. In our case, that is a 0 when the simulated system is not in a QP and a 1 when it is. This is our first attempt at a QP classification, and we create a vector of 0s and 1s and attach it to our dataset as a column. This way, for every row in our dataset, we also have a QP value that determines if we are currently in a QP by the definition. This information will be used to train the classifier on. However, our goal is to predict the upcoming QPs, not to say whether we are in a QP right now, since this prediction would come too late. So, predicting the last five 1s of a QP correctly is useless, as we then only have a second left of the QP instead of the wanted minimum of 30. So, to meet our goal, we have to redefine how to assign 0s and 1s.

A first idea is to only assign 1s to the data points in a QP when the upcoming 30 seconds are still part of that QP. For example, suppose a QP lasts 37 seconds. Then only the first 37-30 = 7 seconds are getting assigned 1s. This will now be called QP_1. A second method of assigning 0s and 1s is inspired by paper Quiescent Periods in Helicopter Landing on Ships [5]. In this paper, Figure 7 was published. In this figure, many time series of heave extrema of waves are laid on top of each other. All of these have the property that a QP starts at the red vertical line. From this figure, it can be seen that right before the start of a QP there seems to be a sudden peak in the extrema of the heave amplitude. This seems to be a clear indicator for the start of a QP and might be well recognizable by the logistic regression model. To base the logistic regression on

this pattern, we want to assign 1s only to the start of a QP. So, we introduce a new QP vector called QP_2 which assigns a 1 to five data points at the start of every actual QP, and everywhere else a 0. We choose five data points since this seems to be a sufficient amount to capture the clear peaking pattern right before the start of the QPs.

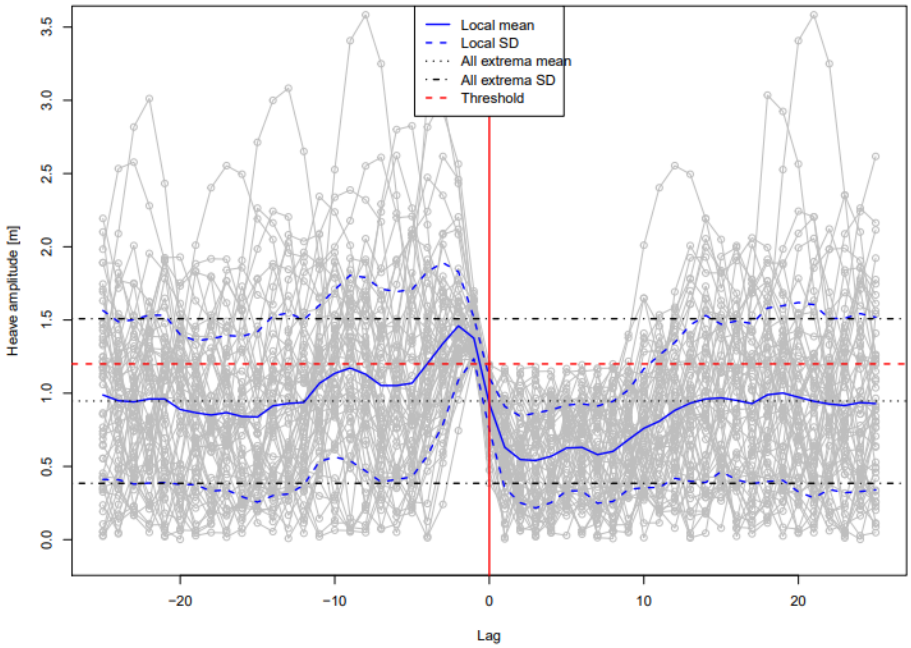

Figure 7: Different sequences of heaves laid on top of each other. All have a QP starting at the red vertical line.

For both QP_1 and QP_2, we have made multiple models with different inputs: One with heave amplitudes as input, one with heave velocity and one with the extrema of the heave amplitudes. To find the best model for each version, we optimized over the lookback window. The lookback window determines how many previous values we use for our prediction. For example, for the model with the heave amplitudes as input, we varied the number of previous heave amplitudes used as input. This ranged from 1 to 30 of the last heave amplitude observations. We also optimized the threshold value.

This optimization was done in two steps with 2 different loss functions. We train the models using the *glm* function from the r-package *stats*. This function can fit logistic regression models and uses Log Loss, also known as Cross-Entropy loss. The formula for this loss function is

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \cdot_i + (1-y_i)\cdot(1-p_i))$$

Where N is the number of data points, y_i the actual class of data point i and p_i the probability of data point i to be of class 1. This is loss function is commonly used for classification models. It assigns more cost to wrongfully predicted high value more than other loss functions. This is very useful for our model since a falsely predicted 1, a false positive, can cause a helicopter to crash, and a false negative, wrongfully predicted 0, only delays the landing of the helicopter.

However, when training the models, it was seen that all probabilities for data points to be of class 1 were very close to each other. Thus, when optimizing over the lookback window, we used a different loss function in which we could also adjust the threshold values to prevent predicting only zeroes or only ones. This loss function takes the predicted class as input, while the Log Loss takes the probability $p(z)$ of a data point to be of class 1 as input. This function counts 1.5 for a false positive and 0.5 for a false negative. This is shown in the formula below. Here, N is the number of data points and x_i is the i^{th} prediction, either being FP, TP, TN or FN.

$$Loss = \sum_{i=0}^{N} (\mathbb{1}_{FP}(x_i) \cdot 1.5 + \mathbb{1}_{FN}(x_i) \cdot 0.5)$$

The difference is again based on the fact that a false positive is much worse than a false negative.

To evaluate the models, the data have been split into a training set and an evaluation set. The training set consists of the first 80% of the data and the evaluation set of the last 20%. That is, (4.1) is fitted to the first 80% of the data points and evaluated by entering the last 20% into the fitted function and comparing the predictions with the actual data.

4.2 Results

The results of the optimization per model are shown in Table 2.

Table 2: Confusion matrices of the best models per category optimised over lookback window and threshold. FP is a false positive (predicted 1, reality 0), FN a false negative (predicted 0, reality 1), TP a true positive (predicted 1, reality 1) and TN a true negative (predicted 0, reality 0).

Model type	FP	FN	TP	TN
QP_1 heave amplitudes	0	4175	0	22825
QP_1 heave velocity	0	4175	0	22825
QP_1 extrema of heave amplitudes	0	215	0	1129
QP_2 heave amplitudes	0	230	0	26770
QP_2 heave velocity	0	230	0	26770
QP_2 extrema of heave amplitudes	0	4	0	1340

All optimized models had a lookback window of 1. The models QP_1 had an optimal threshold of 0.2 and the QP_2 of 0.1.

As can be seen from the table, in each model, only 0s are predicted. From these results, one might think to change the threshold to prevent predicting only zeroes or only ones. When doing this, the models jump from only predicting zeroes to only predicting ones quite quickly. If we use a threshold in between, the model predicts far more false positives than false negatives. Thus this still does not result in a useful model. This is shown for QP_1 with input heave amplitude and lookback windows of 30 and 1 in the Table 3. It is only shown for one model with 2 different lookback windows but all models follow this same behaviour with all lookback windows.

Table 3: Confusion matrices of the best models with constant lookback window and changing threshold.

Lookback window	Threshold	FP	FN	TP	TN	Loss
30	0.18	0	4175	0	22825	2087.5
30	0.16	2627	3797	378	20198	5839
30	0.14	22824	0	4175	1	34236
1	0.16	0	4175	0	22825	2087.5
1	0.15	4175	0	22825	0	34236

From these results, we conclude that applying logistic regression on the heave values of the ship does not result in a working model for predicting quiescent periods. We have tried a lot of different options for input, and none of these resulted in a model that showed any promise of being remotely useful. Thus, the question arises of why this is not a working method.
Logistic regression is based on a few assumptions. One of them is that there is no multicollinearity. This means that the different input variables should not have a high correlation. However, in this problem, the current heave and n heaves before are taken as input. These variables are expected to be highly correlated. One can confirm this is the case by look at Table 7 in Appendix A.1. A second assumption is that the observations are independent. However, since the used data is a continuous simulation, observation n is highly dependent on observation $n-1$. This is also likely the cause of the failure of the models with the heave extrema as input, where the model fails to recognise the peak seen in Figure 7. Since these assumptions are not met for other inputs either, like the roll, pitch and yaw, we do not think that logistic regression can be a solution for the problem of predicting the appearance of a quiescent period in the near future.

5 Random Forest

The random forest is a widely used classification model. This model makes a specified number of random decision trees of a limited depth. A decision tree is a very simple way of deciding a class for a given set of inputs. It is essentially a couple of if-else statements on the input features that lead to the output. The depth of a decision tree is the largest number of if-else statements that are checked before reaching an output. A random decision tree is a decision tree that, for each if-else statement, picks a random attribute/property to split on. For example, when we are at a given time point, an attribute could be the last heave extrema, and the condition could be: is the last heave extrema smaller than 0.5? The decision of the random forest is simply the majority vote of all the random decision trees it encompasses. The following figure neatly shows the working of a random forest.

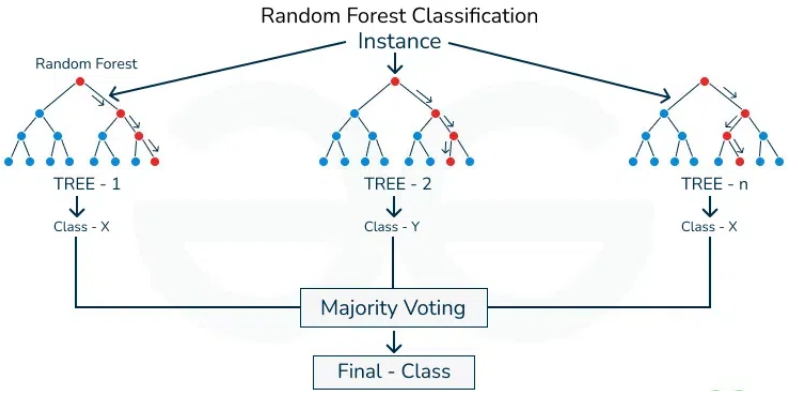

Figure 8: Working of a random forest

5.1 Our approach

We use 100 decision trees for our random forest with a maximum depth of 20. These are common parameters for random forests. The output of a decision tree will be 1 if the decision tree predicts that the next 30 seconds will be safe, according to the QP thresholds, and 0 else. We have also tried other values for the depth than the standard 100, 20 seemed to work the best. We use a specialized version of the random forest classifier called the AdaBoostClassifier. This classifier trains random decision trees like the normal random forest, but in between training of different decision trees it evaluates the trained decision trees and evaluates data points that the trained decision trees perform the worst on. These data points then have a higher probability of being trained on for next decision trees to correct errors. As input, we will give the classifier a feature vector. This feature vector takes the past 100 heave values and extracts the 3 most recent maxima from them. The random forest trains only on these 3 maxima. We have also tried extracting more extrema, and also including the extrema from more variables, but did not improve performance. We train on a train set, which is 80% of the data, and test on a test set, which is 20% of data.

