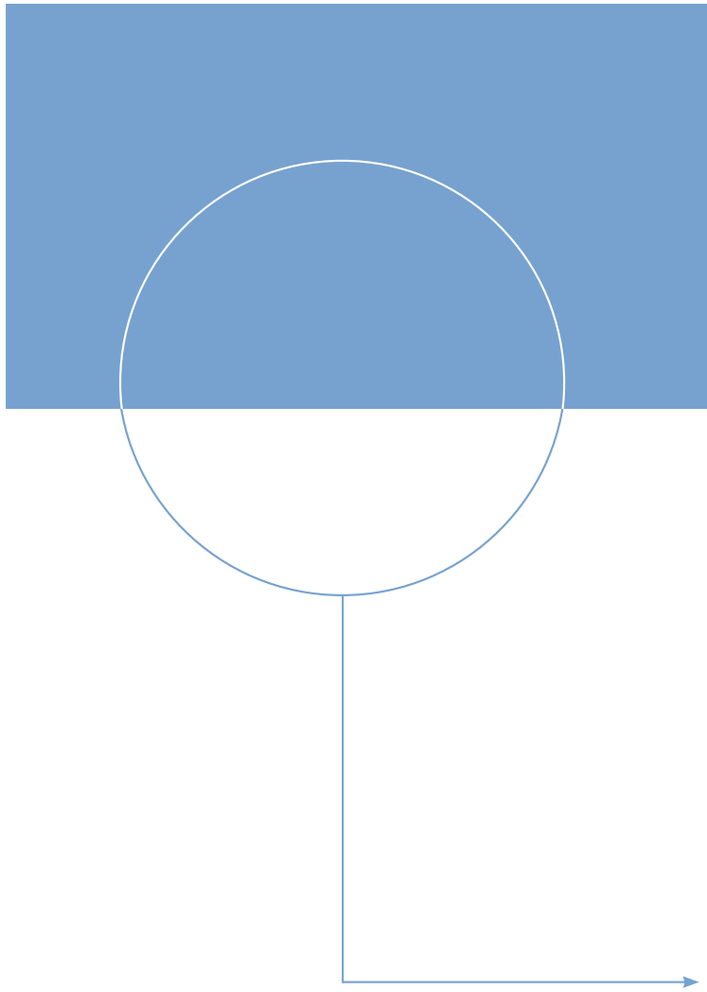


Master's thesis

2021

Haakon Ferdinand Riedesel

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Marine Technology

Haakon Ferdinand Riedesel

Experimental and numerical study of slowly varying added ship resistance in seaway

June 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Experimental and numerical study of slowly varying added ship resistance in seaway

Haakon Ferdinand Riedesel

Submission date: June 2021
Supervisor: David Kristiansen

Norwegian University of Science and Technology
Department of Marine Technology

Contents

List of Figures

List of Tables

Preface

This thesis marks the end of my two years experience at the Norwegian Institute of Science and Technology (NTNU). I feel deeply grateful to be able to say that I could study under dedicated lecturers together with my fellow students at NTNU. It has been an enthralling journey and I want to thank all people that helped me during these times.
Special thanks goes to Marco Nataletti, Øyvind Rabliås and Benjamin Lageman, who helped me during the thesis and answered my questions also at the worst of possible times. And I want to thank Joseph C Sajan, Trygve Kristiansen, Terje Rosten and Marco Nataletti again, who made it possible to do perform experiments at the SINTEF Ocean towing tank.
Finally I want to thank my supervisor David Kristiansen for his great support and endless patience he had with me. His feedback throughout the the thesis was key in keeping me on track when I once again lost sight of my own thoughts.

Trondheim, June 2021

Haakon Ferdinand Riedesel

1 Motivation

Today the sustainability of ships is of utmost interest, to reduce the greenhouse gas emissions and protect the earth and its inhabitants. Private people as well as corporations and organisations, like the United Nations(UN) ascribe to this goal. The last even decided on a set the Sustainable Development Goals(SDG) in 2015 [Program, 2015]. These SDGs consist of 17 individual goals to help humankind in the future and are set to be met by the year 2030. The SDGs include various topics, also climate action. But the UN is not the only global spanning organisation dedicated to fight climate change, the International Maritime Organisation(IMO) proposed their own strategy to prevent air pollution from ships as early as 1997, which has been updated regularly since, including the 2005 addendum [Committee, 2005]. These include strategies to cut CO_2 and NO_X, as well as Sulphur and Ozone discharged by ships.
Over the yeas a lot of different strategies of green shipping have been proposed to reach these goals, with a common theme of all of them being to get a better understanding of how to design green ships. This includes the ability to assess the efficiency of ships and the knowledge of effects acting on a operating vessel. Prominent examples of well established investigation of speed, added mass and ship efficiency are [J.M.J.Journée, 1976] and [M.Kim et al., 2017]. Both identify the added resistance as a key point in improving the efficiency of a vessel.

2 Introduction

The secondary non-linear wave forces have been investigated in various papers and articles for example in [M.Kim et al., 2017], [O.M.Faltinsen and A.E.Løken, 1980] or [J.N.Newman, 1974]. And still not everything about the vessel wave interaction is understood. Many models are too simplistic or too demanding in calculation, thus the results are either not viable without further treatment or too costly to perform. This thesis will investigate the non-linear second order varying drift force, as those are generally not included in state of the art vessel design calculations, see [DNV-GL, 2018].

For this thesis towing test were performed in cooperation with [J.C.Sajan, 2021] and the work on his master thesis. The facility and SO Ship model of a bulk carrier were provided by SINTEF Ocean.
The focus of this thesis lays on slowly varying drift forces f^{SV}, and the comparison of different methods used to estimate f^{SV}. For this reason the test provided a way to obtain added resistance response amplitude operators, which were used to perform simulations of slowly varying drift forces with two different methods. One of these being the method proposed by [J.N.Newman, 1974] and further a timedomain approach used by [O.M.Faltinsen and A.E.Løken, 1980], which was also used by [F.H.Hsu and K.A.Blenkarn, 1970]. In the following the method proposed by Newman will be called Newman approximation, and the timedomain approach will be called Method 2. The Method 2 was adjusted by [T.Kristiansen and Ø.Rabliås, 2021] in their yet unpublished work, this will be called the Øyvind variant.
To calculate the slowly varying drift force a Python program was developed to implement Newmans approximation as well as both variation of Method 2, to simulate a three hour fullscale timeseries
This allows for a indepth comparison of the different methods.

The method to calculate calm water resistance in this thesis was first proposed by [J.Holtrop and G.G.J.Mennen, 1982]. Prior to the experiments the calm water resistance was calculated to have a general understanding of the order of magnitude of forces to expect during the experiments. Calm water calculations were done by developing a Python program based on the work of [Birk, 2019].
It has to be noted, that most dimensional data in this thesis is in fullscale, if it is not noted otherwise. This was done to make comparison easier.

3 Theory

3.1 Calm water resistance

[J.Holtrop and G.G.J.Mennen, 1982] developed a resistance prediction method based on the regression analysis of random model tests and fullscale data from the Netherlands Ship Model Basin. Over the years the method has been expanded and improved. [Birk, 2019] sampled the most recent revisions in his book from 2019.

To estimate the calm water resistance with Holtrop and Mennen's method , a slender body ($5 \leq L/B \leq 8$) is demanded, with a block coefficient c_b between $0.55 - 0.85$.
The speed has to be relatively high, thus the Froude number has to be below 0.33. And the length-displacement ratio is assumed to be between $4 - 6$.

The idea is that the total calm water resistance R_c is a summation of resistances originating from different sources, so it has been subdivided into resistance components according to:

$$R_c = (1 + k_1)R_F + R_{APP} + R_W + R_B + R_{TR} + R_A + R_{AA} \tag{1}$$

Here:

- R_F: Frictional resistance according to ITTC 1957
- $(1 + k_1)$: Form factor to correlate R_F to hull form
- R_{APP}: Resistance from appendages
- R_W: Wave making and wave breaking resistance
- R_B: Pressure resistance of bulbous bow
- R_{TR}: Pressure resistance of immersed transom stern
- R_A: Model ship correlation allowance resistance
- R_{AA}: Air resistance

3.2 Sea keeping

3.2.1 Linear response amplitude operator

In the class guidelines [DNV-GL, 2018], DNV-GL suggest to use linear response amplitude operators(RAO), also called transfer function, to determine linear responses including motions, accelerations, pressures, loads and disturbed wave elevation. The concept of RAOs is also discussed in *Sea Loads on Ships and Offshore Structures* by [O.M.Faltinsen, 1998] and the classes of *TMR 4182 Marine dynamics* [C.M.Larsen et al., 2019]. According to DNV-GL the transfer function $\boldsymbol{F}$ is defined as:

$$\boldsymbol{x}(t) = \zeta_a \cdot \Re\left\{\boldsymbol{F}(\omega, \beta) \cdot e^{i\omega_e t}\right\} \tag{2}$$

Here $\boldsymbol{x}$ is the response and ζ_a is the incident single wave amplitude of a regular frequency ω. The transfer function $\boldsymbol{F}$ can also be noted as follows, using the phase θ_m of the response component.

$$\boldsymbol{F}_m(\omega_m, \beta) = |F_m| \cdot e^{i\theta_m} \tag{3}$$

3.2.2 Strip theory

The program ShipX is used to calculate the RAOs for heave, pitch and the dimensionless added resistance, more information about ShipX will be given in section 5.1. ShipX uses the strip theory to calculate forces and motions.

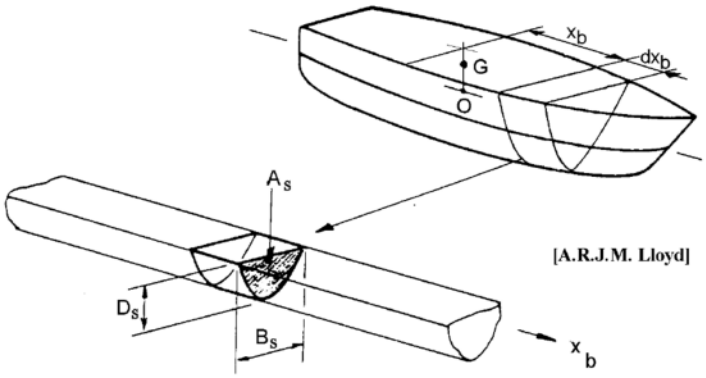


Figure 1: Strip theory: body illustration

The strip theory requires a few assumption:

- A slender body ($3 \leq L/B$) to consist of a finite number of transverse two dimensional slices, rigidly connected to each other [Rostock,]
- The slice's (or cross sections) form and area shall not change much along the x axis. Each cross section is treated hydrodynamically as a segment of a infinite long cylinder.
- All waves produced by the oscillating body and the diffracted wave to travel parallel to the y,z plane of the body.
- The fore and aft side of the body shall not produce waves in x direction.
- For zero forward speed all interactions between cross sections are ignored.

The definition of the following are taken from [Prof.Dr.-Ing.M.Abdel-Maksoud, 2016] and [H.Söding, 1982] and [H.Söding, 1995].
The general approach is to integrate the hydrodynamic parts of the equation of motion $(-\omega_e^2 A + i\omega_e N)\hat{u}$, due to the exciting wave force $\hat{F_E}$ for each frame, over the vessel length L.

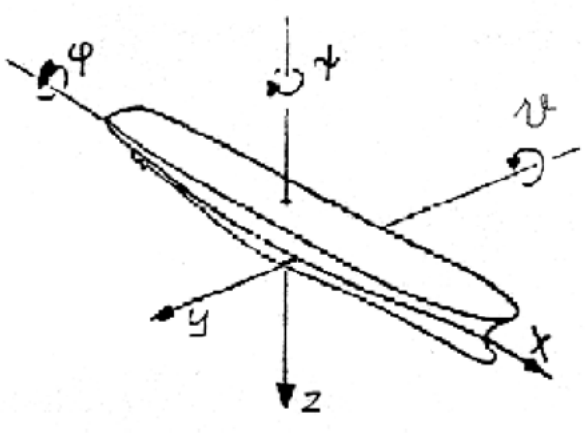


Figure 2: Definition of the coordinate system for strip theory, from [Prof.Dr.-Ing.M.Abdel-Maksoud, 2016]

The point of reference for each frame is defined as the intersection of symmetry line and waterline, while the point of origin for the whole vessel is the base of the main frame.
If the body turns around ϑ, ψ each frame will experience cross flows, for example if during a stomping motion with an angle ϑ. Then the relative movement of the water across a frame will include a $V \cdot \vartheta$ component upwards. This influences the transformation matrix $W(x)$.

$$\underbrace{i\omega_e \begin{bmatrix} \hat{\eta}_{0x} \\ \hat{\zeta}_{0x} \\ \hat{\varphi}_{0x} \end{bmatrix}}_{frame\ speed} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & t_x & 0 & x - v/i\omega_e \\ 0 & 0 & 1 & 0 & -x + v/i\omega_e & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{W(x),\, translates\, body\, motion\, into\, frame\, motion} \cdot \underbrace{i\omega_e \begin{bmatrix} \hat{\xi}_0 \\ \hat{\eta}_0 \\ \hat{\zeta}_0 \\ \hat{\varphi}_0 \\ \hat{\vartheta}_0 \\ \hat{\psi}_0 \end{bmatrix}}_{body\ speed} \tag{4}$$

t_x is defined as the draft at point x.
And for the translation of the force from frame coordinate system to the vessel coordinate system a transformation matrix $V(x)$ is defined.

$$d\hat{F} = \underbrace{\begin{bmatrix} d\hat{F}_\xi \\ d\hat{F}_\eta \\ d\hat{F}_\zeta \\ d\hat{M}_\xi \\ d\hat{M}_\eta \\ d\hat{M}_\zeta \end{bmatrix}}_{body\ forces\ \&\ moments} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & \\ 0 & t_x & 0 & 1 & 0 & \\ -t & x & 0 & -x & 0 & 1 \\ 0 & x & 0 & 0 & 0 & \end{bmatrix}}_{V(x),\, translates\, frame\, motion\, into\, body\, forces} \cdot \underbrace{\begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \hat{f}_3 \\ \hat{f}_4 \\ \hat{f}_5 \end{bmatrix}}_{frame\ force\ \&\ moments} dx \tag{5}$$

The frame force and moments are defined as:

$$\begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \hat{f}_3 \\ \hat{f}_4 \\ \hat{f}_5 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ \omega_e^2 m_{22} - i\omega_e n_{22} & 0 & \omega_e^2 m_{24} - i\omega_e n_{24} \\ 0 & \omega_e^2 m_{33} - i\omega_e n_{33} & 0 \\ \omega_e^2 m_{42} - i\omega_e n_{42} & 0 & \omega_e^2 m_{44} - i\omega_e n_{44} 0 \quad 0 \quad 0 \end{bmatrix}}_{\hat{H}_B(x), translates\ frame\ motion\ into\ frame\ forces} \begin{bmatrix} \hat{\eta}_{0x} \\ \hat{\zeta}_{0x} \\ \hat{\varphi}_{0x} \end{bmatrix} + \underbrace{\begin{bmatrix} -i\rho g k A_x cos(\nu) \\ \hat{f}_{E2} \\ \hat{f}_{E3} \\ \hat{f}_{E4} \\ -i\rho g k A_x s_x cos(\nu) \end{bmatrix} \hat{\zeta}_{Ex}}_{\hat{H}_E(x)} \tag{6}$$

$\hat{H}_E$ translates wave amplitudes into frame forces.
The subscript $_E$ stand for excitation and the subscript $_B$ stands for the German word *Bewegung*, which means motion. The forces and moments are thus composed of a part originating from the frame motion and one part as a result of the excitation force.
m_{22} is defined as the non dimensional horizontal hydromass per length and n_{22} is the non dimensional horizontal damping constant per length.
For a more precise calculation resistance curves, propeller characteristics and hydromasses in x-direction should be added to Equation 6.

The complex wave amplitude $\hat{\zeta}_{Ex}$ at a frame at position x, is dependent on the complex amplitude at the main bulkhead $\hat{\zeta}_E$

$$\hat{\zeta}_{Ex} = \hat{\zeta}_E e^{-ikx\cos(\nu)} \tag{7}$$

The combination of Equation 4 to 7 together with the integration over the body length L results in the hydrodynamic forces.
The force can be separated into one part as the result to motion and one part as result of the excitation force.

$$\omega_e^2 \hat{B} = \int_L V(x) \cdot (1 + \frac{iv}{\omega_e} \frac{\partial}{\partial x})(\hat{H}_B(x) \cdot W(x)) dx \tag{8}$$

The last part of Equation 8 can be interpreted as the impulse change of the water disc in the cross section plane, which follows the frame movement.

$$\hat{F}_E = \int_L V(x) \hat{F}_{E0} e^{-ikx\cos(\mu)} dx \cdot \hat{\zeta}_E - \int_L V(x) \left(i\omega_e - v\frac{\partial}{\partial x} \right) \left(\frac{\hat{H}_{E7}}{\omega^2} \cdot i\omega e^{-ikx\cos(\mu)} \cdot \hat{\zeta}_E \right) dx \tag{9}$$

The excitation force $\hat{F}_E$ has to be separated into a Froude-Kryloff $\hat{F}_{E0}$ and a diffraction $\hat{F}_{E7}$ component.
Equation 8 and 9 both contain a derivative to x. That is the reason why the largest errors occur when the cross section changes heavily along the x axis. This is normally the case at the stern and bow of a ship. At the stern the stream lines abruptly detach from the body, as the impulse behind the body is zero. However at the bow, the flowing water is gradually diverted, which results in a "natural" blurring, which helps to stabilise the force at the bow part and lessens the occurring error.

In combination with Equation 8 and 9 the equation of motion becomes:

$$\left[S - \omega_e^2(M + \hat{B})\right] \hat{u} = \hat{E}\hat{\zeta}_E \tag{10}$$

Thus the complex hydromass matrix can also be described as $\hat{B} = A = -\frac{i}{\omega_e}N$. The real part of $\hat{B}$ can be interpreted as forces per acceleration, while the imaginary part can be seen as the damping forces in phase with the velocity. And the matrix S is the restoring force matrix.
Rudder forces due to course deviation are included in the restoring matrix S. While rudder forces due to vessel movement can be included by assuming the deadwood to be extended to the trailing edge of the rudder.