5.2 Results

We discuss the results from the classifier with parameters and decisions described in the previous section. The results of random forest were not great. To avoid overlap between the test and the

class	precision	recall	f1-score	support
0.0	0.90	0.97	0.94	81164
1.0	0.16	0.06	0.08	8739

Table 4: Best results random forest

train set, we just set the first 80% of the data as train/evaluate set and the last 20% as test set. The following are the results of training a model on the train/evaluate set and finding the best parameters, then testing this model on the test set, with as feature vector the past three heave values. We have used the QP vector known as QP_1, i.e., we label a feature vector with a 1 if and only if the coming 30 seconds are considered a QP, as defined in Section 4.

We see in the output that 8739/89903 points in the output vector are marked as a QP before prediction. Of those 8739 points, the random forest classifier recognized $0.06 \cdot 8739 = 524$ as an instance of the QP_1 vector. Moreover, 16% of all the times when the classifier said it recognized an instance of the QP_1 it was correct. Note that the classifier saying it is recognizing an instance of the $QP1$ vector is equivalent to saying that the helicopter could start its landing procedure at this moment, since we will most likely be in a QP for the next 30 seconds.

We see that the model evaluated above is not a very useful one. It is hard to say why this is the case, since random forest does not assume a distribution and it is generally a pretty versatile model. Our best guess is that being able to effectively predict QPs requires a deeper and long-term understanding of the patterns of the data. The random forest model has no special mechanism like the LSTM-model to retain any important information about past waves. Additionally, the random forest model has no sense of a temporal order, which would, e.g., inherently use the fact that heave values that are most recent are probably more relevant to predicting an upcoming QP than heave values that are a few minutes ago. Also, since the random forest model is piecewise constant, it has difficulty predicting continuous dynamics.

6 Long Short-Term Memory (LSTM)-models for QP classification

The last kind of classification models we have tried are machine learning models. These models use artificial neural networks to learn from data and then generalize their knowledge to unseen data. A neural network is inspired by the working of neurons in the human brain. An artificial neural network consists of layers of artificial neurons. Each neuron receives signals from all the neurons in the previous layer, processes them, and then passes a new signal forward to the next layer of neurons. A signal in this case is a real number. Every connection between two neurons in neighbouring layers has a weight, which is a real number. To compute its output, a neuron adds together all the signals it receives, then applies a non-linear, real-valued activation function to this sum. The result is then multiplied by the weight of the connection and sent forward to the subsequent layer. The neural network "learns" by adjusting these weights. During training, it processes many samples of training data and assesses its performance by computing a loss. The network then tweaks its weights to minimize this loss, thereby improving its ability to make accurate predictions on new, unseen data. To minimize the loss, we typically use an algorithm called gradient descent. This algorithm calculates the "direction" in which the weights should move to reduce the loss. To do this, it first performs backpropagation, a procedure that computes the gradients. The gradients are the rate of change of the loss function with respect to the weights. Using these gradients, an optimizer then adjusts the weights by a small amount in the direction that most reduces the loss. Over many training steps, called epochs, the process results in a network that performs well at the task it was trained for.

The artificial neural networks we described previously are designed to process inputs, which are not sequential, i.e. each point of the input is treated independently of all previous ones. However, our data of waves is highly sequential. To handle such sequential data, a different kind of architecture called recurrent neural networks (RNNs) was developed. RNNs have a form of memory that lets them maintain a hidden state, allowing information to flow forward through time and influence future predictions. The dimensions of this hidden state, often called the hidden size, determine how much information the network can retain and how complex patterns it can learn. Nevertheless, traditional RNNs suffer from issues such as vanishing or exploding gradients, making it hard for them to learn long-range dependency structures. To address these shortcomings, specialized variants of RNNs, such as Long Short-Term Memory (LSTM) networks, were introduced.

LSTM units have a more complex structure than traditional RNN units. They were originally proposed by Hochreiter and Schmidhuber in 1997 [7], and were designed with the purpose of learning the patterns of time series and predicting their future behaviour. LSTM networks proved to be fast learners, and more accurate predictors compared to state-of-the-art models at the time. In 1999, Gers et al. identify a weakness of LSTM networks. Namely, the possibility of the state growing indefinitely, in the case that the input is an infinite data stream without explicit markings in the stream where the state is reset [6]. As a solution, the authors of [6] propose the forget gate. This is a new part of the LSTM unit, which forgets information when the gate is triggered to prevent the state from breaking by being overloaded with information.

6.1 Our approach

In this section, the focus will not be on direct QP detection, but on threshold detection. By threshold detection, we mean that we could detect if a variable surpasses a certain threshold in

the future. If these threshold models have a reasonable accuracy, then a QP can be detected if all or a majority of the variables that define a QP stay below the threshold for the defined time. For our models, we used the dataset and limitations as described in Subsection 2.3. The threshold for heave rate is 1.0 m/s, for pitch ± 2.5 deg and so forth. We also looked at the heave and we used a threshold of 1.0 m. We will look at LSTM-models for binary classification, i.e. layers of interconnected LSTM-cells that have information from one variable as input and a binary output. The output will be zero if the variable exceeds the threshold in a time interval in the future or one if the variable stays below. The time interval in the future will be henceforth referred as the prediction window. Another approach for QP classification is using methods developed from multivariate time series analysis. For these methods the input of the model is information from multiple variables of ship movement, instead of one. These methods lie out of scope for this paper. There are some references given to these methods in section 9 that are outside the context of ship movement.

We chose two different inputs: sequences of extrema, as was suggested in [5], and time series of ship movement of one variable. For the extrema, we wrote a Python script to make a new dataset to feed the model. The entries of this dataset are sequences of k extrema with a label that is either 0 or 1. If T_i is the timepoint of the i^{th} extrema in the timeseries of the variable, then we take the extremas at $T_{i-k}, \ldots, T_i$. We give this sequence a label 1 if variable stays below the threshold for the upcoming s seconds from T_i and a label 0 if otherwise. To prevent overlap in the data points, we increment i by k for our next data point. This process is visualized in Figure 9. A similar procedure is followed for the wave data. If t is a timepoint in the time series and $v(t)$ the value of the variable at time t, a sequence of $(v(t-x), \ldots, v(t))$ is taken with a label. This procedure is seen in Figure 10

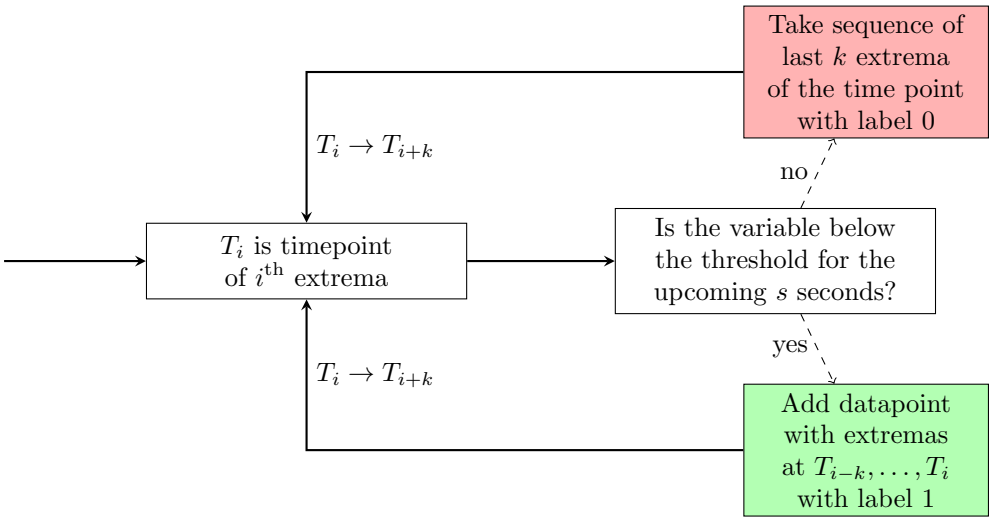

Figure 9: Data process for heave data

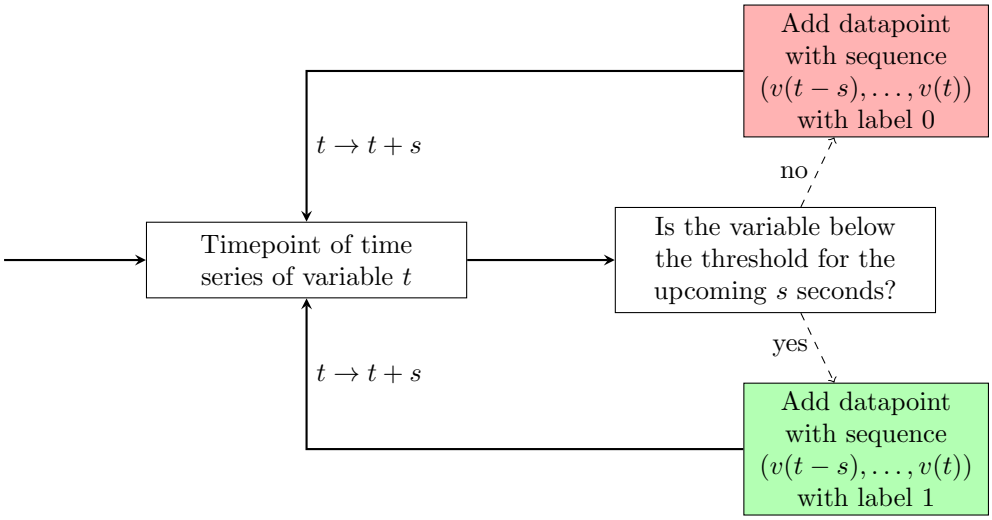

Figure 10: Data process for extrema data

As a framework, we made a LSTM-model in Python using PyTorch [13]. One of the benefits of PyTorch is that the training can be done on a GPU. This framework can be found in the GitHub repository. For the loss function we use Binary Cross Entropy with positive weight proportional to the amount of data points with label 0 divided by the amount of data points with label 1, to counteract the class imbalance by giving more weight to positive instances. Binary Cross Entropy is the industry standard loss function when it comes to binary classification problems. For the learning rate we have a scheduler called cosine annealing with warm restarts. The scheduler fluctuates the learning rate during training. This particular scheduler starts at higher learning rate and decays. After a certain epoch it resets to the starting learning rate to decay again. This cyclic pattern of decaying and resetting ensures that the loss does not stay in a local sub-optimal minimum. The optimizer that we chose is Adam, also a standard choice. The amount of layers, epochs and the hidden size can be chosen depending on the input of the model.