3.3 Sea states

Sea states can for example be created by wind blowing over calm water surface which leads to small ripples on the water. Those ripples then lead to small regular waves. At the same time new ripples are created on top, and thus create new waves. The phase relation between these waves will be random, as the wind direction is as well. A random natural sea state assumes, that the significant wave height H_s and the mean wave period T_p are constant over the observation period T_H. Simultaneously the wave steepness is taken to be within the range of linear theory [Prof.Dr.-Ing.M.Abdel-Maksoud, 2016].

To describe the characteristics at a different places with the same wind, the wave elevation is assumed to be described by :

$$\zeta(t) = A_0 + \sum_{j=1}^{N} \Re\left\{A_j e^{i\omega_j t}\right\} \tag{11}$$

Here $\omega_j = j \cdot \Delta\omega$ and and A_0 is set to be 0, as it is chosen to be the reference plane. The phase angle ϵ_j will be different for each spatial point, thus it is neglected. The spectrum $S(\omega_j)$ is therefore defined as:

$$\tfrac{1}{2}A_j^2 = S(\omega_j)\Delta\omega \tag{12}$$

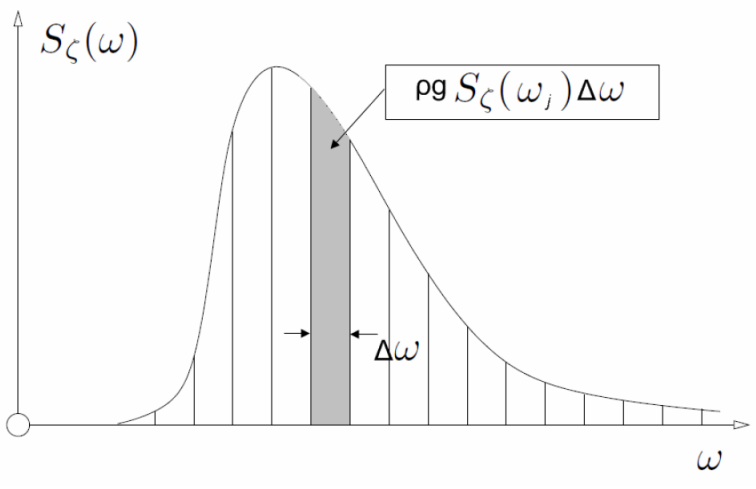


Figure 3: Spectrum, found in [Prof.Dr.-Ing.M.Abdel-Maksoud, 2016]

The area marked in grey corresponds to the wave energy per area for the frequency interval $\Delta\omega$.

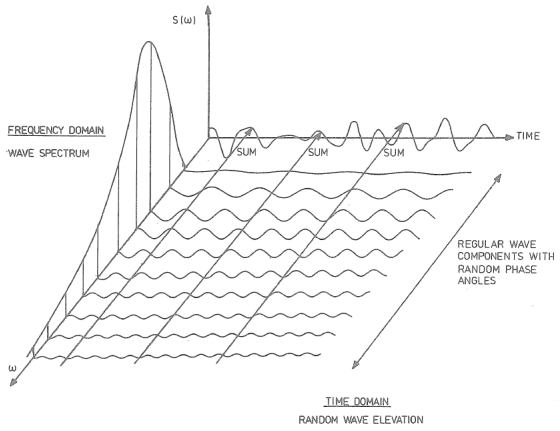


Figure 4: Correlation between frequency domain and time domain for long-crested short term sea states, found in [O.M.Faltinsen, 1998]

For this work we look at a long-crested sea state, meaning we assume all wave components to be unidirectional. In contrast to the short-crested sea state where the spectrum $S(\omega, \mu)$ is two dimensional over frequency ω and wave propagation direction μ [Prof.Dr.-Ing.M.Abdel-Maksoud, 2016].

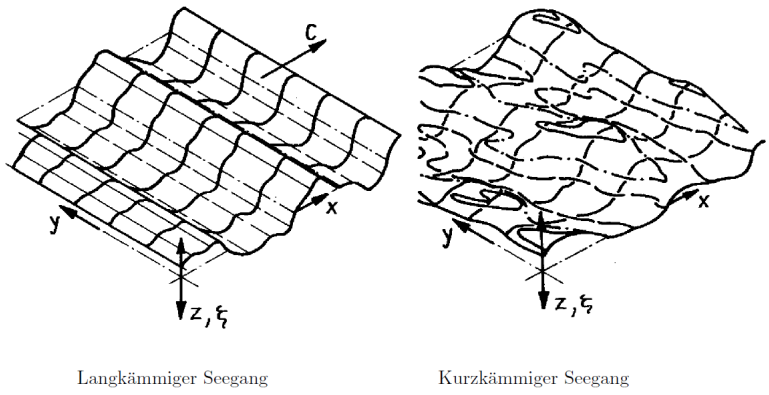


Figure 5: Comparison long-crested sea state(left) and short-crested sea state(right)

Figure 5 was found in [Prof.Dr.-Ing.M.Abdel-Maksoud, 2016].

To model a sea state a spectrum in the frequency domain is needed. The used spectrum in this thesis is the JONSWAP spectrum, which simulates sea states of the northern sea.

3.3.1 JONSWAP

The JONSWAP spectrum was proposed during the Joint North Atlantic Sea Wave Observation Project [K.Hasselmann et al., 1973].
The spectrum $S(\omega)$ has to be zero for large ω , as a high ω value correspond to short waves, which only have small amplitudes before it would break. On the other hand $S(\omega)$ has to be zero for frequencies, where the phase speed largely exceeds the wind speed responsible for the sea state.

$$S_J(\omega) = A_\gamma S_{PM}(\omega)\gamma^{exp\left(-0.5\left(\frac{\omega-\omega_p}{\sigma\omega_p}\right)^2\right)} \tag{13}$$

The JONSWAP spectrum is defined to be identical to the Pierson-Moskowitz spectrum when the non-dimensional peak shape parameter $\gamma = 1$. The DNV-GL recommends a mean value of γ equals to 3.3. In the following γ equals to 3.3.

$$S_{PM}(\omega) = \frac{5}{6} \cdot \frac{H_S^2\,\omega_p^4}{\omega^5} \cdot e^{-\frac{5}{4}\left(\frac{\omega}{\omega_p}\right)^{-4}} \tag{14}$$

The normalising factor A_γ is defined as $A_\gamma = 1 - 0.287 \cdot ln(\gamma)$. While the spectral width parameter σ changes with ω, where $\sigma = 0.07$ if $\omega \leq \omega_p$, while $\sigma = 0.09$ if $\omega > \omega_p$, see Figure 25.

To get a time series from a spectrum the random phase angle ϵ_j has to be set to a value between 0 and 2π. Then the sum over all wave components is taken for each timestep.

$$\zeta(t) = \sum_j \sqrt{2S(\omega_j)\Delta\omega} \cdot cos(\omega_j t + \epsilon_j) \tag{15}$$

3.4 Added Resistance

Added resistance can be described as the longitudinal drift force component. It is caused by interaction between incoming waves and the vessel [O.M.Faltinsen, 2009]. Assuming conservation of fluid momentum and energy in addition to potential flow, added resistance in waves comes from the ship's ability to generate unsteady waves [J.Gerritsma and W.Beukelman, 1972].

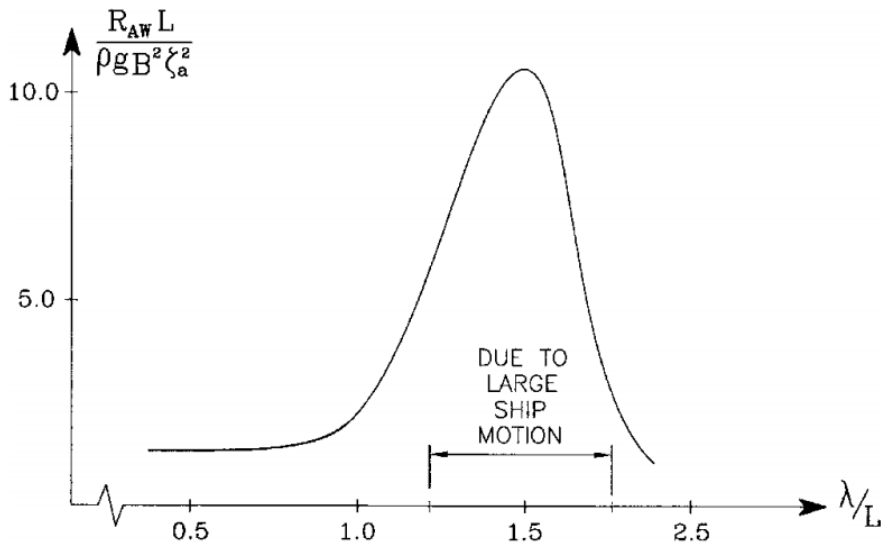


Figure 6: Typical added resistance curve in head sea waves, found in [O.M.Faltinsen, 2009]

Figure 6 showcases how the largest added resistance is due to large ship motion in waves approximately as long as the vessel. The added resistance curves can be seen as two parts, one part is where short waves lead to a limited response. Here the resistance is dominated by wave diffraction. The other part is the range where the response is large due to ship motion, here the added resistance is dominated by the wave radiation as a consequence of the large motion.

The second order non-linear wave force can be separated into two components, the mean wave force and a time varying component. A non zero mean wave force occurs, if the relative vertical motion changes around the waterline, meaning, when the vessel alters the incident wave. And time varying components arise from interaction of two or more wave components with different circular frequencies ω_j and ω_k, leading to additional Bernoulli terms including the difference and sum of ω_j and ω_k.

3.4.1 Mean wave drift forces

Maruo

There are several ways of calculating the mean wave drift force in the following a short explanation of two of the most used methods is given, these being Maruos formular [H.Maruo, 1960] and the direct pressure integration method [J.A.Pinkster and Oortmerssen, 1977] respectively.

In 1960 Maruo [H.Maruo, 1960] derived formulas to calculate mean drift forces on three dimensional structures in incoming regular waves.

$$\bar{f}_x = \frac{\rho g}{4} \int_0^{2\pi} A^2(\theta)(cos(\beta) - cos(\theta))d\theta \tag{16}$$

$$\bar{f}_y = \frac{\rho g}{4} \int_0^{2\pi} A^2(\theta)(sin(\beta) - sin(\theta))d\theta \tag{17}$$

Here β is the propagation direction of the incident wave, relative to the x-axis and $A(\theta)/\sqrt{r}$ is the wave amplitude generated by the body at a large horizontal distance, including radiated and diffracted waves.

Direct Pressure Integration

The second presented method, the direct pressure integration method (DPI) was thought up by [J.A.Pinkster and Oortmerssen, They assume small wave length, such that the body does not oscillate and the wave only affects the upstream side of the body leaving a shadow region downstream. To include the drift force use the Bernoulli equation and integrate over the exact wetted surface [O.M.Faltinsen, 1998]:

$$p = \underbrace{-\rho g z - \rho \frac{\partial \Phi_1}{\partial t}}_{I} \underbrace{-\frac{\rho}{2}\left(\left(\frac{\partial \Phi_1}{\partial x}\right)^2 + \left(\frac{\partial \Phi_1}{\partial y}\right)^2\right)}_{II} \tag{18}$$

Here Φ_1 is the velocity potential of the linear solution, while the wave amplitude at the body ζ is defined as $\zeta = 2\zeta_a sin(\omega t)$.
The results from the two terms in I lead to:

$$\overline{-\rho g \int_0^{\zeta} z dz} - \overline{\rho \frac{\partial \Phi_1}{\partial t}\bigg|_{z=0} \zeta} = \rho g \zeta_a^2 \tag{19}$$

And II leads to:

$$\overline{-\frac{\rho}{2} \int_{-\infty}^{0} \left(\left(\frac{\partial \Phi_1}{\partial x}\right)^2 + \left(\frac{\partial \Phi_1}{\partial y}\right)^2\right) dz} = -\frac{\rho g}{2}\zeta_a^2 \tag{20}$$

Integration over the non shadow part L_1 results in the drift force component in x-direction.
Here β is the wave propagation angle and θ is defined by the alignment of the body, see Figure 7.

$$\bar{f}_x = \frac{\rho g \zeta_a^2}{2} \int_{L_1} sin^2(\theta + \beta) n_x dl \tag{21}$$

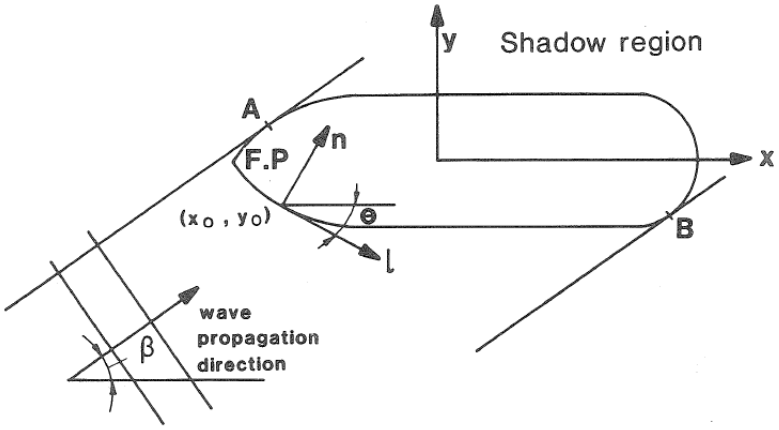


Figure 7: Definition of wave and vessel parameter in Equation 21, found in *Prediction of Resistance and Propulsion of a Ship in a Seaway* by [O.M.Faltinsen et al., 1980]

3.4.2 Slowly varying drift forces

The slowly varying drift force consists of a sum and difference frequency part. For large ships and offshore structures the difference frequency part is crucial, as the small frequencies lead to wave length in the same order as the ship or structure, which can lead to large second order forces, see Figure 6. In the following two methods are introduced on how the slowly varying drift force component can be approximated.

Newman approximation

In reality any ocean bound vessel or structure will experience second order hydrodynamic forces and moments, which in turn include a steady state and a harmonic oscillating part. While Maruo and DPI give good result in regards to the mean values, [J.N.Newman, 1974] developed a method to calculate the harmonic oscillatory part using a simplification where the transfer functions F_{jk} and F_{kj} are approximated by F_{jj} and F_{kk}. The original method was improved and expanded by [J.N.Newman, 1990] and [J.N.Newman, 1996].

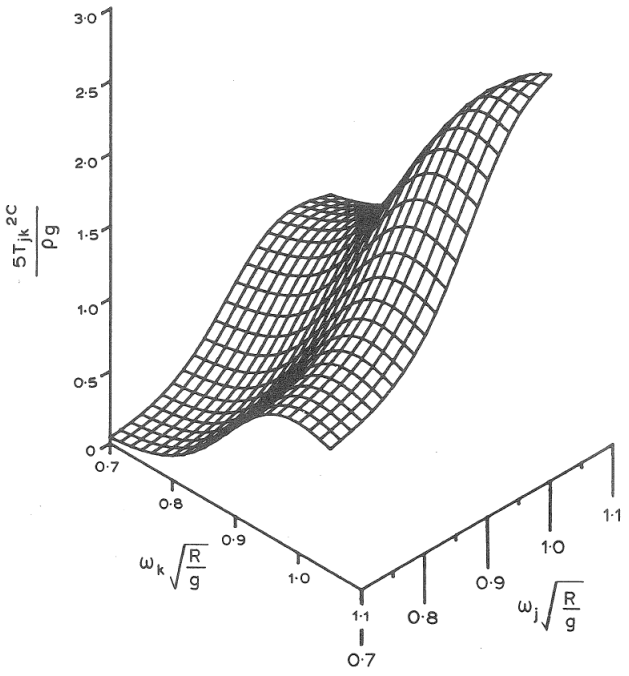


Figure 8: Second order transfer function, taken from [O.M.Faltinsen, 1998]

Starting with the assumption that any incident wave system can be described using a discrete spectrum:

$$\zeta(t) = \Re\left\{\sum_j A_j e^{i\omega_j t}\right\} \tag{22}$$

Where A_j is a complex amplitude $A_j = |\zeta_j| e^{i\omega_j \epsilon_j}$ including random phase angle ϵ_j, which is constant in time and we demand $\omega_{j+1} > \omega_j$ for all j. The first order hydrodynamic force thus resulting in:

$$f^{(1)}(t) = \Re\left\{\sum_j A_j F_j^{(1)} e^{i\omega_j t}\right\} \tag{23}$$

And for the second order hydrodynamic force:

$$f^{(2)}(t) = \Re\left\{\sum_j \sum_k A_j A_k F_{jk}^{(2+)} e^{i(\omega_j+\omega_k)t}\right\} + \Re\left\{\sum_j \sum_k A_j A_k^* F_{jk}^{(2-)} e^{i(\omega_j-\omega_k)t}\right\} \tag{24}$$

For the first and second order transfer function applies that $F_j^{(1)} \equiv F^{(1)}(\omega_j)$ and $F_{jk}^{(2)} \equiv F^{(2)}(\omega_j, \omega_j)$, while the plus and minus indicates a sum or difference frequency transfer function. A_k^* is the complex conjugate of A_k.
If now $|\omega_m - \omega_n| \ll 0.5(\omega_m + \omega_n)$, and the transfer functions F_{jk}^{2-} are demanded to be regular functions of ω_j and ω_k, then:

$$F_{jk} = F_{jj} + \mathcal{O}(\omega_j - \omega_k) \tag{25}$$

And for the second order force

$$\widetilde{f(t)} = \Re\left\{\sum_j \sum_k A_j A_k^* F_{jj}^{(2-)} e^{i(\omega_j - \omega_k)t} + \mathcal{O}(\omega_j - \omega_k)\right\} \tag{26}$$

Equation 26 still includes a double summation and thus, even if it reduces calculation effort compared to Equation 24, still requires a lot of numerical prowess. That is why [J.N.Newman, 1974] introduced additional steps:

$$\begin{aligned}\left(\Re\left\{sum_j A_j \sqrt{|F_{jk}|} e^{i\omega_j k}\right\}\right)^2 = 0.5\ \Re&\left\{\sum_j \sum_k A_j A_k \sqrt{|F_{jj} F_{kk}|} \cdot e^{i(\omega_j + \omega_k)}\right\} \\ &+ \Re\left\{\sum_j \sum_k A_j A_k^* \sqrt{|F_{jj} F_{kk}|} \cdot e^{i(\omega_j - \omega_k)}\right\}\end{aligned} \tag{27}$$

And for the transfer functions:

$$\sqrt{|F_{jj} F_{kk}|} = |F_{jj}| + \mathcal{O}(\omega_j - \omega_k) \tag{28}$$

Combining these equations results in the slowly varying part of the second order hydrodynamic force, thus Equation 26 can be written as:

$$\widetilde{f(t)} = [\widetilde{L^+(t)}]^2 - [\widetilde{L^-(t)}]^2 + \mathcal{O}(\omega_j - \omega_k) \tag{29}$$

Where:

$$L^{\pm}(t) = \Re\left\{\sum_j A_j \sqrt{\pm F_{jj}} \cdot e^{i\omega_j t}\right\} \tag{30}$$

As the difference frequency is the main point of interest for a vessel, the formular for the slowly varying drift force f^{SV} can be rewritten as:

$$f^{SV}(t) = 2\left(\sum_j A_j \sqrt{F_{jj}} \cdot \cos(\omega_j t + \epsilon_j)\right)^2 \tag{31}$$

Time domain approach

[F.H.Hsu and K.A.Blenkarn, 1970] as well as [O.M.Faltinsen and A.E.Løken, 1980], used a method to approximate the total drift force, by dividing a irregular sea in small intervals of regular waves. The interval is chosen to be the length between two zero crossings, while the length is defined as $1/2T_j$, with T_j being the wave period. The highest peak in the interval is defined as the corresponding wave amplitude ζ_j, see Figure 9.
This way the slow drift mean force for each regular wave interval can be calculated using the added resistance

transfer function on each interval.

$$\overline{f^{SD}} = \sum_j \zeta_j^2 F_{jj} \tag{32}$$

Figure 9 shows how the force timeseries is composed of different intervals of mean added resistance. From the timeseries the mean second order non-linear wave force can be calculated by taking the mean over the simulated time. Thus the slowly varying component is found by subtracting the mean component from the timeseries.

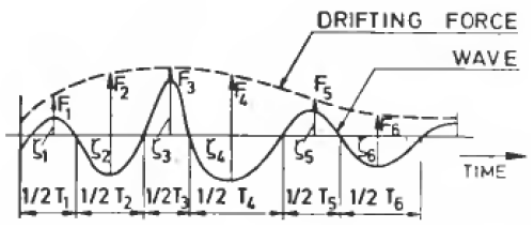


Figure 9: Irregular wave system divided in regular intervals, found in [O.M.Faltinsen and A.E.Løken, 1980]

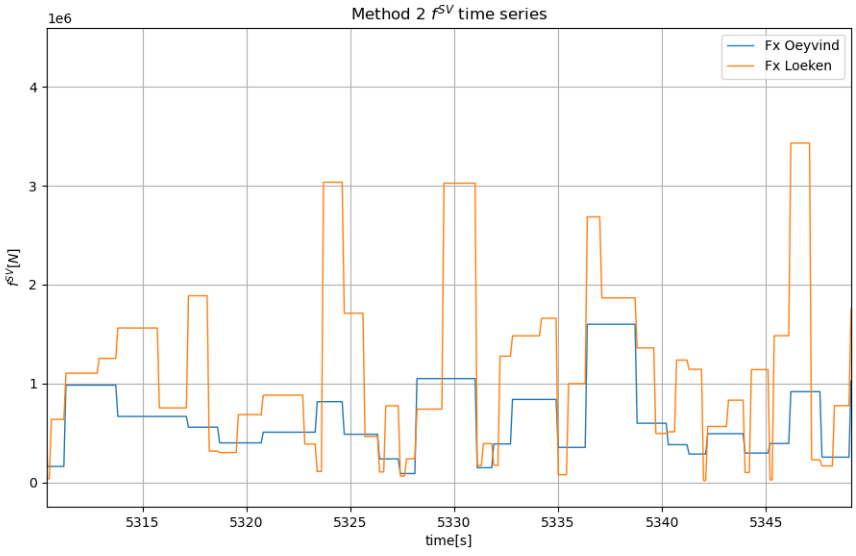


Figure 10: Method 2: f^{SD} time series, showcasing the separation of an irregular sea state in regular sea intervals

Through personal correspondence the idea of a variant of the timedomain approach was conveyed and thus included in the scope of work for this thesis. The method is called the Øyvind variant after Øyvind Rabliås, who allowed for the method to be included before the original paper has been published. The method is

based on the same assumptions as the Faltinsen/Løken variant, but instead of taking the length between two zero crossings, the length between to zero up crossings is used as T_j, while the difference between the highest maximum and the lowest minimum is defined as $2\zeta_j$.

4 Experiment

4.1 Experimental setup

The experiments are used to identify the the added resistance response amplitude operators. They were performed at the towing tank at SINTEF Ocean, using the provided SO Ship model from SINTEF and NTNU. The towing tank at the SINTEF facilities has a length of $260m$ and a width of $10m$. At one end the wave maker is located and at the opposite side a beach to damp the waves and reduce wave reflection. There are two carriages located on the rails along the tank. For this particular set-up the carriage, which is able to hold the hexapod was used. That means the other carriage was blocking part of the tank near the beach, so the actually usable tank length was reduced to approximately $80m$.

Table 1: SO Ship measurements, scalefactor $\lambda = 32$

parameter	fullscale	model scale
$L_{PP}[m]$	190	5.94
$Width[m]$	32.22	1.001
$Draft[m]$	11	0.34
$Displ.[t]$	48.93	1.49
C_b	0.728	0.728
$KG[m]$	11.191	0.35
$X_{COG}[m]$	99.89	3.11
Y_{COG}	0	0
$GM_{xx}[m]$	2.5	0.08
$R_{yy}[m]$	$47,5$	$1,48$

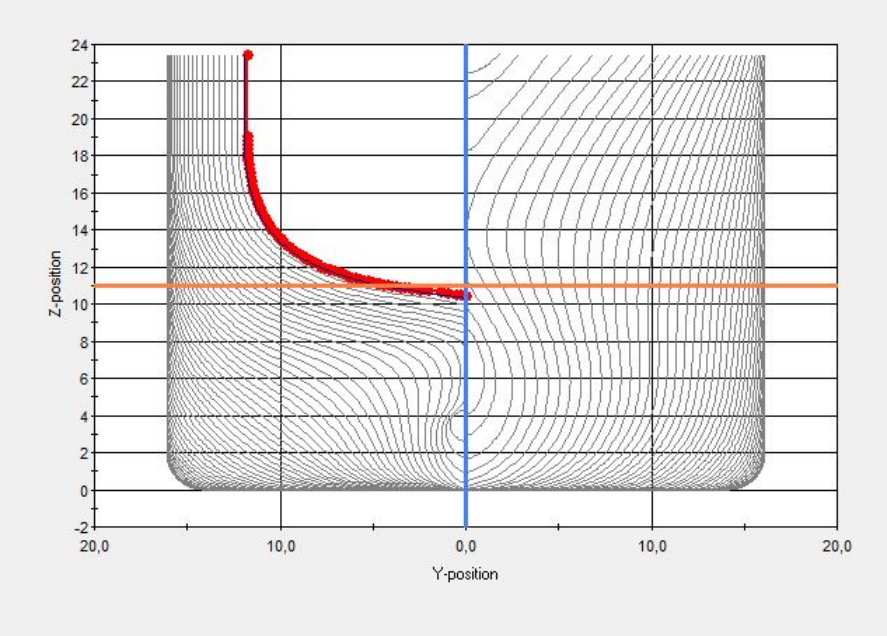


Figure 11: Section diagram of the SO Ship from ShipX

The setup included the model in scale 1:32. The model had no bilge keel, but a bulbous bow and stern. As the experiment was a continental towing test, no propeller was attached. Also no rudder was connected to the model.
The model was connected to the hexapod via a frame and three cylinder in guided bearings. The bearing allowed for vertical motion of the cylinder as well as rotation around the models y-axis, to make sure they do not limit the heave and pitch motion of the model. On the model three cylinder were placed in a bearing clamped on a bar placed along the models longitudinal axis (x-axis). The middle bar was able to move freely in rotation around the models x-axis. This way the model was fixed in surge, sway and yaw, while still being able to move in heave, pitch and roll. The force transducer was connected between the rotation enabled bar and model, while the amplifier was placed inside the model, see Figure 12.
Figure 12 also shows how the bearing connects the cylinder to the bar.

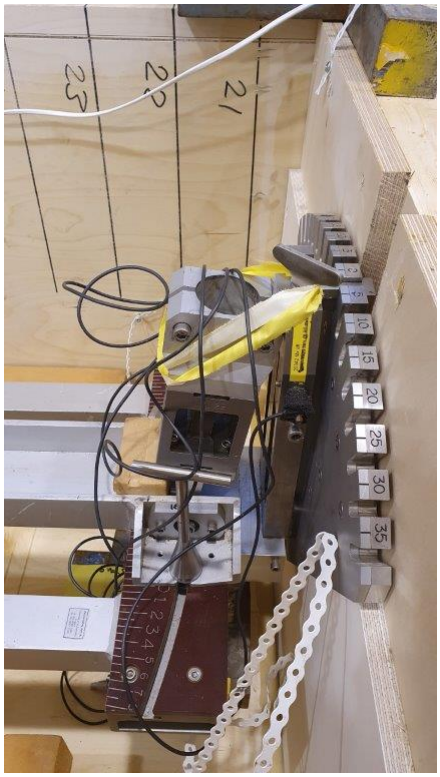


Figure 12: Connection between bar and model with force transducer

To reduce the risk of water entering the inside of the model vessel a small barrier was placed around the interior, in addition to a metal plate at the stern and a polyfoam wave reflector placed in front.
The hexapod, which allowed the setup to change the wave heading direction, was mounted to the carriage used to move the model.

The model had to be weighted and aligned to the supposed draft and trim. This was done by placing weights and adjusting the weights inside the model, see Figure 13. To make sure the trim remains the same during the experiments the weights had to be fastened to the model.

Figure 13: Weight distribution and fastening

Most of the technology used during the experiments is located on the carriage, including the computers used for measurements coming from force transducer and the OQUS system. The section which included the computers was called bridge, in analogy to the bridge of a ship.

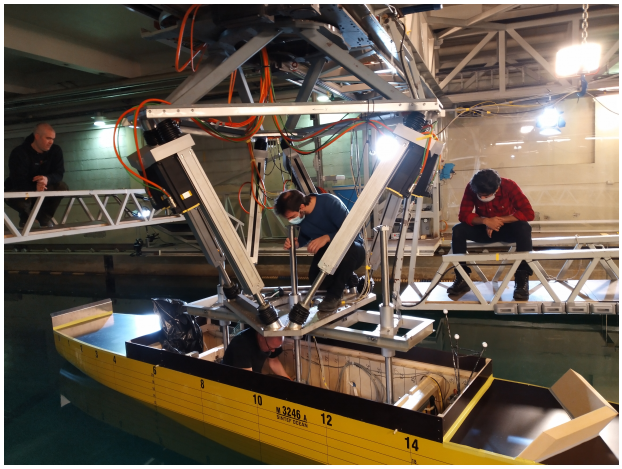
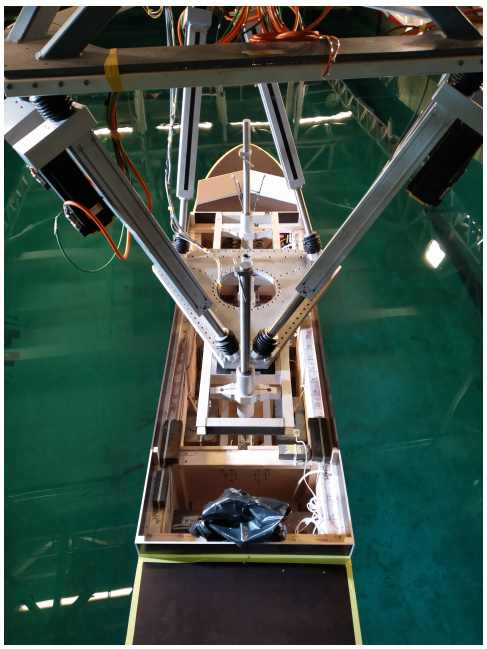
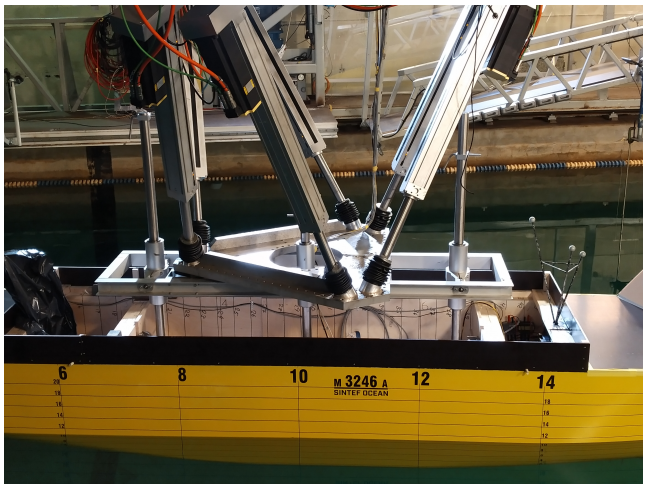


Figure 14: experimental setup: Connection between model and hexapod



(a)



(b)

Figure 15: Model set-up: view from back(a) and from starboard(b)

The force transducer uses strain gauges and a Wheatstonebridge to calculate the the applied force. The strain gauge changes conductivity depending on the applied force and by adjusting one of the resistances R_3 to get the same combined resistance in R_3 and R_x as in R_1 and R_2 the extent of deformation can be calculated, which can be translated to a force.

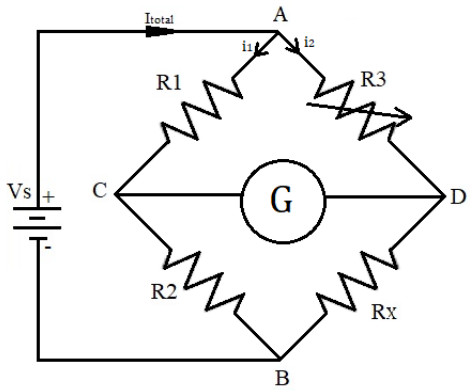
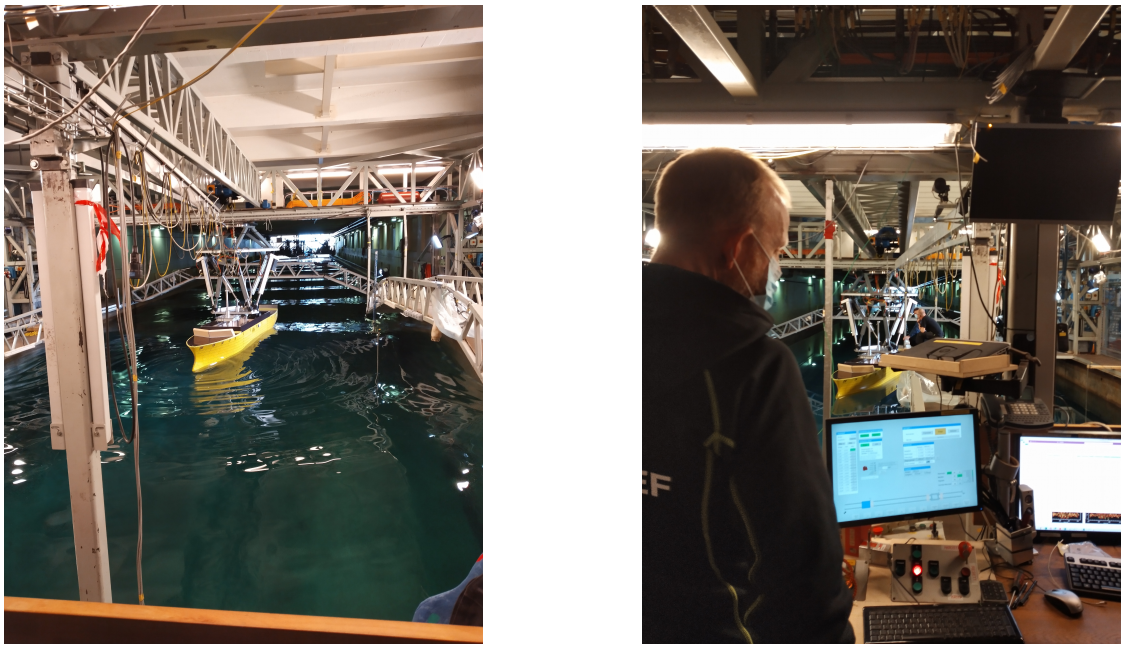


Figure 16: Circuit Wheatstonebridge

The OQUS system is an optical system that tracks three defined point on the model and translates the movements of these markers to the whole model.
Point of origin for the motion tracking was at the intersection of the base water line at the mid of the model,

where the main frame (section 10) was located. This can be seen in Figure 15. The cameras tracing the markers are located at the corners of the carriage. The model had two sets of markers, one at the bow and one at the stern, this was done because the model was also supposed to perform following wave tests and the hexapod impaired the view of the OQUS system. To avoid disturbance from the second set of markers, the unnecessary markers were covered.



(a) (b)

Figure 17: Carriage: view from bridge(a) and computer set-up(b)

To measure the incoming wave five wave probes were used, four inductive sensors and one ultrasonic. Two inductive wave sensors were placed at a five meter distance from the wave maker (WP1 and WP2), while the other two were placed on the carriage near the model (WP3 and WP4), the ultrasonic wave probe was placed in front of the carriage.
The wave probes use the conductivity of water to calculate the water level. With rising or sinking water level the conductivity between two rods changes and can be set in proportion to the water level between the rods.

4.2 Execution

The experiment was done in collaboration with [J.C.Sajan, 2021]. The test included head sea with a vessel speed of 1.36m/s (15kn fullscale) and runs at 1.06m/s (12kn fullscale), in addition to irregular sea states at 1.36m/s. This allows for the response amplitudes to be determined. The runs were defined by a set of carriage speed v_s, wave(H, T) and heading angle μ.
Joseph needed to run tests at 1.36m/s for different heading angles(0°, 9°, 180°, 189°). All test waves had a steepness of 1/40.