6.2 Results

In this subsection we will look at the results of the wave models and the extrema models. The wave models are based on the ship movement of one variable and the extrema models are based on sequences of extrema of one variable. These setups were described in the previous subsection.

6.2.1 Wave models

In our first implementations of the wave models, we found that the F_1 and FPR score of the test set deteriorates as the time interval increases. This could have multiple explanations. One explanation could be that the length of the sequence is too short. In different LSTM-models to predict real-time ship motions using time series of different motions, a ratio of 1:10 of output to input was used. This LSTM-models will be covered in Subsection 9.3. If this ratio is used, we would need 300 seconds of ship movement data to detect if a threshold is surpassed in a time interval of 30 seconds. This would result in too little data points to feed the model. Another

explanation could be found in the hyperparameters of the model. Implementing more layers and increasing the hidden size could improve the performance for larger time intervals, as it can detect more patterns in the data. Therefore we want to answer the following questions:

1. What is the influence on the performance of a LSTM-classification model, as the hidden size and number of layers are increased?
2. What is the influence of a longer prediction window on the performance of a LSTM-classification model, with the same hyperparameters?
3. What is the influence of a longer lookback window on the performance of a LSTM-classification model, with the same hyperparameters?

To answer the first question, we took the dataset and prepared this as described in Figure 10 with a lookback window of 50 seconds and a prediction window of 15 seconds. We did this for the variables heave rate, heave, roll and pitch. The obtained datasets each have 719 entries and are split in a training sets (80% of each dataset) and a test sets (the remaining 20%). We tried different combinations of layers and hidden sizes. The rest of the hyperparameters as described in Subsection 6.1 were kept constant. For every combinations we trained the model for 250 epochs and find the epoch with the best average of the performance metrics F_1-score, accuracy and RFPR , as defined in Subsection 2.4, on the test set. The results for the heave rate can be seen in Table 5.

Model type	Accuracy	F_1 score	RFPR	Average	Best epoch
(1, 16)	0.701389	0.661417	0.627659	0.663488	216
(1, 32)	0.7361111	0.698412	0.659574	0.698032	128
(1, 64)	0.701388	0.619469	0.702127	0.674328	142
(1,128)	0.6875	0.628099	0.663043	0.659547	32
(2, 16)	0.722222	0.718309	0.638554	0.693028	81
(2, 32)	0.701388	0.612612	0.676767	0.663589	75
(2, 64)	0.673611	0.629921	0.655172	0.652901	179
(2,128)	0.71527	0.577319	0.750000	0.680865	225
(3, 16)	0.708333	0.631578	0.709677	0.709677	4
(3, 32)	0.763888	0.706896	0.711340	0.727375	83
(3, 64)	0.75	0.708695	0.784313	0.748002	211
(3,128)	0.763888	0.716666	0.720430	0.733661	18

Table 5: Performance metrics for different combinations where model type is (number of layers, hidden size) for the wave model with the variable heave rate.

These tables for the other variables are not in this paper, as these would occupy too much space. One can see the model type (3, 128) has the best results, but it does have one massive disadvantage: training time. It took 15 minutes to train the (3,128)-model, which is not that long but for the other questions the model needs to be retrained. For the variables pitch and roll, all the models preformed very poorly. They always classified either everything as 0 or everything as 1. Otherwise is true for heave. Almost all the models preformed slightly better on the heave then heave rate. The best performing model for the heave was (3, 32), but only scarcely better then (3,64): 0.765311 versus 0.753282. So our model of choice is (3, 64) as it has quite a lower training time than the (3, 128) and the best average performance for heave and heave rate. There does

not seem to preform better when the hidden size is increased and the layers stay constant or vice versa. However, bigger models are preforming slightly better, but only hardly.

To answer the second question we almost have the same procedure as question 1. It only differs in that we now take a fixed number of layers and hidden size, but take different lookback window sizes. We let the prediction window range from 5 to 30 seconds. The number of layers are 3 and the hidden size is 64. The results can be seen for the heave rate in Figure 11. In the figure, a clear downward trend for all performance metrics can be seen. It seems that there is a downward trend that is strongest for the RFPR. This pattern was also prevalent for heave, but the dip came around the 14 second mark. The model performed poorly for every prediction window for pitch and roll. In training the model kept classifying either everything as 0 or everything as 1. To conclude, the performance of the model only decreases when the prediction window is shorter than 15 seconds. After the 15 seconds the performance keeps oscillating for a reason we could not find.

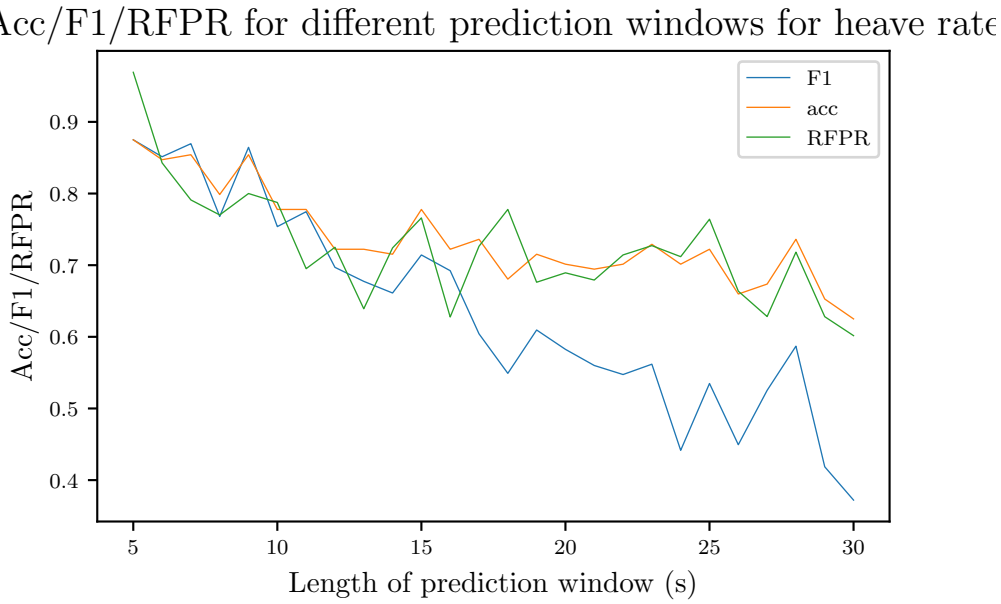

Figure 11: Performance metrics for different lengths of prediction windows for the wave model with heave rate

To answer the third question, we used the same procedure as for question two, but now we have a fixed prediction window of 12.0 seconds and let the lookback window range from 30 until 70 seconds. The prediction window of 12 seconds was chosen, because that is the prediction window where the performance of the models seems to decrease heavily. The results for the heave rate can be seen in Figure 12. There is no clear upward trend visible that we would have expected. The performance is only getting better after 50 seconds and it is still oscillating. For heave, there was also no general upward trend, but the oscillation was less heavy than the heave rate. So a longer lookback window does not necessarily increase performance. However, this should be tested for even longer lookback windows than done here.

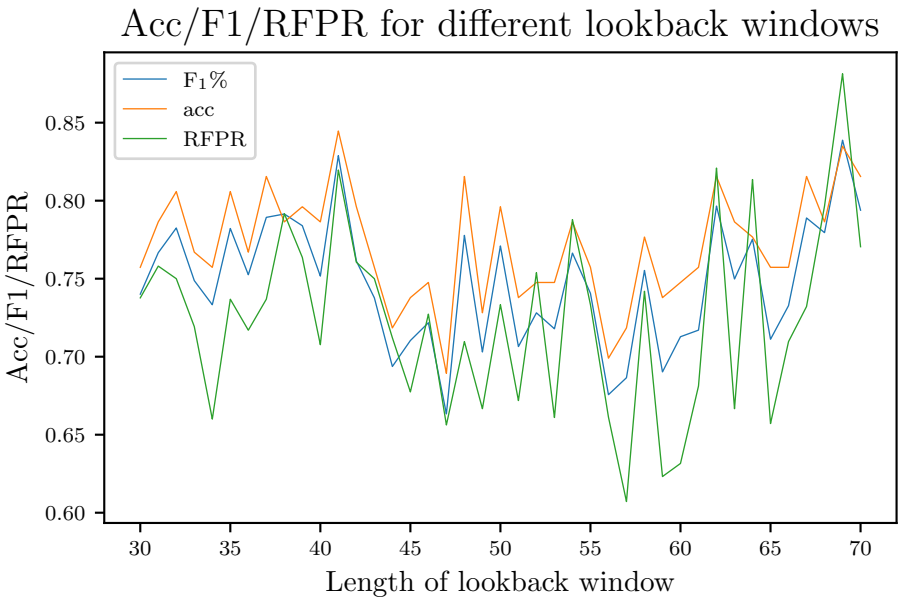

Figure 12: Performance metrics for different lengths of lookback windows for the wave model with heave rate

6.2.2 Extrema models

To compare the extrema models to the wave models, we want to answer the same three questions posed in Subsection 6.1, but in the context of the extrema models.

For the first question the same procedure was followed as in question 1 for the wave models. However, the data was prepared as described in Figure 10 instead of Figure 10. The last 8 extrema were chosen as input. The results can be seen in Table 6 for the variable heave rate. The best performing model is the (3, 32). The differences between all models are marginal and there is no clear outlier. Increasing the hidden size only yields a better performance if the amount of layers are one or two.