The waves were created by a wave maker and were dependent on the physical capabilities of the wave maker. That means the waves were created with an error margin and the difference between the waves could not be infinitely small. The bigger the difference between wave length, the bigger the distance for sampling point for the response amplitude operators. Therefore the accuracy of the response amplitude operators was limited by the error margin of the wave maker. Waves beneath a certain threshold were also not feasible, as they would dissipate quickly, possibly before reaching the model.

In table 6 to 9 all planned runs with corresponding speed v_s, heading angle μ and wave parameter H, T and λ/L_{PP} are listed. λ/L_{PP} is the ratio between wave length λ and length between perpendiculars L_{PP}. The last column in these tables show whether a test run had to be cancelled either because of too little time or because of physical limitations of the set-up.
Speed and dimensional wave parameters are given in fullscale.

In addition to the regular wave runs two irregular sea states test were performed. Those tests were different, as the input signal for the wave maker consisted of a JONSWAP sea state and to test the whole sea state more than one run was necessary. The way to perform these tests was to perform five run in one test, meaning the carriage drove from start to end position and backed up, while the wave maker continued to create waves, to then drive through the tank additional four times. This way the test recorded the waves throughout the spectrum instead of just the first few waves.
Each run results in approximately 90 seconds of viable data, thus five runs in model scale correspond to 42 minutes of fullscale irregular sea state.

The original time frame from the 25th of January to the second of February had to be pushed back to the second of February up and including the eighth of February. Because of the delay, there was no time to perform a wave validation, instead measurements of the wave at the wave maker were taken to keep track of the incident wave. Because of technical difficulties during the tests plus the delay in sat the start, an addition of two day were granted to perform tests.
The time was still not enough to perform all desired tests.

4.3 Experiment logbook

On the fist day the model vessel had to be ballasted and balanced, before it could be mounted to the hexapod. Because of the time issue the results of the heeling and swing test were done after the experimental runs.

The second day was used to place the sensors in a way to get as little interference with the model or other sensors as possible. During the experiments the placement of the wave sensors had to be adjusted, to not block the camera view as well as not giving false measurements because of interference with the tank wall. Heeling tests for roll pitch and yaw were performed to verify the measurements for force moments. At the end of the day a calm water run with zero degree heading and 1.36m/s was done to check the functionality of the setup.

The third and fourth day were necessary to perform checkups and improve the setup, as the measured data was not in unison with the expected. The OQUS system needed a new calibration, as it did not track the vessel correctly. This had to be done by someone outside of the assigned team, since no one had experience in working with the OQUS system. On the fourth day the computer running the OQUS system needed to be replaced, as it was not compatible with the set-up. This included replacing and recalibrating the OQUS set-up. The middle cylinder connecting the model and the frame had to be replaced with a shorter cylinder, as it did not slide smoothly in the bearing, but got stuck during the tests, this distorted the force and motion measure-

ments. The video hard drive had to be replaced on Monday the eighth. Additionally two of the cameras had a defect, hence the video data from Monday is greatly reduced.

Two irregular sea state runs were performed at the end of the eighth and at the beginning of the ninth day. Also on the ninth day the model setup was turned to enable the following sea tests for Josephs thesis. This had to be done by a SINTEF employee outside of the designated team. To perform the following sea tests the WP 3 and WP 4 were relocated to not interfere with the experiments. The last day was used to perform tests at $1.06m/s$

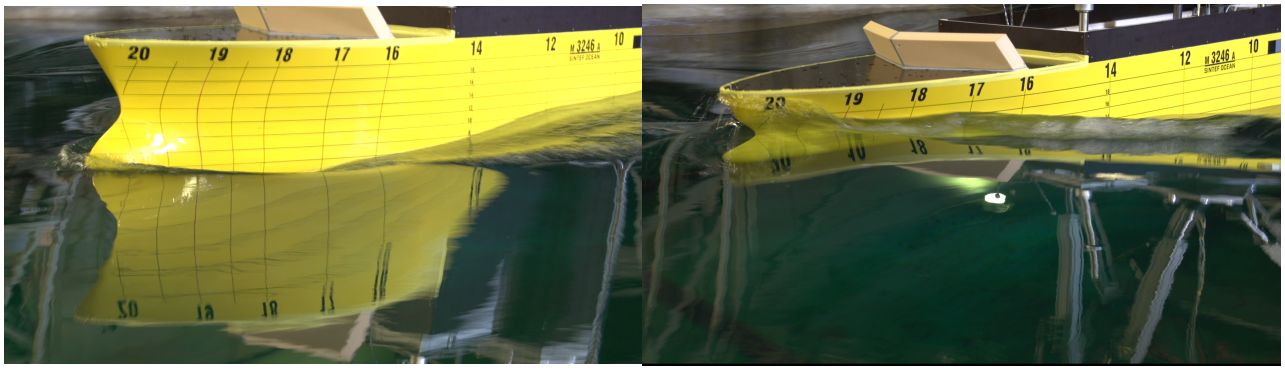
4.4 Observations and uncertainties

This section covers observations made during the experimental runs, which leads to uncertainties in regard to the measurements.

During test with $H > 3.8m$ the frame connected to the hexapod visibly bent due to the force acting on it. This may influence the force measurement as well as the tracking from the OQUS system.
The gears connecting the hexapod to the carriage had slack and lead to small uncertainties in the yaw angle. An unwanted yaw angle will produce unsought sway forces, by splitting the force in two directional components. The gear slack could have been minimised, but would have taken an additional day to do. Because the time was limited it was chosen not to pursue any further.
The middle cylinder had to be changed to a shorter specimen, that lead to a reduced hub in heave and pitch motion, thus waves with a wave height of $H = 7.6m$ or higher could not be tested.

The wave probes positioned on the carriage bent during the test runs as a result of the incoming waves. This influences the accuracy the of the wave probes.
For small waves (up to $H = 2.38$ and $T = 7.8$) the unevenness of the carriage rails can be seen. This means the noise in the same magnitude as the wanted measurement. Consequently we expect a larger error in these measurements.

Starting from a wave of height $H = 3.8$ with a period $T = 9.9$ we observed huge motion at the stern and bow. As it can be seen in Figure 18a the bulbous bow is almost completely out of water and thus changing the waterline area A_{WL}. The change in A_{WL} as well as resulting slamming in stern and bow regions create large viscous forces, which are not considered in the numerical models.



(a) (b)

Figure 18: 45°front view during test, taken from video of run #4120, to show bow movement

The towing tank has been prolonged during its lifetime and the new part $8.5m$ is deeper than the old part $5.8m$. Another difference between old and new part is, that the old part is equipped with wave dampers in form of floaters. Thus waves behave differently in the old and the new part of the tank.

4.5 Pre-processing

The recording of the measured data was switched on shortly before the wave hit the model. This way each data recording contained a short period of calm water state, which was used as zero settings. That enabled to quickly compare the measurement to their respective zero setting also during the experiment. It is also useful, as the wave probes are not set to zero, see Figure 19. But this means, that the measurement data includes unsteady transient part during accelerating and decelerating of the carriage.
The purpose of pre-processing the data was to target a steady interval within the recordings, in which the speed and the incident waves are constant over time, as well as filtering noise and higher harmonics.
The OQUS data was sampled at a sampling rate of 50Hz while all other data was sampled at 200Hz. This allowed to record frequencies as high as 100Hz (25Hz for OQUS) to be found in the data.

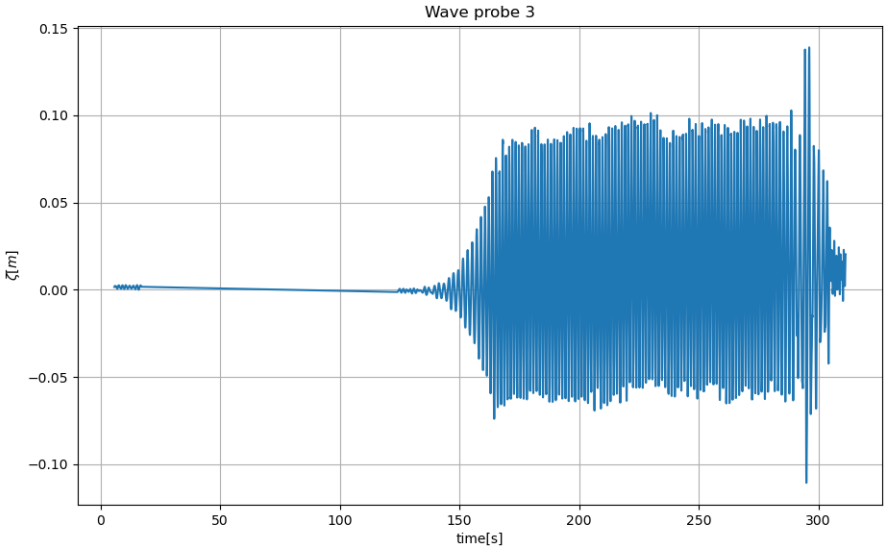


Figure 19: Total timeseries as measured from WP 3 for run #4120

The calculation interval has to fulfil several conditions. It has to exclude any transient part, where the data is unsteady. It has to be separated into new and old tank part, as the external conditions in both parts differ. And the time interval has to be an integer of the periods of the measured data. The last condition is to make sure that the result does not get impaired because of uneven distribution of sampling points.

Figure 20: Z-position as measured by the OQUS system for run #4120, showing plots before and after pre-processing with a low-pass filter

A low-pass filter allows to exclude higher harmonics as well as high frequency noise, see Figure 20. In this example the unprocessed data was especially rough near the peaks. This may be because of viscous effects during slamming, or due to mechanical parts(e.g. in the bearings). The corresponding slamming motion can be seen in Figure 18. A low-pass filter helps to remove the oscillatory parts at the peaks, as it can be seen in Figure 20.

Mean values, e.g. for the added resistance, were calculated using peak data. This was done to avoid inadvertently errors, since not every period is sampled with the exact same amount of sampling points. The added resistance was found by subtracting the calm water resistance from the measured resistance for F_x.

4.6 Wave validation

Since the wave were not calibrated before the experiments the recorded data was used to verify the wave. This was done by using the data from WP 1 and WP 2, which were located near the wave maker. The waves during the chosen calculation interval are not the same as the ones the WP 1 and WP 2 measure at the same time. That is why the time shift had to be taken in account for using the wave progression speed, as well as the carriage position.

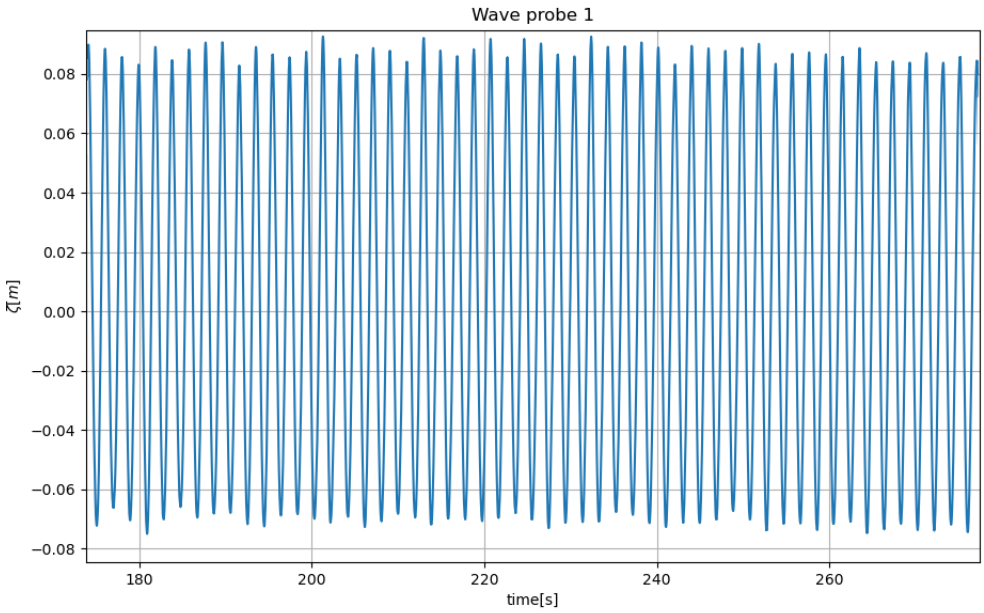


Figure 21: Measurements from WP 1, for run #4120

The python code corresponding to Figure 21 calculates a wave amplitude of 0.156m in model scale, thus 4.99m in fullscale. This is an 4.8% error compared to the input of 4.75m and closer to the wave height expected at #4140 where $H = 4.99m$ and $T = 11.3$.
But during the time the wave travels through the tank it loses energy and changes form. This can be seen for WP 3 in Figure 22. Change in the wave amplitude will inevitably lead to change in the resulting motion and forces and modify the results.

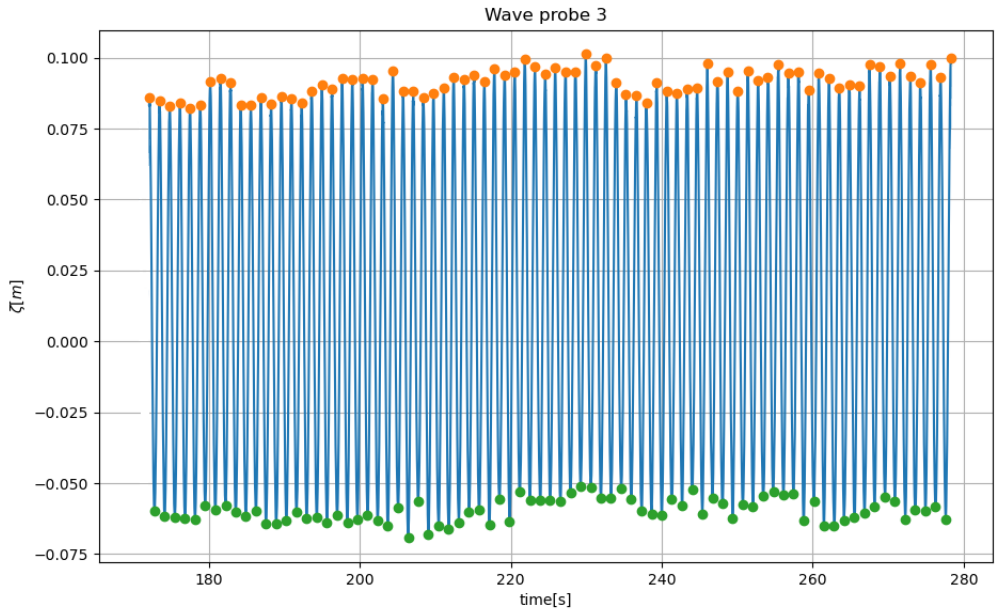


Figure 22: Measurements from WP 3, for run #4120

5 Numerical examination

5.1 ShipX

One of the numerical tools used was ShipX from MARINTEK A/S. The vessel response program application (VERES) in ShipX allows to calculate the hydrodynamic forces as well as vessel motion in waves. For this thesis VERES version 4.09.7 was used.
The input consists of a model geometry and the test set-up consisting of waves and speeds. The model geometry can be seen in Figure 11. VERES then calculates hydrodynamic forces and response motion using strip theory.
This tool allowed to have an additional set of RAOs for pitch, heave and added resistance to compare with the experimental data, or as input in the slowly varying force approximations. The results from the ShipX computations are shown in Appendix B.

5.2 Calm water resistance

The Python implementation of the Holtrop and Mennen method was done according to [Birk, 2019] and validated with data given in the same book and further validated using data for the S175 vessel.
The S175 data was taken from the [26th ITTC, 2011], [R.Vettor et al., 2018], [N.Fonseca and C.G.Soares, 2004b] and [N.Fonseca and C.G.Soares, 2004a], see Table 2.

Table 2: Data for Holtrop and Mennen input

	S175
$U\ [kn]$	$19-24$
$L_{PP}\ [m]$	175
$B\ [m]$	25.4
$D\ [m]$	15.4
$T\ [m]$	9.5
$\nabla\ [m^3]$	24742
$LCB\ [m]$	2.5
$C_B\ [-]$	0.572
$C_M\ [-]$	0.98
$A_V\ [m^2]$	756
$A_T\ [m^2]$	0
$A_{BT}\ [m^2]$	6.0
$h_B\ [m]$	2.0

In his book Birk provided data to verify any Holtrop and Mennen calculation. These values were taken to perform the first validation step. The results were seen as reasonable, as the error was lower than 0.5% and decreased with increasing vessel speed, as shown in Figure 23. This is beneficial as the speed range of the S175 vessel was between 15 and 19 knots, while the speeds tested for the SO Ship were $12kn$ and$15kn$ respectively.

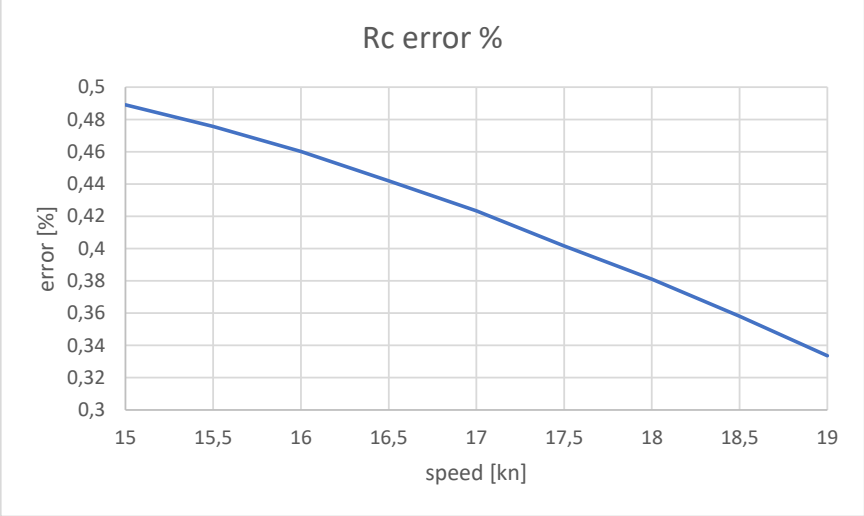


Figure 23: Error Python calculation [%] v speed [kn]

Using the developed Python program and data from table 2 the calm water coefficient C_C and under water surface S were compared to the results from [Tezdogan et al., 2016].

$$C_C = \frac{R_C}{0.5\rho U^2 S} \tag{33}$$

Table 3: Calm water results $S175$

	Tezdogan	Python
S	5371.564	5433.813
$C_C(19kn)$	$2.227 \cdot 10^{-3}$	$2.467 \cdot 10^{-3}$
$C_C(24kn)$	$3.099 \cdot 10^{-3}$	$3.117 \cdot 10^{-3}$

Additionally to the self developed Python program two other programs were used to get an estimation of accuracy for the calm water resistance. The two other programs used were a Java implementation of Holtrop Mennen[Lagemann, 2022] and an online application ShipLab [holtrop online application,].
In Figure 23 the calm water resistance coefficients calculated using Holtrops method are put in comparison to the VERES values calculated by [Tezdogan et al., 2016]. Corresponding with the founding during the first validation step the difference between the VERES values and the Python implementation decreased for higher vessel speeds.

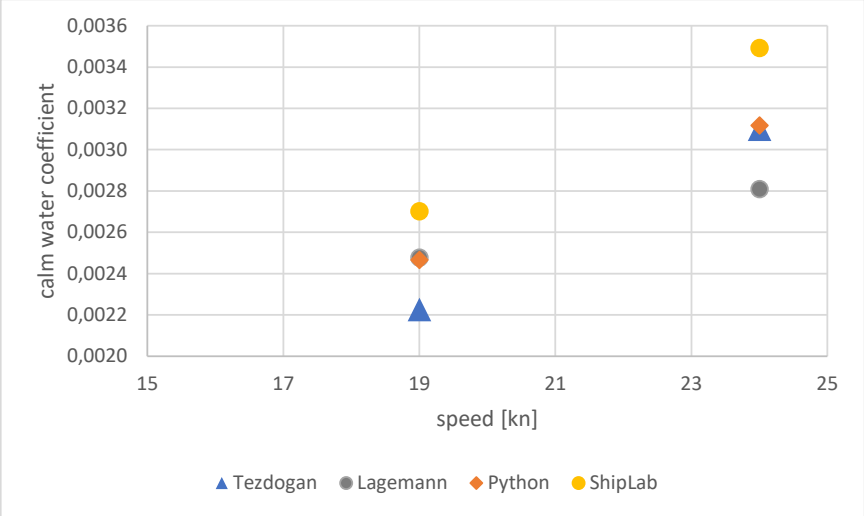
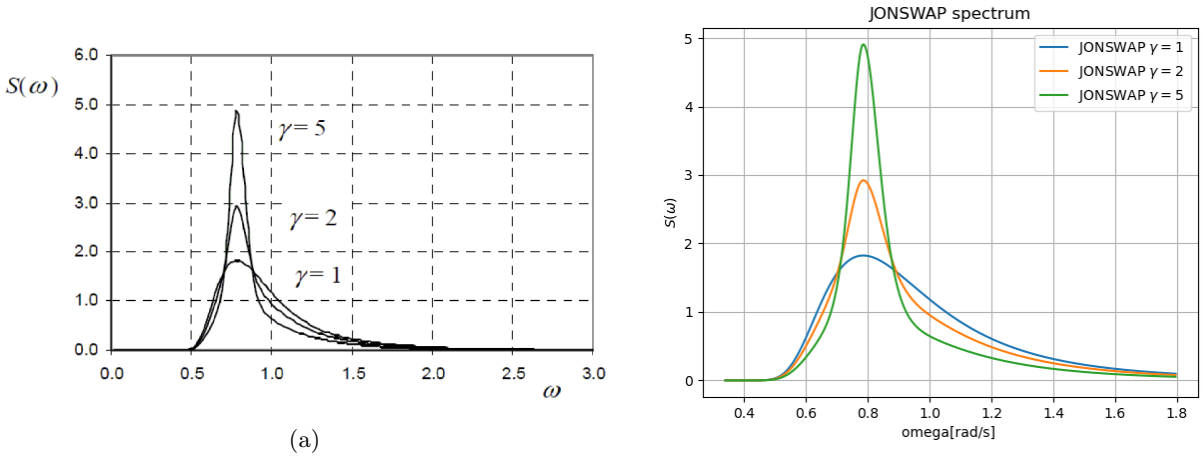


Figure 24: Calm water coefficient v speed in knots

5.3 Slowly varying drift force

The self developed Python program was built to take the pre-processed added resistance transfer function results from the experiment as well as create a JONSWAP spectrum on the basis of the [DNV-GL, 2018] procedure.



(a) (b)

Figure 25: Side by side comparison of the JONSWAP spectrum ($H_s : 4|T_p : 8$)
(a): JONSWAP spectrum given by [DNV-GL, 2018]
(b): JONSWAP spectrum calculated by Python implementation

The energy spectrum was then translated into an amplitude spectrum using the relation stated in Equation 12.

For the Newman approximation the next step was to define for which ω_j the transfer function F_{jj} is an acceptable approximation. IN this paper a range of 5% was chosen with the reasoning that it fulfils the Newman condition of$|\omega_j - \omega_k| \ll 0.5(\omega_j + \omega_k)$.

$$\begin{aligned} |\omega_j - \omega_k| &\ll 0.5(\omega_j + \omega_k) \\ \Rightarrow |0.05\omega_j| &\ll 0.5(1.95\omega_j) \\ \Rightarrow |0.05\omega_j| &\ll 0.975\omega_j \end{aligned} \tag{34}$$

The range for for the Newman condition is not specifically defined, that is why this subject should be of further interest for future investigations.

The next step was to use Equation 31 to calculate the slowly varying drift force for each time step in a three hour fullscale simulation. Each time step has a Δt of 0.2 seconds, this is to include force oscillations with a period of one second, since those were expected to occur.

To model Method 2 the JONSWAP spectrum was made into a time series using Equation 15. And both variants of Method 2 were applied to the irregular sea made out of the JONSWAP spectrum, to get period and amplitude for each regular interval. Then the mean slow drift force for each interval was calculated by using Equation 35:

$$f^{SD} = F_j A_{jj}^2 \tag{35}$$

The slowly varying component is calculated by subtracting the mean slow drift force $\overline{f^{SD}(t)}$ from $f^{SD}(t)$.

The resulting slowly varying drift force time series $f^{SV}(t)$ for Newman approximation and Method 2 were made into histograms to make them easier to compare.

5.4 Irregular sea

The results from the irregular sea states tested during the experiments had to be be treated differently than the results from the regular wave runs. As the datafile of the irregular sea states consisted of five separate runs it had to be divided in five parts, showcased in Figure 26 Each of these intervals treated as a separate test during pre-processing. The added resistance for each run was calculated the same way as for the regular wave runs, by subtracting the calm water resistance from F_x. A histogram was made out of each of the resulting total wave induced force time series $f^W(t)$ and then summed up to reveal the probability of the f^W amplitude of the whole recorded irregular sea state test.

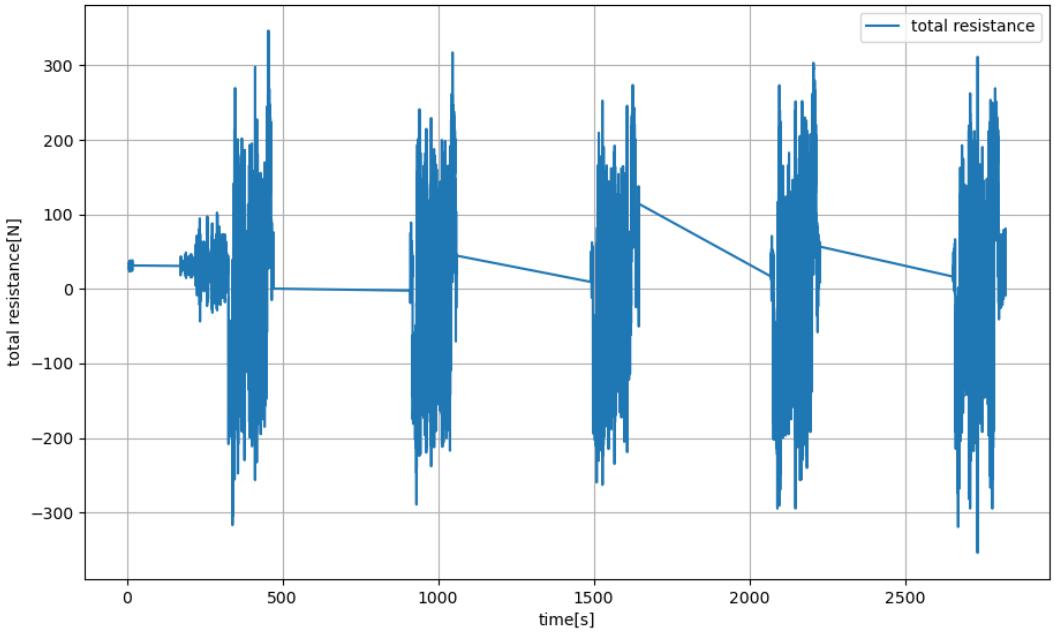


Figure 26: Total resistance as recorded during test #7700

6 Results and discussion

6.1 Calm water resistance

The Holtrop and Mennen Python code was used to calculate the calm water resistance of the SO Ship. The used data is showcased in Table 4.

Table 4: SO Ship data for Holtrop and Mennen input

	SO Ship
$U\ [kn]$	$12 - 15$
$L_{PP}\ [m]$	190
$B\ [m]$	32.22
$D\ [m]$	0
$T\ [m]$	11
$\nabla\ [m^3]$	50128
$LCB\ [m]$	0
$C_B\ [-]$	0.728
$C_M\ [-]$	0.99
$A_V\ [m^2]$	0
$A_T\ [m^2]$	0
$A_{BT}\ [m^2]$	0
$h_B\ [m]$	0

The calculated results differ significantly from the measured data. One main reason is the fact that the water was still in motion during the experiments as the time needed to get absolute calm water would take too long. Another reason are the tank conditions. In the numerical model deep water conditions are assumed, but since the basin depth is limited, shallow water resistance components are unavoidable. Additionally, the Holtrop and Mennen method is an approximation and does have an error range.

Table 5: calm water resistance in fullscale

	12kn	15kn
Holtrop/Mennen	$3.46 \cdot 10^5$ N	$5.93 \cdot 10^5$ N
Experiment	$5.30 \cdot 10^5$ N	$9.86 \cdot 10^5$ N

6.2 Wave response

For the response amplitude operator calculation the wave amplitude from WP 3 was used, as it was placed closest to the model.

Comparing the plots for heave and pitch responds amplitude operators, the larger difference in heave RAO becomes evident. While the pitch RAO matches the numerical results from ShipX, the heave RAO deviates. These differences include a less pronounced peak at a incident wave period of approximately 13s and a deflection from the ShipX values at lower periods.

Because of set-up limitations no test waves with a period longer than 15.6s could be performed. For this reason it can not be said whether the experimental results for the pitch RAOs would converge towards the ShipX plot at higher incident wave periods. The values for shorter periods seem to match as well, despite the fact that the data recorded for lower periods is less reliable compared to larger waves. In the ShipX plot a small peak at period about 8 seconds can be identified, this peak is indicated in the experimental results as well. But uncertainties and errors during the measurements makes it impossible to be absolute certain, whether the peak is genuine or only occurred due to fluctuating data.

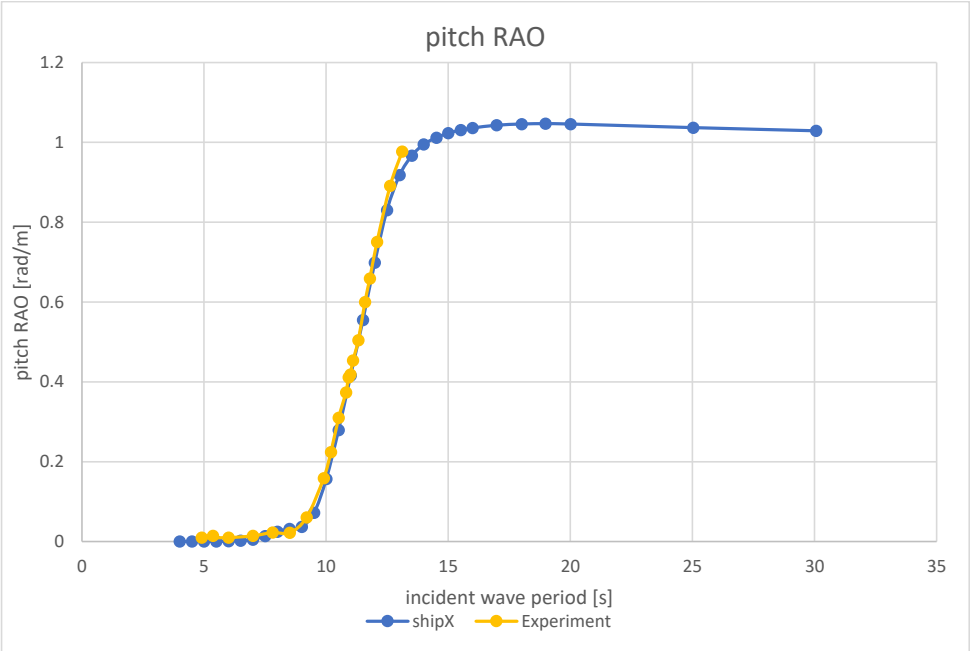


Figure 27: Pitch RAOs over incident wave period: experimental data v ShipX results

The experimental heave RAO has a peak which is about 30% lower than the ShipX plot peak. This suggests a large difference in damping, specifically a higher damping in the experiment setup.
This difference may come from the fact, that ShipX is based on linear strip theory, and thus neglects second and higher order forces as well as viscous effects. Additional mechanical damping, from the fastening mechanism in the experimental set-up might also contribute to the higher damping. Since the set-up did not allow for larger waves it is not possible to make well-founded assumptions on how the heave RAO would behave for larger incident wave periods. It would have been of interest to see if the 30% difference would sustain or if the experimental plot would converge towards the ShipX plot.
The additional uncertainty for lower wave periods can be seen, as the experimental plot rises for low periods. Similar to the pitch RAO the heave RAO has a peak at approximately 8 seconds. But the small peak from the experiments seems to be shifted slightly towards lower periods.

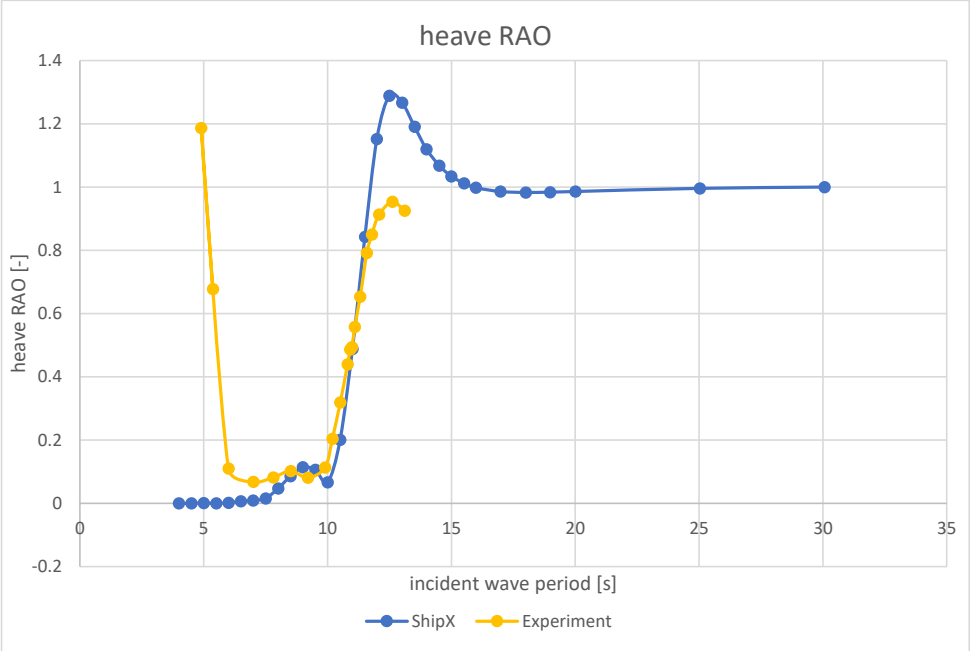


Figure 28: Heave RAOs over incident wave period: experimental data v ShipX results

The results for very small periods seem to be numerical or measurement errors, as they rise above the eigen-frequency values. By taking a look at the heave values from the experiments, it can be seen that they are in order of magnitude of $10^{-3}m$ with the wave amplitude being in the same order of magnitude. Therefore measurements errors or noise in the order $1mm$ have great impact on the resulting RAO. These errors could for example be caused by waves reflected from the tank walls.
Similar results have been found by Josephs investigation using MATLAB[J.C.Sajan, 2021]. That indicates the problem may be located during the data measurement.