For the next question the same procedure was followed as in question 2 for the wave models. Model type (3, 32) was chosen. The input for the model was the last 8 extrema, where the prediction window ranged from 5 to 30 seconds. The results for the heave rate can be seen in 13. The other variables are not displayed in this figure. There was a general downward trend, which was expected. The decay seems to be linear and not exponential. This was the case for all variables. For roll and pitch was the performance much better than with the wave models. It didn't assign everything as 0 or 1, but is was slightly worse than heave and heave rate. So longer prediction windows yield lower performances and this relation could be linear, but this should be tested with more data.

For the third question the same procedure was followed as in question 2, but now the prediction window is held constant and the number of extrema changed from 3 to 12. The results for the heave rate can be seen in 14. There is no general upward trend, but after 8 extrema it stabilizes.

Model type	Accuracy	F_1 score	RFPR	Average	Best epoch
(1, 16)	0.782426	0.737373	0.765100	0.761633	67
(1, 32)	0.769874	0.739336	0.736111	0.748440	80
(1, 64)	0.786610	0.760563	0.748251	0.765142	68
(1, 128)	0.794979	0.754193	0.751773	0.766981	63
(2, 16)	0.774058	0.742857	0.685897	0.734271	156
(2, 32)	0.786610	0.784810	0.673758	0.748393	166
(2, 64)	0.786610	0.764976	0.729166	0.760251	145
(2, 128)	0.794979	0.753768	0.756578	0.768442	80
(3, 16)	0.799163	0.755102	0.740506	0.764923	139
(3, 32)	0.785242	0.783488	0.745624	0.771451	161
(3, 64)	0.715481	0.626373	0.716981	0.686278	211
(3, 128)	0.774058	0.750000	0.697986	0.740681	134

Table 6: Performance metrics for different combinations where model type is (number of layers, hidden size) for the extrema model with variable heave rate.

This was the case for all variables. So feeding the model more extrema does not seem to yield better performance. However, this should be tested with more data and more extrema.

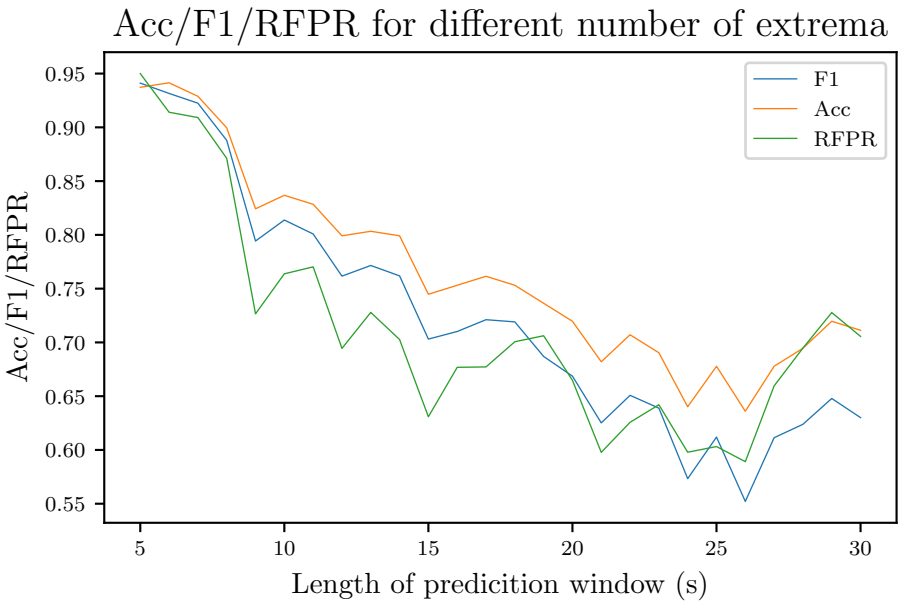

Figure 13: Performance metrics for different lengths of prediction windows for the extrema model with heave rate

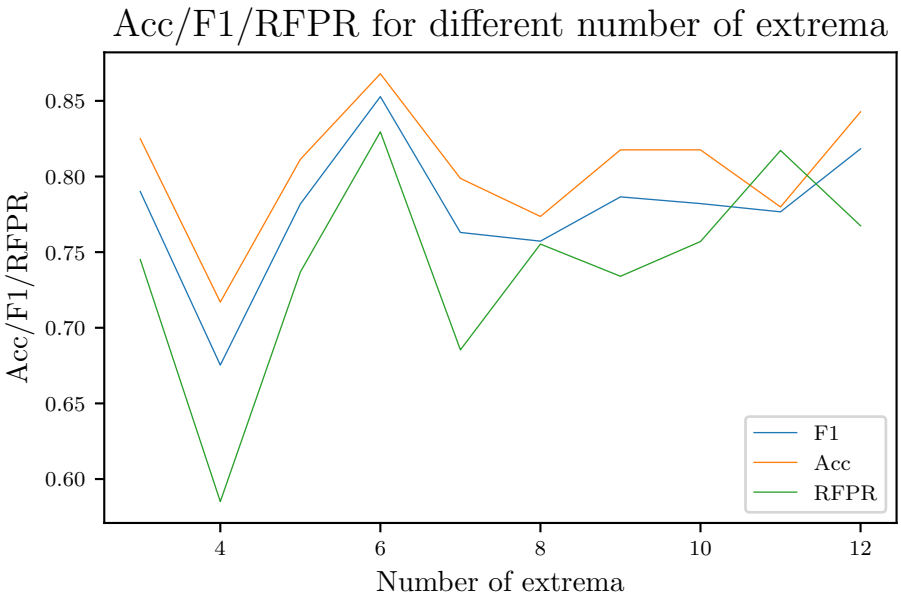

Figure 14: Performance metrics for different lengths of lookback windows for the extrema model with the variable heave rate

7 ARIMA

ARIMA stands for autoregressive integrated moving average. It is a class of models used for time series forecasting. The autoregressive part of the model looks at the previous values to predict the values we want to forecast, we denote the amount of values we look back at with p. Furthermore, we want to remove trends in the data. This is done by taking the difference between consecutive observations. This is called differencing of order $d = 1$. If this does not remove the trend we can difference again by taking the difference between consecutive observations in the already differenced data. This is differencing of order $d = 2$. We can do this multiple times, which is thus differencing of order d. Differencing can also be used to remove seasonality in the data. Seasonality is a trend that occurs on regular intervals. Lastly, the moving average models the error term from the previous time steps. This has order q if we look back at the previous q values. The ARIMA model is then notated as ARIMA(p, d, q).

We wanted to use ARIMA to forecast the wave time series to look if it could predict a quiescent period by looking at trends of previous waves. If the ARIMA model could do this, we could further explore this to do this in real time to predict a quiescent period.

7.1 Our approach

The time series we trained the ARIMA model on was the heave. The heave is the variable that describes waves the best as it signals a clear wave height by looking at the ship's vertical displacement.

For the heave variable, we do not need to difference our data. As the heave measures the vertical displacement, there is a clear equilibrium, hence no overall trends in the heave variable. Furthermore, there is no need to remove seasonality. The data we are looking at does have seasonality, as waves have a recurring pattern, however it is this recurring pattern that we are interested in as a change in this pattern would indicate a possible quiescent period. We would expect some sort of seasonality by looking at when a quiescent period occurs, but these are irregular, ergo not seasonal. Hence we can use an ARIMA model with order $d = 0$.

To determine the order of p and q we use ACF (autocorrelation function) and PACF (partial autocorrelation function) plots. The ACF plot measures the correlation between the time series and its lags and can be used to determine the order q. The lags are the previous values in the time series. The PACF plot measures the correlation between the time series and lags of itself, excluding contributions of intermediate lags. This plot can be used to determine the p order. The rule of thumb in these plots is if there is a cut-off in the correlation, the amount of lag this happens at is the order of p and q.

As can be seen in Figure 15, the autocorrelation function does not have a strong cut-off, but a sinuous wave form. This might suggest that the data is seasonal, but as we have discussed above, this is not the case. Not having a cut-off implies that we can take $q = 0$. In Figure 16 we have the partial autocorrelation function. This function does have a sharp cut-off. Although it is an interpretation of the plot, we took $p = 2$ as order because from the 3rd lag there is a strong decline in the correlation. In conclusion, we used a ARIMA$(2, 0, 0)$ model.

7.2 Results

The model uses a time series of 200 data points as this is the optimal amount of data points for an ARIMA model to fit to. The model predicts the wave-like motion of the data points, but does

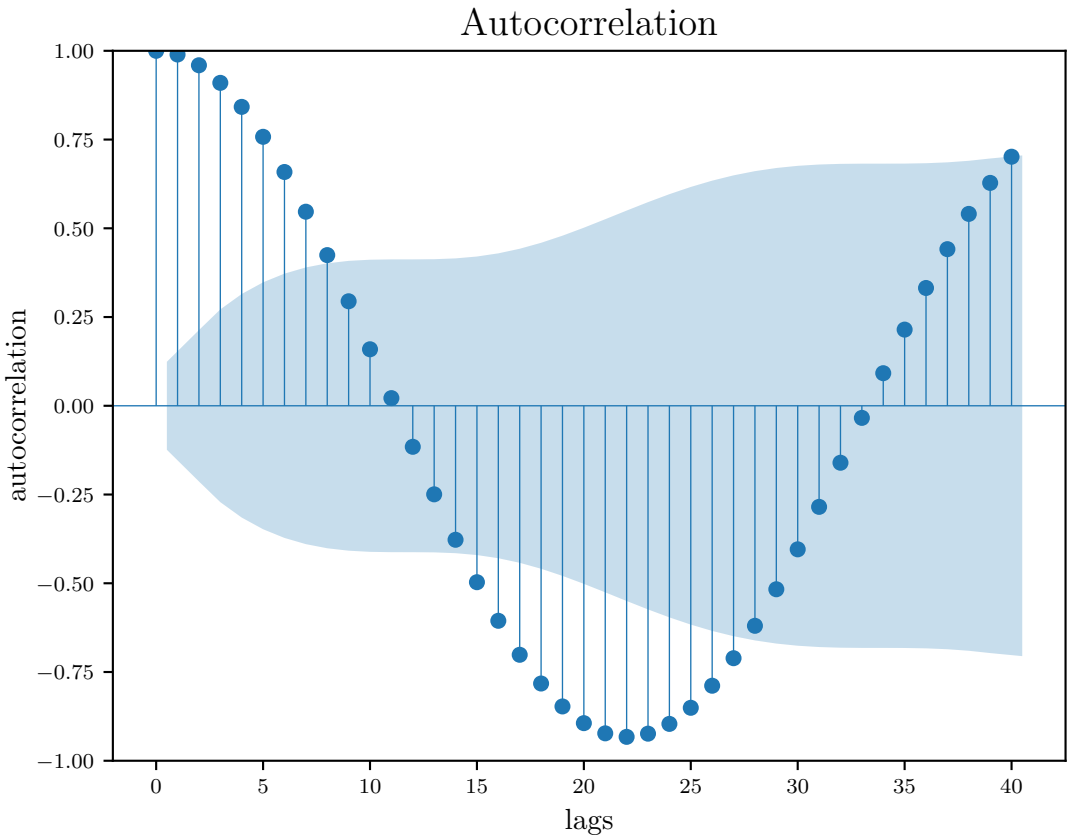

Figure 15: ACF plot for 40 lags. The dots represent the correlation between the current value and the lags. The blue shaded area represents the confidence interval.

not predict the changes in the amplitude and phase of these waves. This can be seen in Figure 17 and Figure 18. These are two plots of an ARIMA forecast with two different starting points in the dataset. There is a selection bias in these plots, as we have chosen plots that show a large deviation from the actual data. Nevertheless, these plots show that ARIMA in this form is not the solution to predict quiescent periods.

In Figure 17 we see a change in phase after the point in which we start to forecast. This phase change is not predicted by the ARIMA model. As the fit also includes a change in amplitude, we would expect to see a greater 95% confidence interval as the model cannot be sure such a change would occur once more. However, this is not the case. The confidence interval greatly overestimates the zone in which the prediction is accurate.

This is also the case in Figure 18. Here a change in the amplitude occurs after the forecast is started. This is also not predicted by the model. Here is also the case that the confidence interval is to narrow to predict the actual data.

The problem with the 95% confidence intervals can be explained by the assumptions that are

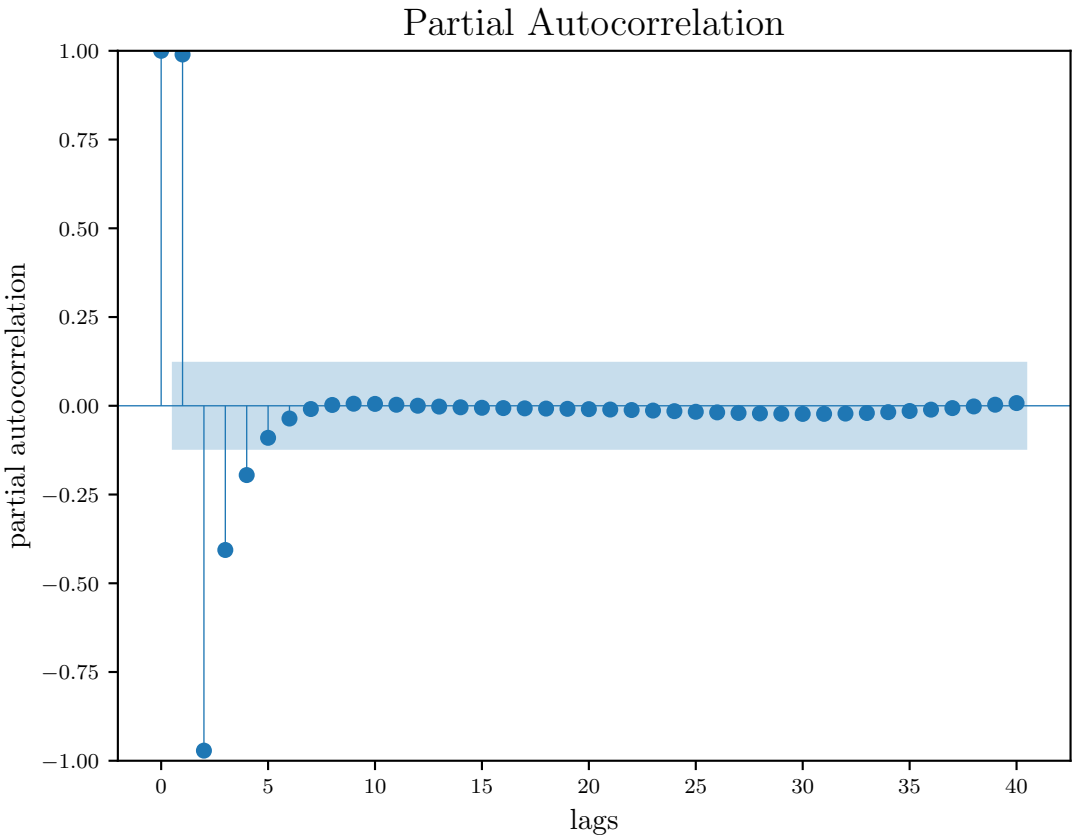

Figure 16: PACF plot for 40 lags. The dots represent the correlation between the current value and the lags. The blue shaded area represents the confidence interval.

made. These are that the residuals, i.e. the difference between the forecast and observations, are uncorrelated and normally distributed. Neither is the case. The residual of a certain data point is clearly correlated with the other residuals. The forecast does not follow the observations, meaning we have residuals are the difference of two sines, which is correlated. Furthermore, the residuals are also not normally distributed. For example, the phase is completely off in Figure 17.

In conclusion, the features of the data we want to capture in a forecast, are not captured. With this we mean the changes that occur in the time series that result in a change of phase of amplitude. This means that the ARIMA model in this form is not a good model to forecast the heave variable.

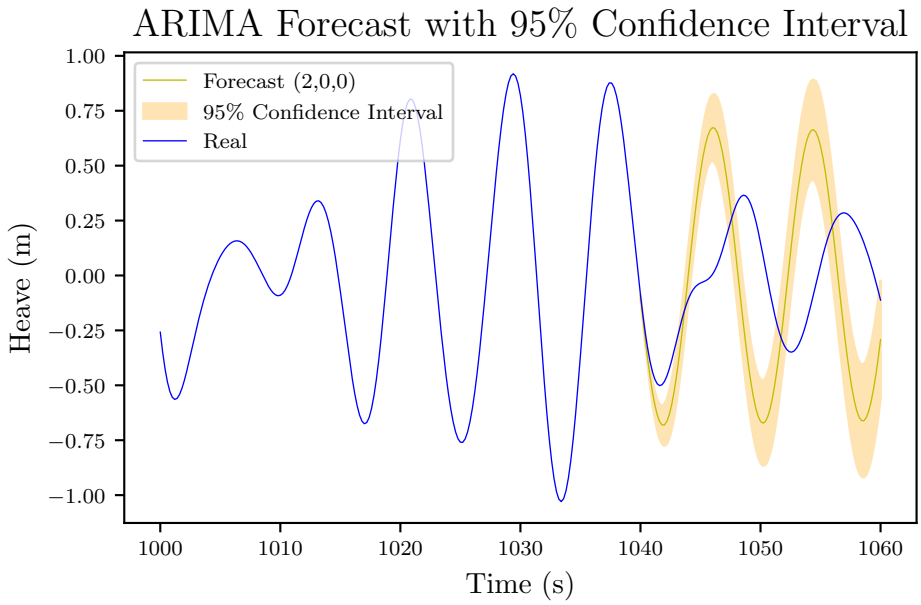

Figure 17: Forecast of the heave using an ARIMA(2,0,0) model with 95% confidence interval starting at $t = 1000$.

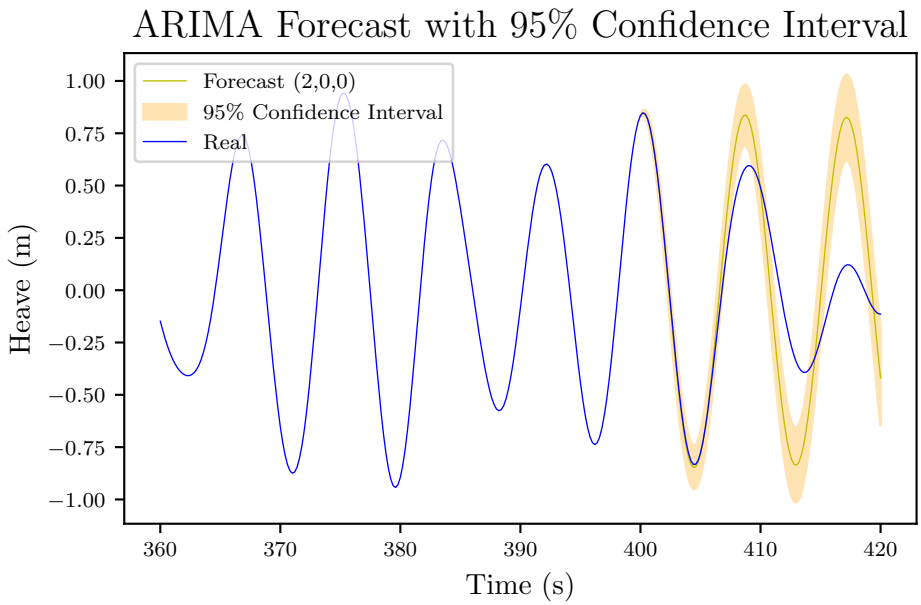

Figure 18: Forecast of the heave using an ARIMA(2,0,0) model with 95% confidence interval starting at $t = 360$.

8 Markov models

The goal of this section is to predict the height of an extremum of the time series of wave heights, given the heights of the preceding k extrema. This will be done by assuming that the sequence of extrema forms a Markov chain. That is, we assume that the height of an extremum depends solely on the height of the previous k extrema, where k is the depth of the Markov chain. This is clearly a large oversimplification. However, it may still provide a sufficiently accurate prediction to estimate when the next QP will begin.