6.2.1 Added resistance

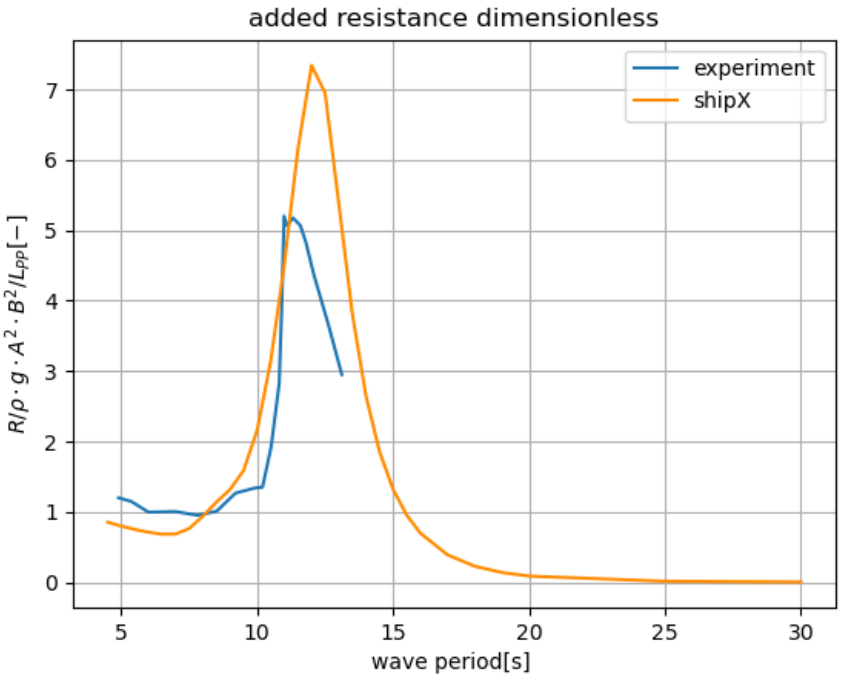


Figure 29: Dimensionless added resistance curve over incident wave period: experimental data v ShipX

The experimental added resistance curve has a lower peak at a lower incident wave period, compared to ShipX. Similar to the pitch and heave RAO, it can not be seen whether the curve would further digress from the ShipX results for larger periods or not. In context to the over prediction in the heave RAO the difference in added resistance seems to be, due to the fact that the same effects that impact the heave RAO, i.e. linear strip theory and neglect of viscous effects, also effect the added resistance. As for wave length close to he vessel length, the added resistance is dominated by wave radiation coupled to the heave motion. The peak difference of approximately 30% is also the same, while the eigenfrequency is slightly shifted towards lower periods. The shift in heave or added resistance could have been influenced by the fastening of the model, which may influence the stiffness in forward direction. Or might be the consequence of an increase in mass due to added mass, which can result from remaining water motion from previous runs.
These suspicions can be supported by looking at the definition of the eigenfrequency:

$$\omega_e = \sqrt{\frac{K}{M}} \tag{36}$$

The added resistance response amplitude operator is computed by dividing the dimensional added resistance by the corresponding wave amplitude square, hence any errors in the added resistance will also turn up in the added resistance RAO.
In order to use the RAO for the Newman approximation and Method 2, the RAO has to be interpolated to fit the corresponding periods. Figure 30 shows the interpolations used for Method 2.

It can be seen that the ω range of Method 2 is not the same as the range found by experiments. This leads to additional uncertainties as all datapoints underneath the experimental frequency range are set to be equal to the lowest value from the experimental added resistance RAO.
This adds to the inevitable uncertainties that comes with simulating short waves, as numerical tools (e.g. ShipX) also lose accuracy when calculating short waves.

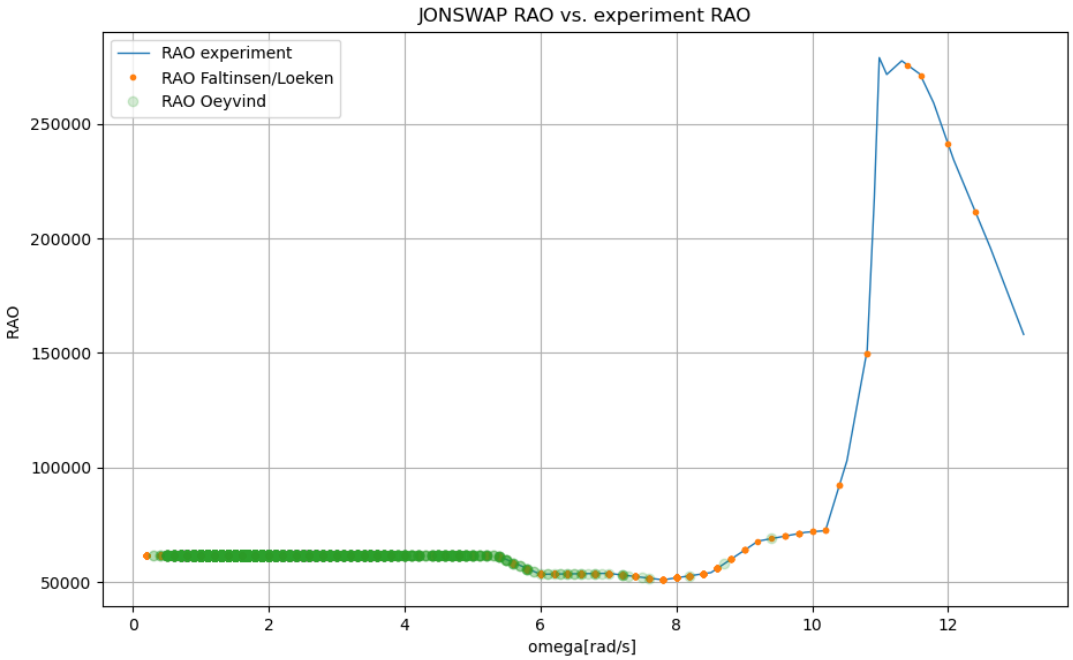


Figure 30: Interpolation of added resistance RAO over angular frequency for Method 2

6.3 Newman approximation

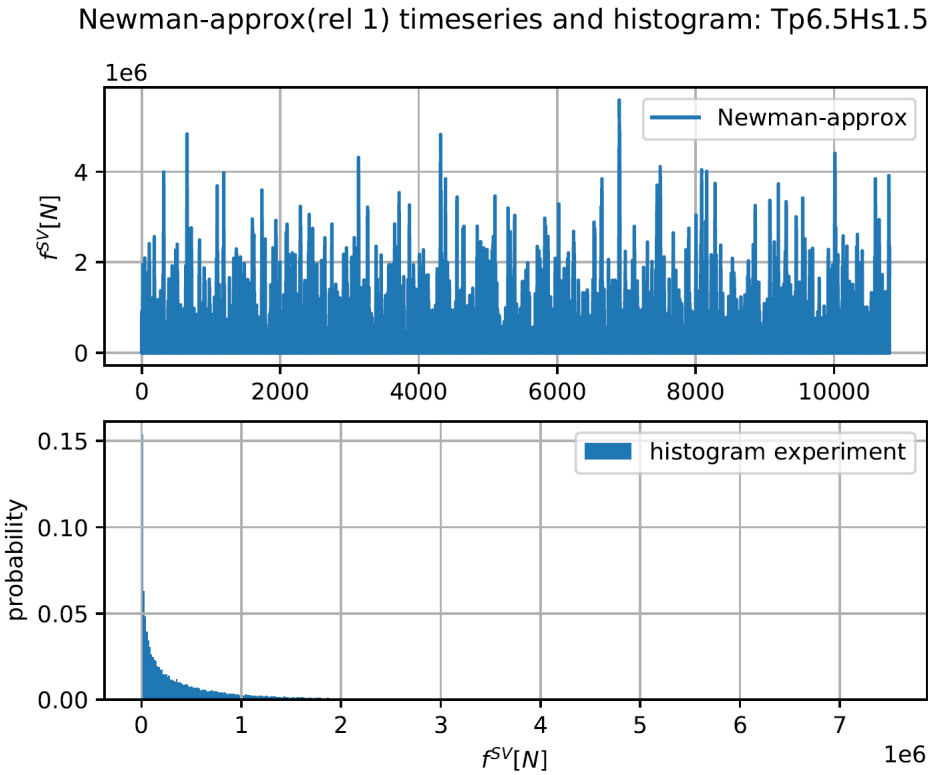
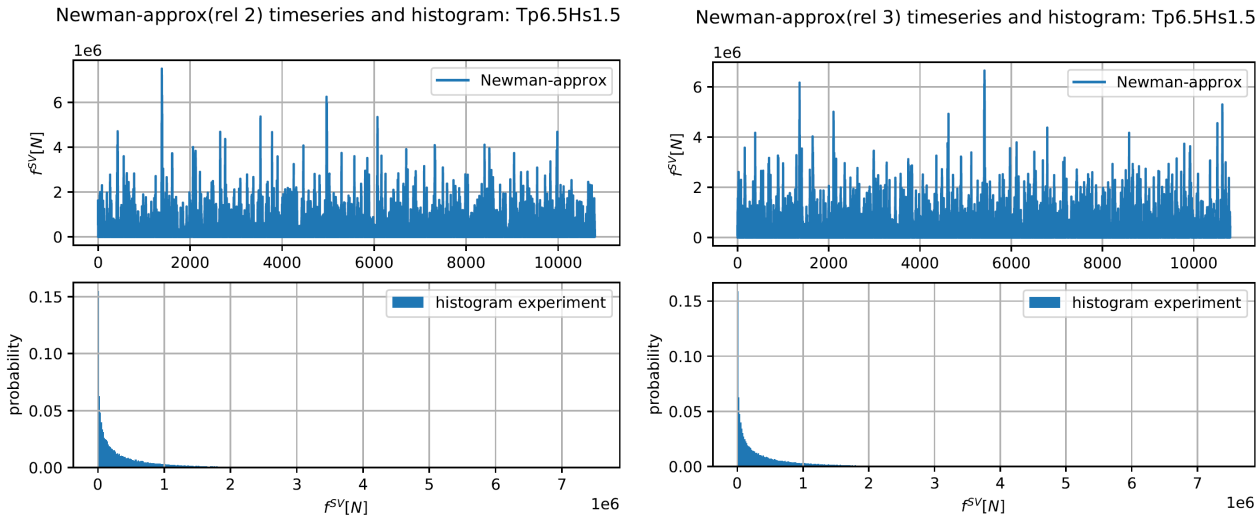


Figure 31: Newman approximation; time series and histogram, realisation 1

The peaks from the resulting time series of the slowly varying drift force calculated by the Newman approximation are sampled into a histogram and normalised by the total amount of peaks. This way the probability of a force amplitude range is obtained.
By choosing the same force amplitude range different realisations of the Newman approximation can be compared in a single plot, see Figure 33.



(a) (b)

Figure 32: Newman approximation; time series and histogram, realisation 2(a) and 3(b)

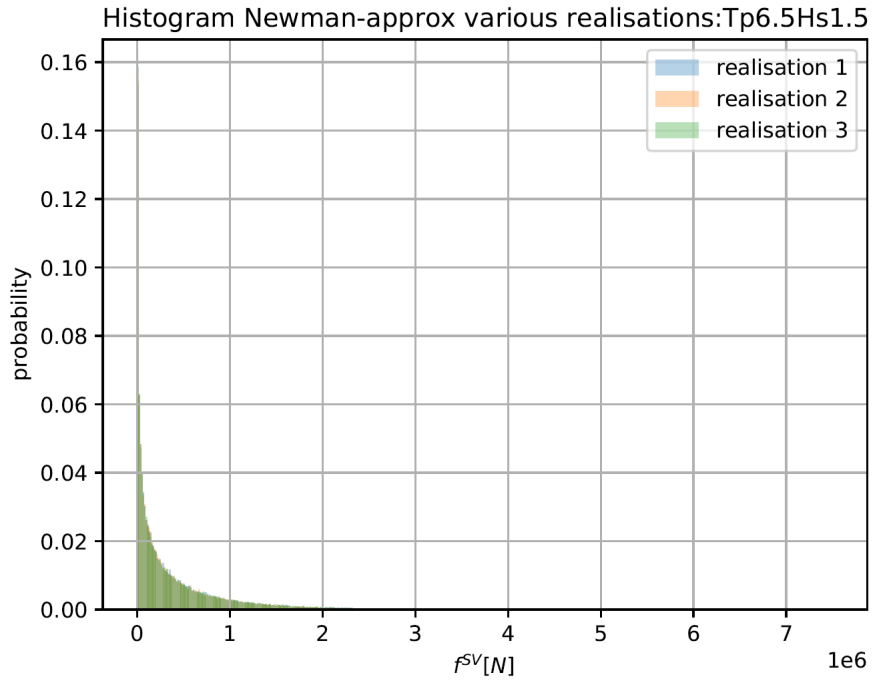


Figure 33: Histogram comparison of Newman approximation realisation 1,2 and 3

While all runs have a slightly different time series, the histogram shows, that the overall distribution is similar. The histogram show a force amplitude probability, which decreases rapidly with increasing force amplitude.

The histogram has the form of half a standard distribution.
Because the Newman approximation has to be summed up over the viable frequency range the calculation time is relatively high. The Python program developed for this thesis in combination with the available computational prowess it took about 50 minutes to calculate one realisation of $f^{SV}(t)$.

For further comparison the Newman approximation was taken of the same spectrum as used in run#7700, $H_s = 2.5$ and $T_p = 7.5$.
The force timeseries $f^{SV}(t)$ is linearly dependent on the amplitude response operator from added resistance. Thus by using the ShipX computed RAO instead of the experimentally obtained, the time series will change. Figure 34 shows that the force amplitudes using the RAO obtained by during the experiments covers a wider range of force amplitude values, compared to Figure 35.

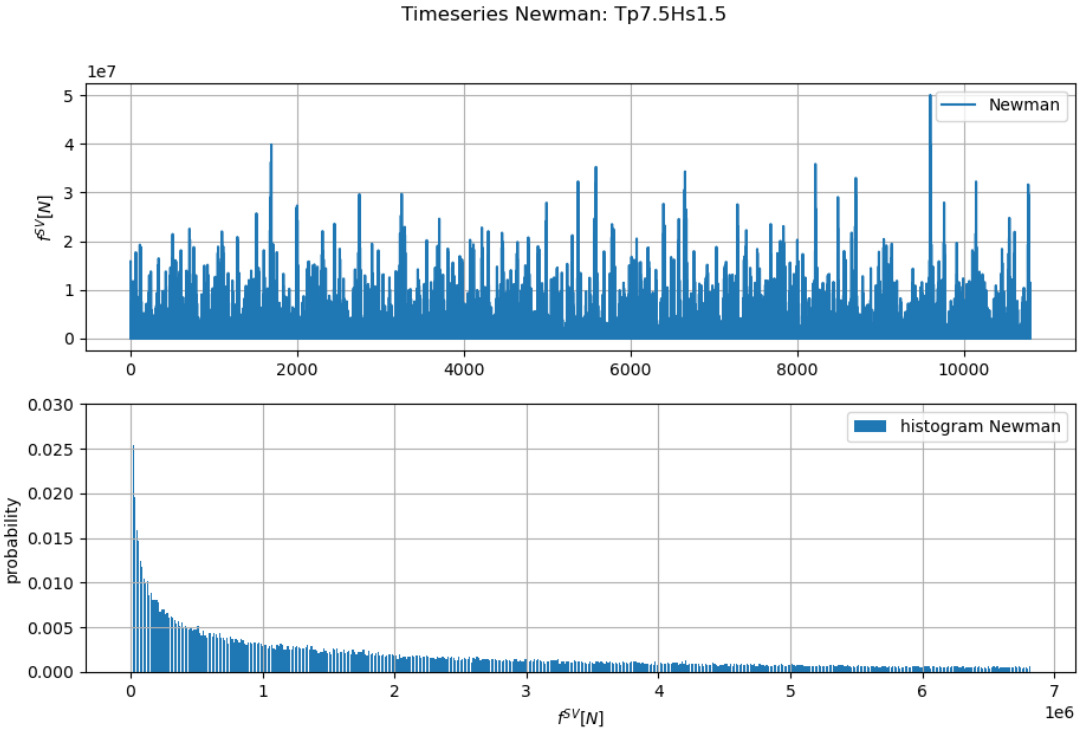


Figure 34: Newman approximation using RAOs obtained through Experiments

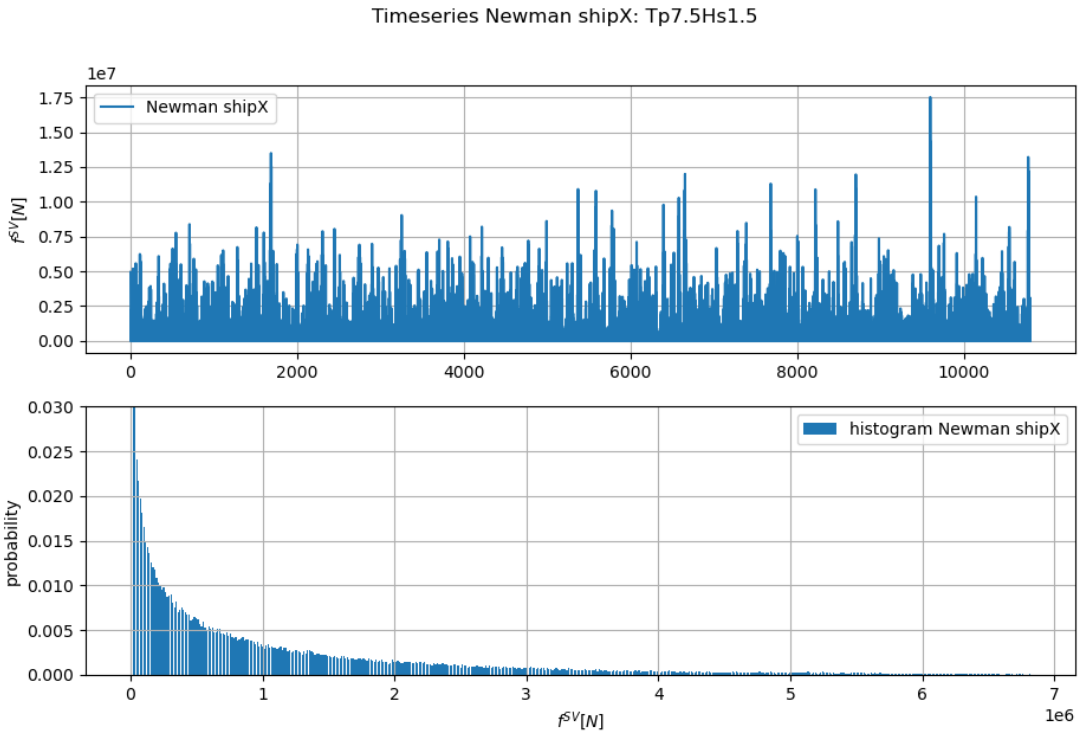


Figure 35: Newman approximation using RAOs obtained through ShipX

6.4 Timedomain approach

Method 2 approximates the irregular sea state into regular intervals. Because the Faltinsen/Løken variant defines an interval between two zero crossings, while the Øyvind variant defines the interval as the range between two zero up-crossings, the Faltinsen/Løken variant results in double the amount of intervals.
Mathematically speaking, this means that the Øyvind variant takes the mean of two intervals, compared to the Faltinsen/Løken variant.
That is why the Øyvind variant has less large amplitudes and periods. This is also reflected in the histograms shown in Figure 38, where it can be seen that the Øyvind variant has more force amplitudes in the lower ranges of f^{SV}.