8.1 Our approach

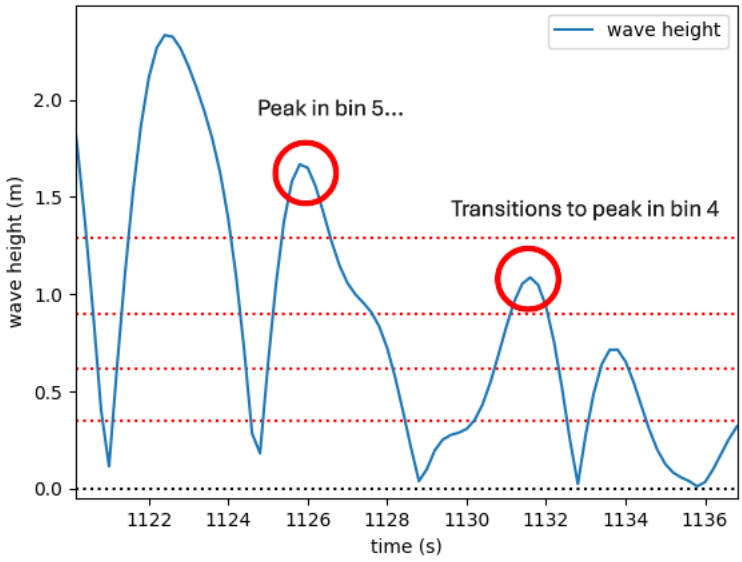

Figure 19: Illustration of a transition from $s = 5$ to $s = 4$, in the time series of the absolute value of the wave heights. The red dotted lines define the boundaries of each bin.

We begin by constructing a model of depth $k = 1$. The first step is to make a decision on the possible states of the model. We do this using bins. Each extremum with height $h \in \mathbb{R}_{>0}$ is assigned a value $s \in \{1, \ldots, 5\}$. If h falls within the lowest 20% of all extremum heights in the time series, we assign $s = 1$. If h falls somewhere within the lowest 20 to 40% of all extremum heights, we assign $s = 2$, and so on. We could also have chosen a different amount of bins, but it turns out that 5 bins gives the highest accuracy for the amount of data provided. Applying this to the dataset, we can construct a 5×5 matrix A, where the value in A_{ij} indicates how frequently an extremum from bin i transitions into an extremum in bin j. For the given dataset

we obtain the matrix

$$A = \begin{pmatrix} 2348 & 1239 & 305 & 52 & 21 \\ 1241 & 1496 & 971 & 209 & 47 \\ 309 & 961 & 1626 & 965 & 104 \\ 49 & 232 & 949 & 1971 & 764 \\ 18 & 37 & 114 & 768 & 3031 \end{pmatrix}.$$

To obtain the transition matrix P, we normalize A such that the sum over each row is equal to 1. We obtain

$$P = \begin{pmatrix} .59 & .31 & .08 & .01 & .01 \\ .31 & .38 & .25 & .05 & .01 \\ .08 & .24 & .41 & .24 & .03 \\ .12 & .05 & .24 & .50 & .19 \\ .00 & .00 & .03 & .25 & .76 \end{pmatrix}.$$

One interesting phenomenon that P shows is that the highest waves are likely to be followed by another high wave. However, it is clear from P that a depth of $k = 1$ is not sufficient for providing a useful prediction. This is not surprising, since the height of a wave is unlikely to depend solely on the height of the preceding wave. A way to remedy this is to instead look at the preceding k waves.

8.1.1 Methods without overlap

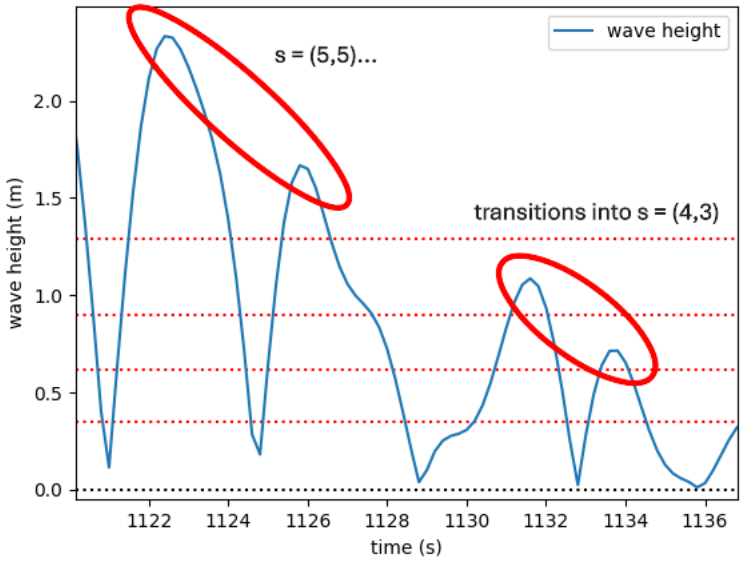

Figure 20: A transition from $s = (5, 5)$ to $s = (4, 3)$.

In the simple model, the Markov chain had five states: one for every bin that the previous extremum could fall into. To generalize this to a sequence of k extrema, we define a state $s := (s_1, \ldots, s_k)$ by a k-tuple, where the value of s_i is the value of the bin that the i-th extremum

falls in. The transition matrix will then track how often a state $(s_1, \ldots, s_k)$ transitions into the state $(s_{k+1}, \ldots, s_{2k})$. Since there are five possible values for each s_i, there are 5^k possible states for our Markov chain. This implies that the transition matrix will be a $5^k \times 5^k$ matrix. Even for $k = 2$, this already implies a 25×25 matrix.

The dataset only contains 19828 extrema. Hence, with 625 entries in the matrix, we find that in our case there are never more than 99 sequences that fall into the same entry. Moreover, when we normalize the matrix as in the previous model, we find that the highest probability of a state transitioning into another given state is less than 20%. This is not very surprising: since each sequence can transition into 25 different states, the probability of a transition into one of these is usually very low. A possible solution to this problem is to use overlap between the states.

8.1.2 Methods with overlap

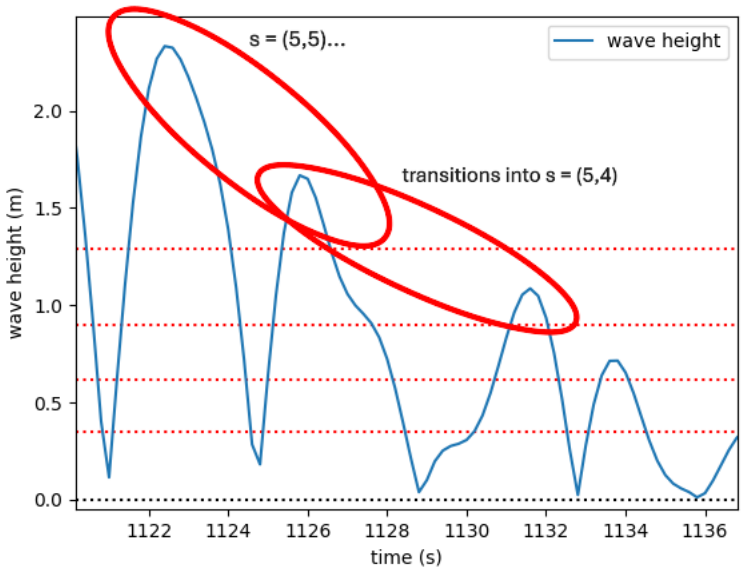

Figure 21: A transition from $s = (5, 5)$ to $s = (5, 4)$, where by definition the second entry of the first tuple is equal to the first entry of the second tuple.

Instead of trying to predict the next k extrema given the preceding k extrema, we could try to only predict the next extremum using the previous k extrema. Again, we define a state $s = (s_1, \ldots, s_k)$ as a k-tuple. However, in this model the transition matrix will track how often a state $(s_1, \ldots, s_k)$ transitions into a state $(s_2, \ldots, s_{k+1})$. Note that for $i \in \{2, \ldots, k\}$, the value s_i appears in both states. Like for methods without overlap, this will also return a $5^k \times 5^k$ transition matrix. However, all entries where the values of s_2 to s_k would be different in the next state are 0. This means that our matrix will in practice only be a $5^k \times 5$ matrix. This makes it practical to use a depth of $k = 2$. However, for depth $k = 3$ and up, the scaling of the size of the matrix is still a problem. Leaving out the zeros, the first row of the normalized transition

matrix looks like this:

$$P_{1j} = \begin{pmatrix} .61 & .34 & .05 & .00 & .00 \end{pmatrix}.$$

How well this model and the previous models provide useful predictions will be tested in the next section.

8.2 Results

8.2.1 Accuracy for predicting future extrema

The most conventional way to test the accuracy of a Markov model is to square the transition matrix, and compare this to the matrix obtained from the data when moving two states. This describes the accuracy of the model for predicting future extrema. Let P_1 be the transition matrix obtained from the data. That is, it is the transition matrix obtained by tracking the states that a given state transition to in a single step. We then define P_2 as the transition matrix obtained by tracking the states that a given state transitions to in 2 steps. If P_1 was a perfect predictor of reality, we would find that $P_2 = P_1^2$. However, we should not expect this to be the case, and we will quantify the difference between these two matrices, taking into account their size and the importance of each element (since elements of the matrix with a very low probability of occurring should matter less for the accuracy).

One way to achieve this proceeds like this: We start with the matrix $P_1^2 - P_2$. We then square each entry of the matrix, and scale each entry by the entry of the unnormalised transition matrix of P_2. We then find the norm of the matrix, defined for an $n \times n$ matrix A as

$$||A|| = \left| \sum_{0 \leq i,j \leq n} a_{ij}^2 \right|^{\frac{1}{2}}.$$

Denote the unnormalized transition matrix of P_2 as P_2^*. The process above can be summarized as the equation

$$a = P_2^* \times (P_1^2 - P_2) \times (P_1^2 - P_2),$$

where $\times$ denotes the element-wise product of matrices. For depth $k = 1$ we find $a \approx 116.86$, and for depth $k = 2$, using overlap, we find $a \approx 112.81$. For depth $k = 3$, using overlap, we find that $a \approx 625.48$. We thus find that for the amount of data used here, a depth of $k = 2$ is optimal. For a depth of $k \geq 3$, there is too much variance in the data to provide an accurate prediction.

8.2.2 Applicability to predicting QPs

Markov models are clearly not useful for long-term prediction of QPs. For any depth, the inaccuracy after more than a few extrema is too high to predict the length of an upcoming QP.

Markov models perform better at short-term prediction. If we choose the right bins, we could predict relatively accurately if the next few peaks are low enough to establish a QP. However, it is hard to obtain more optimal results for this, because of the scaling that happens when increasing depth. For a higher depth we need significantly more data to keep the prediction accurate. However, collecting this much data costs a lot of time. Moreover, it is very susceptible to long-term changes in the overall sea state. In practice, data taken from hours ago might not correlate at all with data collected later. This problem with increasing depth forces Markov models to only be a mediocre predictor for QPs on the short term.

9 Future work

In this section, we discuss some recommendations for future work on the problems described in this paper. First, we discuss possible future work on finding the distribution of QPs. Second, we list some possibilities to tackle short-term predictions of wave ship motion. The options we found for these short-term predictions are all AI-based solutions, which have proven to be good predictors of general time series in the literature [3, 4].

9.1 The distribution of quiescent periods

In this paper, we established that a suitable Poisson process models the time between QPs and the duration of QPs adequately in the simulated data. In the future, we would recommend studying how well these Poisson processes model the QPs for real ship motion, and how one best predicts the parameters for these distributions if one takes into account the changing sea state over time.

Additionally, the low certainty is still a point of contention. One valid approach would be to further increase the amount of data to try and establish convergence to the Poisson process. However, we think there might be some merit in approaching this problem using Bayesian statistics. Such an approach would use the beliefs we already have about the distribution of quiescent periods, and then update those beliefs based on the data. Taking this approach would certainly be a bit more technical, but could result in a more robust model.

9.2 AI models for QP classification

In this paper a LSTM-model was proposed to detect if a variable surpasses a certain threshold. This could be used to implicitly detect QPs. We concluded that the performance of the model decayed as the prediction window increased. We would suggest predicting less far in the future, to create a warning system for the HLO to call of the landing if the warning systems detect a big wave in the coming 10 seconds.

The LSTM-model has as input information from only one variable. Because all degrees of freedom are dependent on each other, we would recommend looking at techniques from Multivariate Time Series Analysis to use all the degrees of freedom as input. We found a paper by Karim et al. [8] that combines LSTM-networks with Convolutional Networks to form so called Long Short-Term Memory Fully Convolutional Network (LSTM-FCN). The LSTM-FCN-models preformed better on most datasets of the EUA archive compared to state-of-the-art-models form 2019.

Another suggestion would be to look at so called transformer models, which are AI-models based on the attention mechanism which was developed in the landmark paper "Attention is all you need" by Vaswani et al. [15]. Transformer models have found multiple applications, most notably in Natural Language Processing. The most famous class of Transformer models are Large Language Models, like ChatGPT and BERT. Another application of transformer models lay in Multivariate Time Series Analysis. We found a recent paper by Le et al. [11]. In this paper Shapelet Transformers, which comprises class-specific and generic transformer modules, are proposed to address the problem of imbalanced classes and small differences in minor class-specific details. The Shapelet-Transformer-model performs better on most datasets of the EUA archive compared to state-of-the-art-models.

9.3 Long Short-Term Memory networks for wave forecasting

We discovered a recent paper by Chen et al. [4] that seemed to be very useful for our assignment of predicting QPs on a short term. The study presents real-time prediction techniques for short-term prediction of so-called resting periods in ship motion. These resting periods are equivalent to QPs. The technique uses LSTM networks. The authors of the paper start by mentioning that they will be predicting the next 8 seconds of the time series. This is significantly lower than the 30 seconds that we have been trying to predict. The paper uses LSTM's differently in another way than we have tried, as well. Namely, we have tried to use direct classification techniques to be able to say something about QPs. However, in the paper, the resting periods are predicted by first predicting what the waves will look like in the future on a short-term. Then, use this information to predict when the resting periods/QPs will be.

In [4], the authors use this most recent version of the LSTM network, which uses an input-to-output ratio of 10 to 1. That is, they need the past 80 seconds of ship data to be able to predict the coming 8 seconds with reasonable accuracy. Here, a prediction system has reasonable accuracy when the prediction is reliable and accurate enough to be used for an application like predicting when helicopters can land on the ship. They use the past 80 values of a certain variable as input, which they feed to the LSTM network to predict the following 8 values of that variable.

9.3.1 Possible applications for future work on this problem

Since the research in [4] focused specifically on predicting ship movement time series, and in particular, on predicting resting periods using LSTM networks, we concluded that LSTM's would be a very interesting choice for any future researchers that are working on this problem. It would be interesting to see if the exact methods used in [4] could also be applied to the data from MARIN.

9.4 Neural Hierarchical Interpolation for Time Series Forecasting

Another interesting paper was by Challu et al. [3]. It seems not to be as directly relevant to our problem compared to the previous paper, but it also has interesting techniques whose applications look promising. The researchers introduce a model called N-HiTS. N-HiTS is a deep learning model designed for accurate and efficient multi-step forecasting. This means that the model will predict multiple steps into the future at once, compared to a step at a time. It utilizes novel techniques, and is mainly focused on long-horizon forecasting. Long-horizon forecasting is generally defined as predictions that exceed more than two seasonal periods. To clarify, if we had hourly data, our predictions would be in the order of days, and with daily data, our predictions would be in the order of weeks. This is not the case for us, but the researchers find good results for time series forecasting, which might make this a good option for future work.

9.4.1 Preliminary testing

We did some preliminary testing on applying N-HiTS to our problem using the associated python package `neuralforecast` [12]. We were not able to completely finish it and test it on predicting QPs. However, we did make future predictions of the heave series, based on recent wave motion. We used the Python package `optuna` [1] to tune the hyperparameters of the N-HiTS model. This means that we tested the performance of the model for various values of the input size, hidden

size, learning rate, dropout probability, number of blocks and downsampling frequency. These parameters all affect the model in different ways, and need to be selected carefully in order to ensure the best performance. We only tried parameters for about 15 minutes, in a narrow range. This suggested the following hyperparameters; input size: 318; learning rate: 0.001; max steps: 141; dropout probability: 0.100; n blocks: [3, 3, 3]; frequency downsample: [5, 3, 1].

Using these hyperparameters typically yields results like Figure 22. We find that the prediction is often relatively good for the first 15 seconds (75 timesteps), but that performance deteriorates significantly after that. We only had very limited time to try this technique, and since these preliminary results are promising, we would recommend further pursuing this avenue. To obtain successful results that are applicable to practical situations, we expect it will be necessary to work with much larger and more varied datasets.

Note that due to random influences in the model, these exact results are likely not reproducible with the script on GitHub. This problem could be mitigated by fixing a seed. This means that you specify the input for the random number generator, so that we can always reproduce the same result. We did not do so due to time constraints.

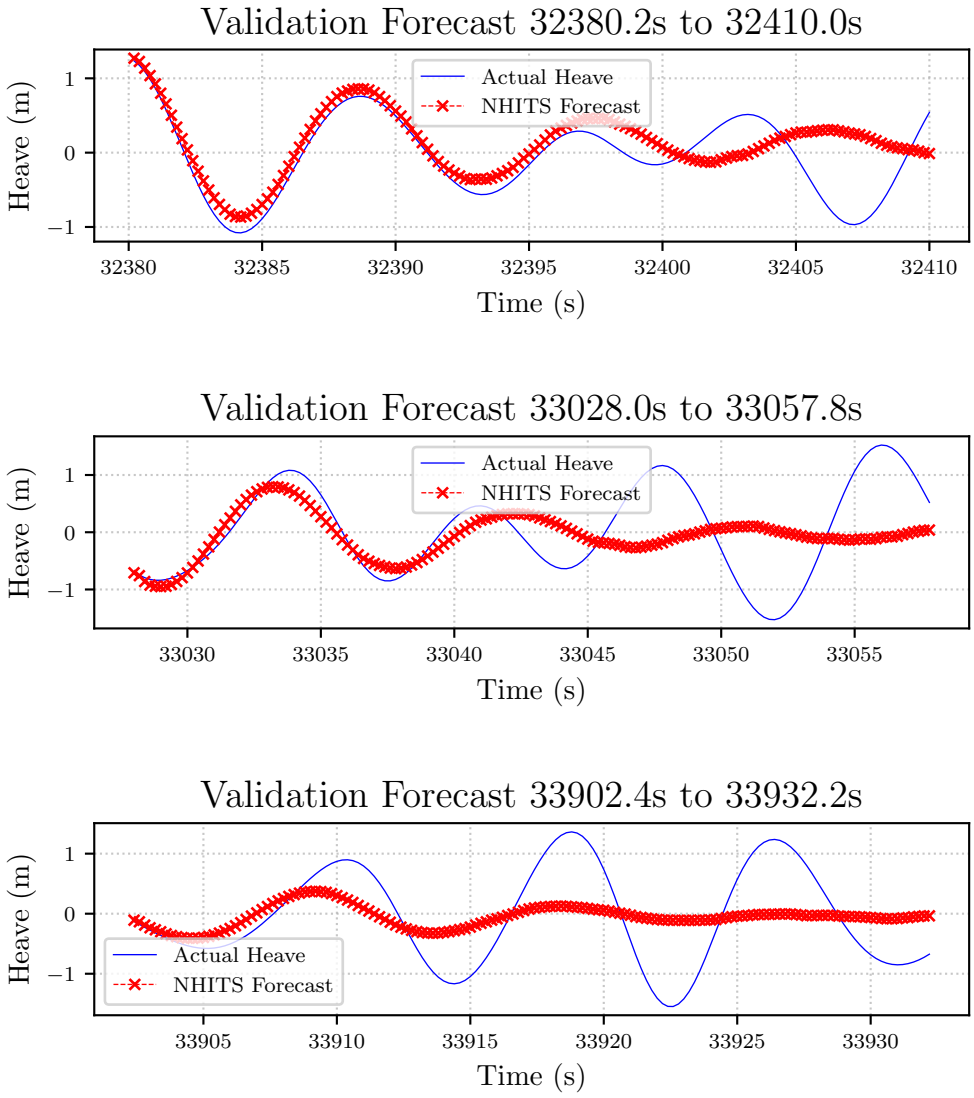

Figure 22: An arbitrary selection of N-HiTS forecasts

10 Conclusion

Our findings can mainly be split into two sections: the results of the investigations into the possible distribution of QPs, and the results concerning the short-term prediction of QPs using real-time ship data.

10.1 Distribution of quiescent periods

A previous report [5] showed that the distribution of QPs could likely be modelled reasonably well by with specific waiting times. In the first part of this report we investigated this claim. We assumed times between QPs and the QP length minus 30 were both exponentially distributed. Statistically, we established we cannot reject the hypothesis that our data follows this distribution. Consequently, this model for the QP distributions is suitable since it is simple and quite accurate.