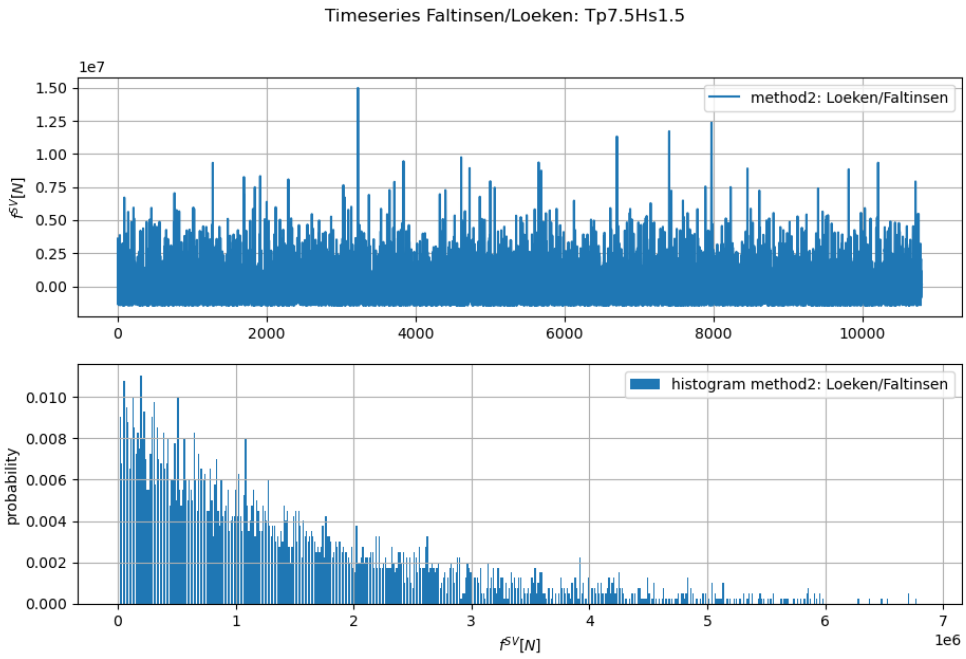


Figure 36: f^{SV} timeseries and histogram Faltinsen/Løken variant

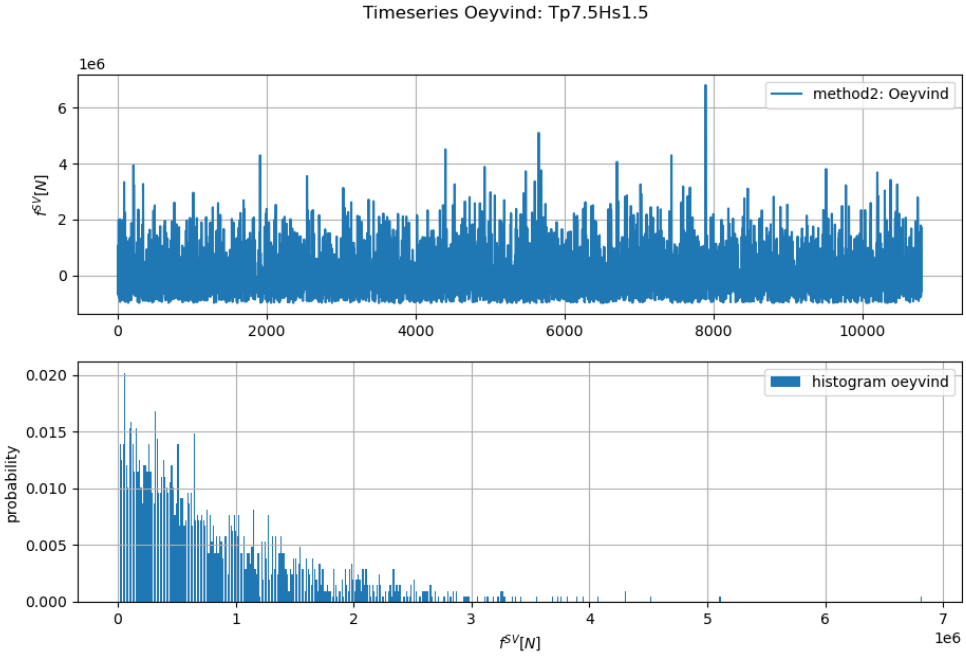


Figure 37: f^{SV} timeseries and histogram Øyvind variant

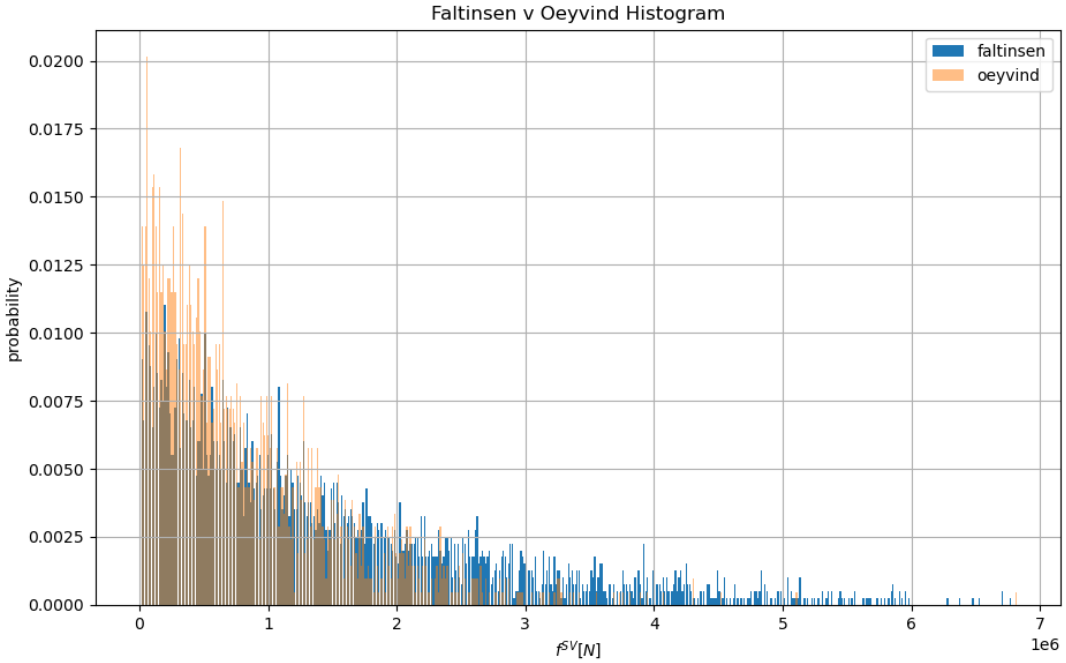


Figure 38: Histogram comparison Faltinsen/Løken v Øyvind, histograms taken from Figure 36 and Figure 37

The calculation time with available computational prowess was about 5 minutes, for both variants together.

6.5 Newman approximation v Method 2

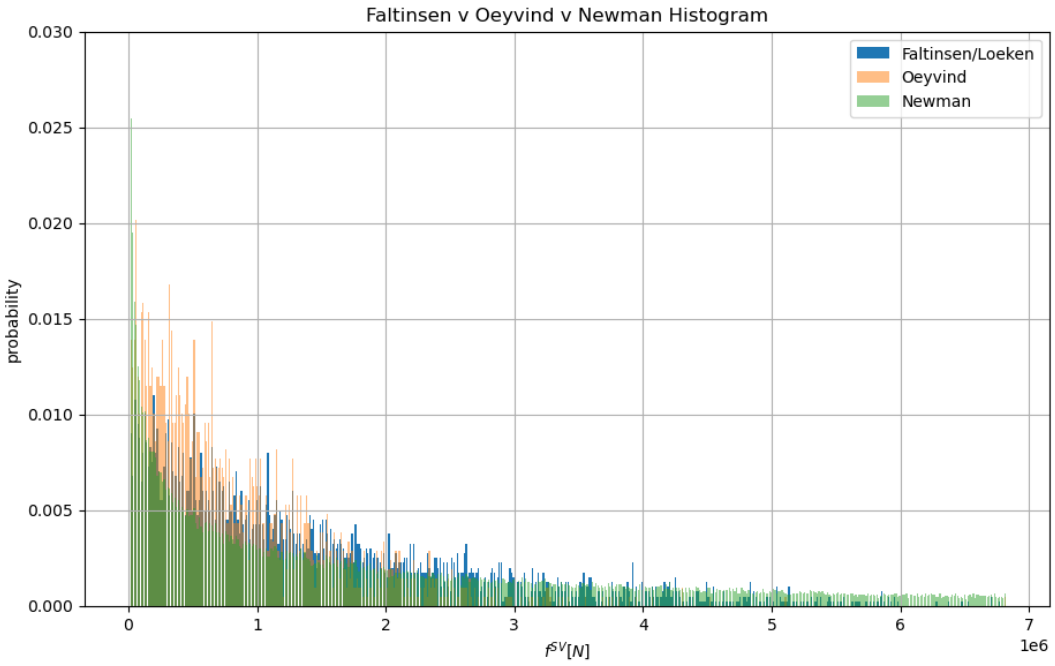


Figure 39: Histogram comparison Newman v Method 2, histograms taken from Figure 38 and Figure 34

Figure 39 shows the histograms of Newman approximation and both variants of Method 2 in one plot. The random phase angle used for the Newman approximation is the same as the one used to create the irregular sea state for Method 2
The distribution for the Faltinsen/Løken variant is very similar to the Newman approximation, with a shape close to half a normal distribution with a peak at zero. In this case the peak is less pronounced in comparison to the Newman approximation. Most of the amplitudes in the Newman approximation are in the amplitude ranges close to zero, the shape shows tendencies of half a normal distribution with a pronounced peak at zero.

Figure 39 shows that the Newman approximation covers a similar range as the Faltinsen/Løken variant, with similar probability distributions.

Method 2 approximates the irregular sea state in intervals of regular seas, thus many of the lower amplitudes are overshadowed by the larger amplitudes. The combination of small and large amplitude leads to even larger amplitudes. The Øyvind variant practically takes the mean over two intervals of the Faltinsen/Løken variant, that is the reason why the probability distribution for the Øyvind variant is less concentrated on the lower and higher f^{SV} ranges.

By looking at the dimensional added resistance we can see that f^{SV} is of the same order of magnitude as the added resistance.

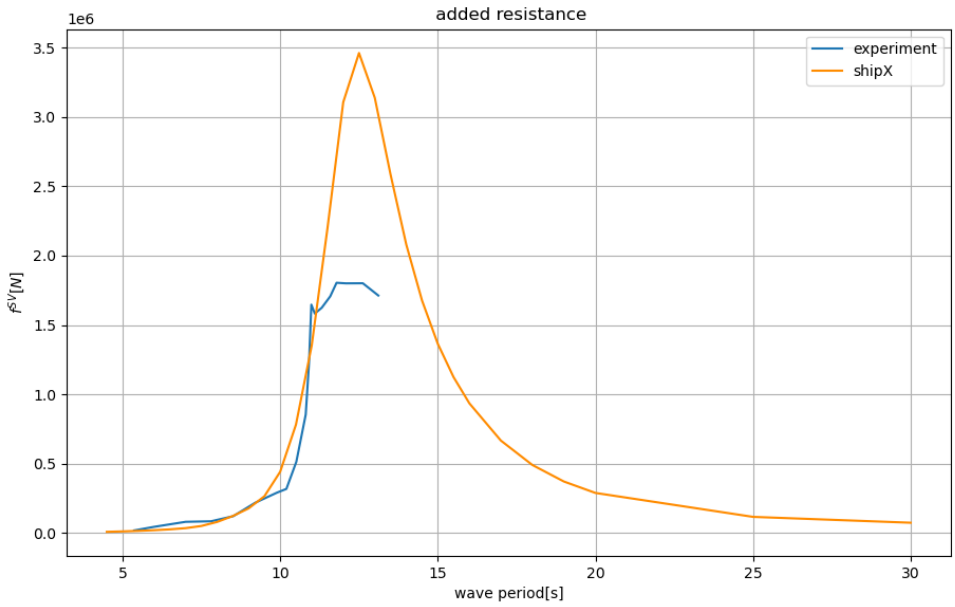


Figure 40: Dimensional added resistance[N] over incident wave period[s]

6.6 Experimental results: irregular sea

As it has been stated in section 5.4 each irregular sea state test included five runs. In Figure 41 the resulting total wave induced force time series for the first run of test #7700 is shown. The remaining timeseries measured during the experiments in irregular sea are given in Appendix C.

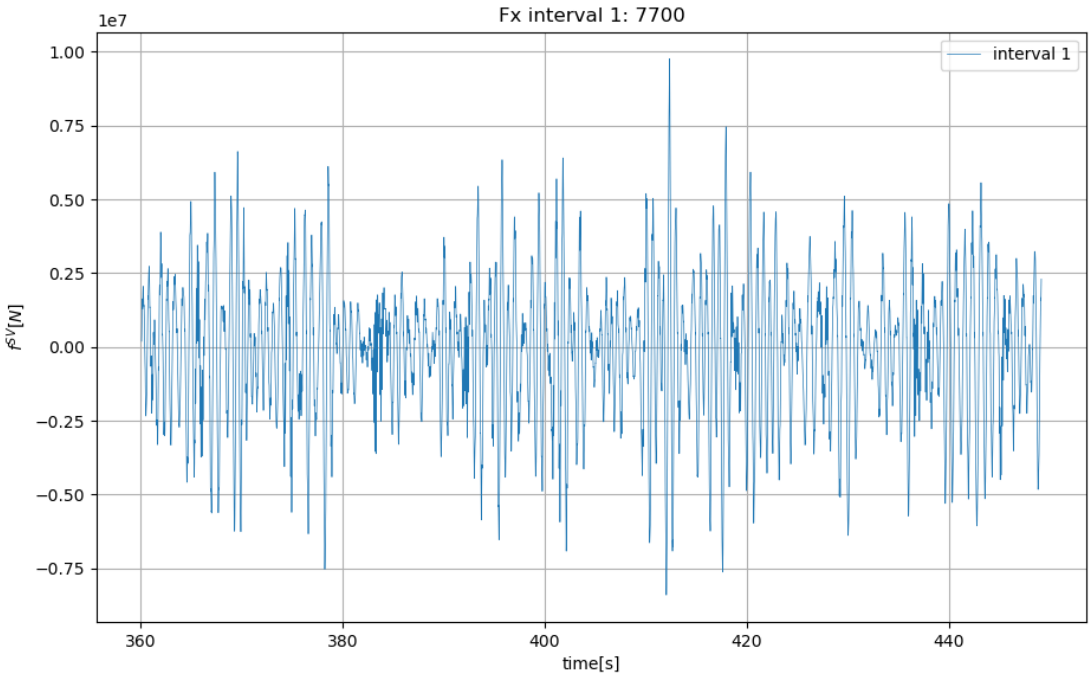


Figure 41: Wave induced force: irregular seaway run #7700, interval 1

The order of magnitude of the total wave induced force is in the vicinity of the numerical results from Method 2, compare to Figure 36 and 37. While the Faltinsen/Løken variant seems to produce higher results in magnitude compared to the Øyvind variant, the results are for the most part still within the range of the experimental collected data. The obtained results from the Newman approximation are within a broader range than experimental results, see Figure 31.

Figure 42 depicts he combined probabilities of all five intervals and reveals a shape close to a normal distribution with two additional peaks, for simplicity the possible shape of the normal distribution is marked in red.
The normal distribution corresponds to the linear wave induced force, which leads to the peaks being the second order non-linear wave force.
The range force range depicted by these peaks are of interest when comparing to the numerical result of f^{SV}.

Accurate comparisons are difficult, as the results from the experimental runs contain the linear wave induced component. Thus the shape of the force amplitudes for the second order non-linear component is hard to tell. But the range of second order non-linear wave component seem to reach from 0 up to a point of $2 \cdot 10^6 N$ and is thus in the same range as most f^{SV} amplitudes from the Newman approximation as well as Method 2.

For more accurate results the linear wave induced force as well as the mean second order non-linear wave force need to be separated. But the results in Figure 42 give an idea of the expected range in which the second order non-linear wave force will be located.

Because the experimental tests of run #7700 and run #7710 only corresponds to 42 minutes of fullscale simulation, it is difficult to make definitive statements about the viability to compare $f^{SV}(t)$. Longer simulation

time would return in more accurate results.

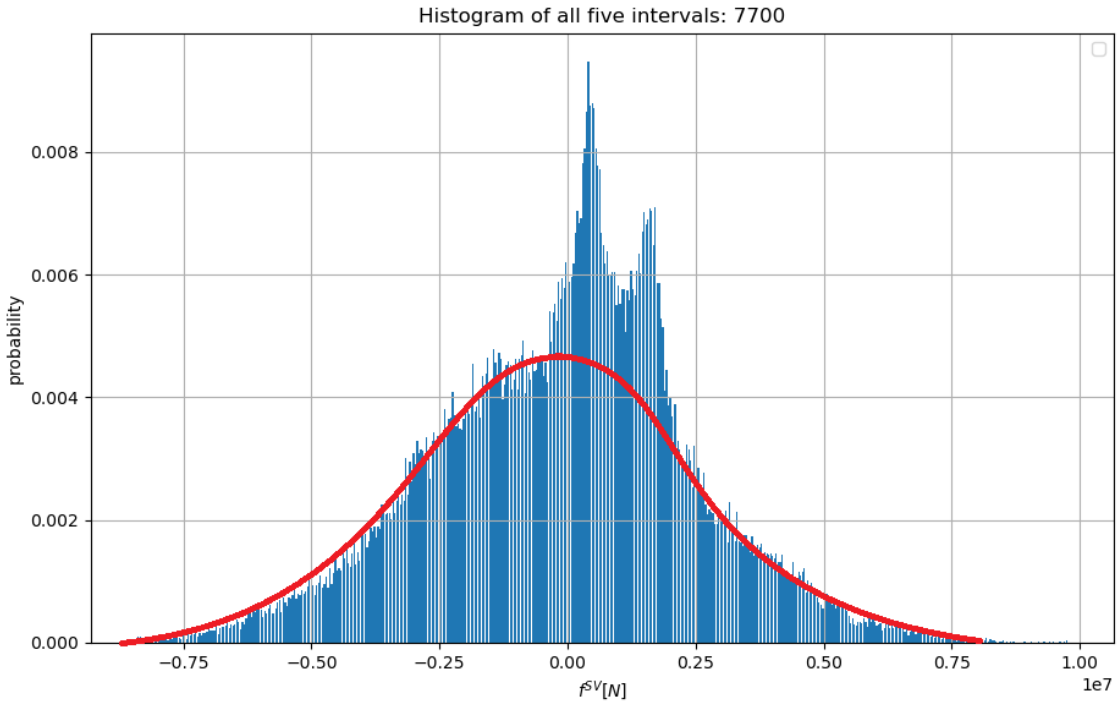


Figure 42: Irregular seaway run #7700, histogram for all five intervals combined; with natural distribution markings

7 Conclusion

A comparison between experimental and numerical values showed a discrepancy especially under small wave conditions, including calm water conditions. Heave and added resistance highlighted a difference in resonance condition, where the perceived wave length is equal to the ships L_{PP}.
The large difference in calm water resistance between numerical Holtrop an Mennen results and experimental measured data, suggest inaccuracies during test conditions. To confirm these suspicions additional tests would be recommended, where the mentioned uncertainties would be considered. It can also be recommended to calculate the added resistant curves using additional numerical tools not based on linear strip theory, akin to the comparison of [R.Nabergoj and J.Prpić-Oršić, 2007] This would give a better comprehension of the added resistance and show predispositions of the strip theory.