10.2 Short-term prediction of QPs

Our initial task from MARIN concerning the short-term prediction of QPs was to find out if the statistical method of logistic regression can be used for predicting QPs. Our results were negative. We attempted to make this work by training a logistic regression model on n recent heave values. The output of the model was, given a time point, if the coming 30 seconds fulfilled the QP requirements. This is equivalent to asking ourselves the question, given that we are at a certain time point, and we have these recent heave values, can a helicopter start a landing procedure at this moment? We tried different values for n, we tried to include other variables, and we tried looking at extrema only. For all the options we tried, we only got models that would predict all time points as "not safe", as we would punish the model more for predicting "safe" when it was actually not safe. We think that the logistic regression failed to perform well in this task, because of the assumptions the underlying model makes. The most important one is the assumption that the input variables are independent from each other. In our case, since we input heave values from the same time sequence, this assumption is definitely broken. Another assumption is that the observations, i.e., our data points, are all independent, which is also not the case.

We took the liberty of trying out other prediction and classification models for predicting QPs. We used the well-established model for time series forecasting ARIMA. This model is an autoregressive integrated moving average model that uses previous values of a time series to forecast the upcoming values in the time series. We used an ARIMA$(2, 0, 0)$ model to predict the heave. This model did forecast the wave-like nature of the model, but did not forecast the changes in amplitude or phase. These features in the time series give rise to the quiescent periods. Not capturing these changes means that the ARIMA model in this form is not a useful model to forecast the time series for our purpose.

We also tried to predict QPs using the classification model random forest. An advantage of this model is that it does not implicitly assume that the data follows some distribution. Because of this, the random forest model is very versatile and often performs well on generic classification tasks. However, on our dataset, we had no such luck. The model, even with many different parameters, failed to predict QPs successfully. There are a couple of possible reasons for this, the most probable is that random forests have no specific mechanism for retaining temporal information and understanding temporal dependencies. The model inherently gives equal weight

to all attributes, which is not realistic. Also, the random forest model is not designed to capture continuous dynamic relations.

We also attempted to make a Long Short-Term Memory model to predict if the ship motion variables heave, heave rate, pitch and roll surpass a threshold in some time in the future. Long Short-Term Memory is a specific type of Recurrent Neural Networks that can capture temporal dependencies in data. We looked at two different inputs for our models: regular time series data of the variable and the last extrema of the variable. In general the model with extrema as input preformed significantly better for all variables and specifically for pitch and roll, as the wave model had a particular difficulty predicting those. Next, we examined the effect of the number of layers and hidden size on the performance. For the wave models performance got a bit better as the models dimension grew, but the extrema model performed quite well for smaller models. We also looked at the influence of the length of prediction and lookback windows. Longer lookback windows result can result in better performance but this depends on the type of input and variable. The opposite is true for longer prediction windows. While the performance of the wave model decays rapidly as the prediction window increases, the performance of the extrema model decreases more slowly.

Another model we employed was the Markov model. We applied this by classifying wave extrema into bins by their height, and assuming that the sequence of bins behaves like a Markov chain. We found that this was a relatively accurate predictor on the short term (on the order of a few waves). However, the scaling of the transition matrix size with increasing depth made it impossible to look back further than 2 extrema. This severely limited the capability of the model to make any longer term predictions.

Finally, we found a lot of online evidence and papers supporting the claim that there are AI models, in particular artificial neural networks. In the future work section, we concluded from a literature study that these models will probably excel the most at capturing the complex continuous dynamics of the wave functions. Thus, we suggest starting any future work on short-term prediction of QPs in the direction of AI models, in particular LSTM-type models.

11 Discussion

There are few points that remain unclear about the results in this report. The following three points apply to all results that showed promise.

- What is the impact of the simulated data on the results?
- Can the models be scaled to work on more than a single sea state.
- What impact would more data have on the performance of the models?

Throughout this report, we have been using simulated data for our results. This is adequate for this exploratory phase of research, but it is still unclear how our models perform on real data. Additionally, it is unclear how one would adjust our models to deal with a continuously changing sea state, since our models are currently trained one specific sea state. Finally, for all the models in this report that seem to work, we have not been able to establish whether or not the certainty of predictions continues to improve if the models are trained on larger datasets or better choices of parameters.

When it comes to the distribution of quiescent periods, one might object to our conclusion that we can model the distribution well using a Poisson process, due to the low statistical certainty. Due to this, we would be cautious when applying these results in cases where high precision is imperative, however as an indication for the number of quiescent periods, we think the models works adequately.

At short-term prediction of QPs, we were not very successful. This raises the question of whether a simpler version of the prediction problem might be more feasible. One could simply try to predict less far into the future, as most of the works listed in the future work section do. One might also consider the opposite problem. Namely, not predicting QPs, but predicting upcoming non-safe periods. The model would then function as a warning device, rather than a clearance device, which we have been trying to develop. This would be a fundamentally different way of looking at the problem, but might be necessary to create a reliable and accurate device.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] George E. P. Box and Norman R. Draper. *Empirical Model-Building and Response Surfaces*. Wiley Series in Probability and Statistics. Wiley, New York, 1987.

[3] Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. N-hits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI)*. PMLR, 2023.

[4] Zhenyu Chen, Xiaoming Liu, Xiaolei Ji, and Haoran Gui. Real-time prediction of multi-degree-of-freedom ship motion and resting periods using lstm networks. *Journal of Marine Science and Engineering*, 12(9):1591, 2024.

[5] Bisewski et al. Quiescent periods in helicopter landing on ships. 2017.

[6] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[8] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, August 2019.

[9] J. Koning and K.D.S. Zeilstra. Current practice offshore helicopter operations. 2012.

[10] Sean M. Law. STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. *The Journal of Open Source Software*, 4(39):1504, 2019.

[11] Xuan-May Le, Ling Luo, Uwe Aickelin, and Minh-Tuan Tran. Shapeformer: Shapelet transformer for multivariate time series classification, 2024.

[12] Kin G. Olivares, Cristian Challú, Federico Garza, Max Mergenthaler Canseco, and Artur Dubrawski. NeuralForecast: User friendly state-of-the-art neural forecasting models. PyCon Salt Lake City, Utah, US 2022, 2022.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[14] Robert H. Stewart. *Introduction to Physical Oceanography*. Robert H. Stewart, 1st edition, 2008. Copyright by the author, available online.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

A Appendices

A.1 Correlation of variables for logistic regression

In Table 7 (a), (b) and (c), the correlations of the heave amplitude, the correlation of velocity of the heave amplitude and the correlation of the extrema of the heave amplitude are given respectively. In each table, the correlation of the variable with its previous 30 are given. For example, in Table 7 (b) the correlation of the heave velocity with the 30 heave velocities before is shown. The value heave_v1 is one observation before the current heave velocity, heave_v30 is the heave velocity of 30 observations before the current one. In all of these tables, it is seen that most variables are highly correlated.

Table 7: Correlations of heave, heave velocity and heave extrema.

(a) heave amplitude

Variable	**z_wf**
heave30	-0.2877
heave29	-0.3965
heave28	-0.4998
heave27	-0.5954
heave26	-0.6808
heave25	-0.7540
heave24	-0.8130
heave23	-0.8560
heave22	-0.8818
heave21	-0.8893
heave20	-0.8779
heave19	-0.8474
heave18	-0.7980
heave17	-0.7304
heave16	-0.6458
heave15	-0.5457
heave14	-0.4321
heave13	-0.3074
heave12	-0.1741
heave11	-0.0352
heave10	0.1062
heave9	0.2469
heave8	0.3836
heave7	0.5132
heave6	0.6327
heave5	0.7391
heave4	0.8299
heave3	0.9029
heave2	0.9564
heave1	0.9890
z_wf	1.0000

(b) velocity of heave

Variable	**z_velocity**
heave_v30	-0.0740
heave_v29	-0.1941
heave_v28	-0.3132
heave_v27	-0.4284
heave_v26	-0.5367
heave_v25	-0.6351
heave_v24	-0.7210
heave_v23	-0.7918
heave_v22	-0.8453
heave_v21	-0.8797
heave_v20	-0.8937
heave_v19	-0.8863
heave_v18	-0.8572
heave_v17	-0.8066
heave_v16	-0.7353
heave_v15	-0.6447
heave_v14	-0.5367
heave_v13	-0.4136
heave_v12	-0.2785
heave_v11	-0.1345
heave_v10	0.0148
heave_v9	0.1655
heave_v8	0.3139
heave_v7	0.4560
heave_v6	0.5880
heave_v5	0.7065
heave_v4	0.8082
heave_v3	0.8903
heave_v2	0.9507
heave_v1	0.9876
z_velocity	1.0000

(c) extrema of heave

Variable	**z_extremum**
extremum30	0.7295
extremum29	-0.7259
extremum28	0.7237
extremum27	-0.7244
extremum26	0.7273
extremum25	-0.7303
extremum24	0.7303
extremum23	-0.7336
extremum22	0.7343
extremum21	-0.7330
extremum20	0.7296
extremum19	-0.7296
extremum18	0.7290
extremum17	-0.7312
extremum16	0.7317
extremum15	-0.7292
extremum14	0.7262
extremum13	-0.7249
extremum12	0.7266
extremum11	-0.7295
extremum10	0.7315
extremum9	-0.7321
extremum8	0.7377
extremum7	-0.7405
extremum6	0.7448
extremum5	-0.7526
extremum4	0.7632
extremum3	-0.7905
extremum2	0.8538
extremum1	-0.9460
z_extremum	1.0000

A.2 Code

All code used for this paper can be found in this GitHub repository[1].

A.3 Distribution of tasks

In Table 8 one can find the distribution of tasks during the models laboratory project. Please note that fewer items in the column content does not imply that this person has done less work. For example, Emma's work on logistic regression was an important part of the report and more comprehensive than that of Dinand, who also spent a lot of his time on random forest and the literature study. Similarly, Steven's work on classification models took a comparable amount of time to the time Casper spent on his three tasks.

Table 8: Distribution of tasks

Person	Administrative tasks	Content
Casper Algera	Communication with Ariyan en MARIN.	Analysis consistency data, distribution of QPs, NHiTS.
Dinand Blom		Logistic regression, random forest, literature future work.
Emma Ex	Agendas meetings.	Logistic regression.
Jort Jacobs		Finding QPs, distribution of QPs.
Steven Janssen	Planning meetings.	AI classification models.
Jan Schobers		ARIMA, PCA and research DeepAR.
Jorian Weststrate		Markov models.

[1] https://github.com/dinandb/Modellenpracticum