The numerical acquired drift force data showed different weaknesses and strength. The histogram of the Newman approximation shows similar tendencies as the experimental tests, with the addition of higher force amplitudes. While both variants from Method 2 have a similar force amplitude range as the experimental irregular wave tests, especially the Øyvind variant.

The distribution of slowly varying force amplitudes for the experimental data can only be suspected, as the data is not available. And making assumptions on the basis of Figure 42 remains highly uncertain.
Nevertheless does the Newman approximation calculated using the ShipX RAO show better agreement with the expected range. These might be because of the uncertainties in the added resistance RAO interpolation, as the added resistance curve resulting from the ShipX calculation is defined over a wider incident period range, see Figure 40.

What was also found during the investigation of the slowly varying drift force is the fact, that the computation of f^{SV} using the Newman approximation has a runtime which is about 10 time larger than the needed runtime for Method 2. The combination of less computational time and simultaneously having the same accuracy, makes Method 2 a timesaving alternative compared to the Newman approximation. That way Method 2 and the Faltinsen/Løken variant especially seem to be the superior numerical method, as it give very similar results, see Figure 39. But to make an well-founded decision more investigation is needed.
This thesis assumed the Newman condition $|\omega_j - \omega_k| \ll 0.5(\omega_j + \omega_k)$ to be met at 5% difference between ω_j and ω_k. Different differences might impact the accuracy of the result, but that additional investigation to be proven.

Further investigations should include experimental simulation of least three hour fullscale time. As the random nature of the time varying drift force increases the uncertainties for shorter simulation time.

References

[26th ITTC, 2011] 26th ITTC (2011). The seakeeping committee, final report and recommendations to the 26th ittc.

[Birk, 2019] Birk, L. (2019). *Fundamentals of Ship Hydrodynamics: Fluid Mechanics, Ship Resistance and Propulsion*. John Wiley & Sons Ltd.

[C.M.Larsen et al., 2019] C.M.Larsen, W.Lian, E.E.Bachynski, T.Kristiansen, and D.Myrhaug (2019). "lecture notes - tmr 4182 marine dynamics". Department of Marine Technology, Faculty of Engineering Science and Technology, NTNU.

[Committee, 2005] Committee, M. E. P. (2005). "prevention of air pollution from ships – marpol annex vi - proposal to initiate a revision process". submitted by Finland, Germany, Italy, Netherlands, Norway, Sweden and the United Kingdom.

[DNV-GL, 2018] DNV-GL (2018). "class guidelines for wave loads". Edition January 2018.

[F.H.Hsu and K.A.Blenkarn, 1970] F.H.Hsu and K.A.Blenkarn (1970). Analysis pf peak mooring force caused by slow vessel drift oscillation in random seas. *Offshore Technology conference*.

[H.Maruo, 1960] H.Maruo (1960). The drift of a body floating on waves. *Journal of ship research*, 4(3):1–10.

[holtrop online application,] holtrop online application, S. last accessed on 9.1.2020.

[H.Söding, 1982] H.Söding (1982). "lecture notes - bewegung und belastungen der schiffe im seegang". Institute of Manoeuvring and Fluiddynamics, TU Hamburg-Harburg.

[H.Söding, 1995] H.Söding (1995). "lecture notes - manövrierfähigkeit von schiffen". Institute of Manoeuvring and Fluiddynamics, TU Hamburg-Harburg.

[J.A.Pinkster and Oortmerssen, 1977] J.A.Pinkster and Oortmerssen, G. (1977). "computation of the first and second order wave forces on oscillating bodies in regular waves".

[J.C.Sajan, 2021] J.C.Sajan (2021). "added resistance and effect of drift angle in wind assisted ships".

[J.Gerritsma and W.Beukelman, 1972] J.Gerritsma and W.Beukelman (1972). Analysis of the resistance increase in waves of a fast cargo shipe. *Intern. Shipbuilding Progr.*, 19(217):285–293.

[J.Holtrop and G.G.J.Mennen, 1982] J.Holtrop and G.G.J.Mennen (1982). An approximate power prediction method.

[J.M.J.Journée, 1976] J.M.J.Journée (1976). Prediction of speed and behaviour of a ship in seaway. *ISP*, 23(256). reprinted 2001.

[J.N.Newman, 1974] J.N.Newman (1974). Second-order, Slowly-varying Forces on Vessels in Irregular Waves.

[J.N.Newman, 1990] J.N.Newman (1990). Second-harmonic wave diffraction at large depth. *Journal for Fluid Mechanics*, 213:59–70.

[J.N.Newman, 1996] J.N.Newman (1996). The second-order wave force on a vertical cylinder. *Journal for Fluid Mechanics*, 320:417–443.

[K.Hasselmann et al., 1973] K.Hasselmann, T.P.Barnett, E.Bouws, and H.Carlson (1973). "measurements of wind-wave growth and swell decay during the joint north sea wave project (jonswap)".

[Lagemann, 2022] Lagemann, B. (2022). Conceptual Ship Design Platform.

[M.Kim et al., 2017] M.Kim, O.Hizir, O.Turan, S.Day, and A.Incecik (2017). Estimation of added resistance and ship speed loss in a seaway. *Ocean Engineering*, 141:465–576.

[N.Fonseca and C.G.Soares, 2004a] N.Fonseca and C.G.Soares (2004a). Experimental investigation of the non-linear effects on the statistics of vertical motions and loads of a containership in irregular waves. *Journal of Ship Research*, 48(2):147–167.

[N.Fonseca and C.G.Soares, 2004b] N.Fonseca and C.G.Soares (2004b). Experimental investigation of the non-linear effects on the vertical motions and loads of a containership in regular waves. *Journal of Ship Research*, 48(2):118–147.

[O.M.Faltinsen, 1998] O.M.Faltinsen (reprint 1998). *Sea Loads on Ships and Offshore Structures*. Cambridge University Press 1990.

[O.M.Faltinsen, 2009] O.M.Faltinsen (reprint 2009). *Hydrodynamics of High-Speed Marine Vehicles*. Cambridge University Press 2006.

[O.M.Faltinsen and A.E.Løken, 1980] O.M.Faltinsen and A.E.Løken (1980). Slow drift oscillations of a ship in irregular waves. *Modelling, Identification and Control*, 1(4):195–213.

[O.M.Faltinsen et al., 1980] O.M.Faltinsen, K.J.Minsaas, N.Liapis, and S.O.Skjørdal (1980). "prediction of resistance and propulsion of a ship in a seaway".

[Prof.Dr.-Ing.M.Abdel-Maksoud, 2016] Prof.Dr.-Ing.M.Abdel-Maksoud (2016). "lecture notes - skriptum zur vorlesung schiffdynamik". Institute of Manoeuvring and Fluiddynamics, TU Hamburg-Harburg.

[Program, 2015] Program, U. N. D. (2015). "sustainable development goals".

[R.Nabergoj and J.Prpić-Oršić, 2007] R.Nabergoj and J.Prpić-Oršić (2007). "a comparison of different methods for added resistance prediction".

[Rostock,] Rostock, U. "strip theory". Chair of Modelling and Simulation, University Rostock.

[R.Vettor et al., 2018] R.Vettor, J.Prpić-Oršić, and C.G.Soares (2018). Impact of wind loads on long-term fuel consumption and emissions in transoceanic shipping. *Brodogradnja*, 69(4):15–28.

[Tezdogan et al., 2016] Tezdogan, T., Incecik, A., Turan, O., and Kellett, P. (2016). Assessing the impact of a slow steaming approach on reducing the fuel consumption of a containership advancing in head seas. *Transportation Research Procedia*, 14:1659–1668.

[T.Kristiansen and Ø.Rabliås, 2021] T.Kristiansen and Ø.Rabliås (2021). A rational model for maneuvering in irregular wavs with the effect of waves on the rudder inflow taken into account – working title. *Ocean Engineering*.

A Wave parameter for experimental tests

Table 6: Test runs during experiment with zero speed

# wave series	$v_S[kn]$	$\mu[deg]$	$H[m]$	$T[s]$	λ/L_{PP}	cancelled
1000	0	0	0	0	0	
2000	0	0	1.9	7	0.4	
2010	0	0	2.38	7.8	0.5	
2020	0	0	3.33	9.2	0.7	
2030	0	0	4.28	10.5	0.9	
2040	0	0	4.75	12.1	1	
2050	0	0	5.7	12.1	1.2	
2060	0	0	6.65	13.1	1.4	
2070	0	0	7.6	14	1.6	
3000	0	9	1.9	7	0.4	
3010	0	9	2.38	7.8	0.5	
3020	0	9	3.33	9.2	0.7	
3030	0	9	4.28	10.5	0.9	
3040	0	9	4.75	11	1	
3050	0	9	5.7	12.1	1.2	
3060	0	9	6.65	13.1	1.4	
3070	0	9	7.6	14	1.6	

Table 7: Test runs during experiment, relevant for RAO calculations

# wave series	$v_S[kn]$	$\mu[deg]$	$H[m]$	$T[s]$	λ/L_{PP}	cancelled
4000	15	0	0	0	0	
4010	15	0	0.95	4.9	0.2	
4020	15	0	1.43	6	0.3	
4030	15	0	1.9	7	0.4	
4040	15	0	2.38	7.8	0.5	
4050	15	0	2.85	8.5	0.6	
4060	15	0	3.33	9.2	0.7	
4070	15	0	3.8	9.9	0.8	
4080	15	0	4.04	10.2	0.85	
4090	15	0	4.28	10.5	0.9	
4100	15	0	4.51	10.8	0.95	
4110	15	0	4.66	10.9	0.98	
4120	15	0	4.75	11	1	
4130	15	0	4.85	11.1	1.02	
4140	15	0	4.99	11.3	1.05	
4150	15	0	5.23	11.6	1.1	
4160	15	0	5.46	11.8	1.15	
4170	15	0	5.7	12.1	1.2	
4180	15	0	6.18	12.6	1.3	
4190	15	0	6.65	13.1	1.4	
4200	15	0	7.6	14	1.6	X
4210	15	0	8.55	14.8	1.8	X
4220	15	0	9.5	15.6	2	X
7777	12	0	0	0	0	
7500	12	0	0.95	4.9	0.2	X
7510	12	0	1.9	7	0.4	
7520	12	0	2.85	8.5	0.6	X
7530	12	0	3.33	9.2	0.7	
7540	12	0	3.8	9.9	0.8	
7550	12	0	4.04	10.2	0.85	
7560	12	0	4.28	10.5	0.9	
7570	12	0	4.51	10.8	0.95	
7580	12	0	4.66	10.9	0.98	X
7590	12	0	4.75	11	1	
7600	12	0	4.85	11.1	1.02	X
7610	12	0	4.99	11.3	1.05	
7620	12	0	5.23	11.6	1.1	
7630	12	0	5.46	11.8	1.15	X
7640	12	0	5.7	12.1	1.2	
7650	12	0	6.18	12.6	1.3	
7660	12	0	6.65	13.1	1.4	X
7670	12	0	7.6	14	1.6	X
7680	12	0	8.55	14.8	1.8	
7690	12	0	9.5	15.6	2	

Table 8: Test runs during experiment, relevant for Joseph's thesis

# wave series	$v_S[kn]$	$\mu[deg]$	$H[m]$	$T[s]$	λ/L_{PP}	cancelled
5000	15	9	1.43	6	0.3	
5010	15	9	1.9	7	0.4	
5020	15	9	2.38	7.8	0.5	
5030	15	9	2.85	8.5	0.6	
5040	15	9	3.33	9.2	0.7	
5050	15	9	3.8	9.9	0.8	
5060	15	9	4.28	10.5	0.9	
5070	15	9	4.75	11	1	
5080	15	9	5.23	11.6	1.1	
5090	15	9	5.7	12.1	1.2	
5100	15	9	6.65	13.1	1.4	
5110	15	9	7.6	14	1.6	X
6000	15	180	1.43	6	0.3	
6010	15	180	1.9	7	0.4	
6020	15	180	2.38	7.8	0.5	
6030	15	180	2.85	8.5	0.6	
6040	15	180	3.33	9.2	0.7	
6050	15	180	3.8	9.9	0.8	
6060	15	180	4.28	10.5	0.9	
6070	15	180	4.75	11	1	
6080	15	180	5.23	11.6	1.1	
6090	15	180	5.7	12.1	1.2	
6100	15	180	6.65	13.1	1.4	
6110	15	180	7.6	14	1.6	
7000	15	189	1.43	6	0.3	
7010	15	189	1.9	7	0.4	
7020	15	189	2.38	7.8	0.5	
7030	15	189	2.85	8.5	0.6	
7040	15	189	3.33	9.2	0.7	
7050	15	189	3.8	9.9	0.8	
7060	15	189	4.28	10.5	0.9	
7070	15	189	4.75	11	1	
7080	15	189	5.23	11.6	1.1	X
7090	15	189	5.7	12.1	1.2	X
7100	15	189	6.65	13.1	1.4	X
7110	15	189	7.6	14	1.6	X

Table 9: Irregular sea states

# wave series	$v_S[kn]$	$\mu[deg]$	$H_s[m]$	$T_p[s]$	cancelled
7700	15	0	1.5	7.5	
7710	15	0	4.5	10.5	
7720	15	0	3.5	10.5	X
7730	15	0	1.5	11.5	X

B Response amplitude results: ShipX

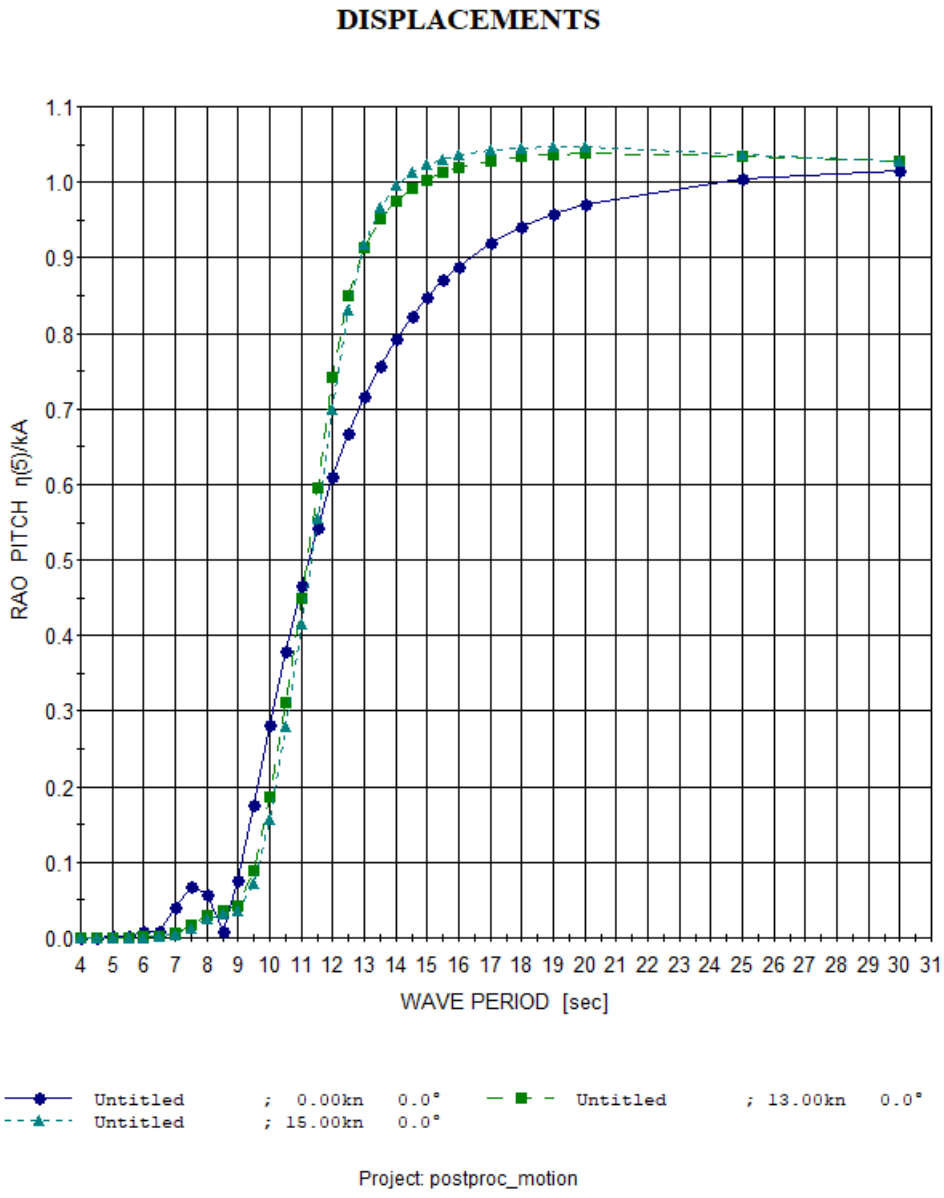


Figure 43: Resulting pitch RAOs from ShipX

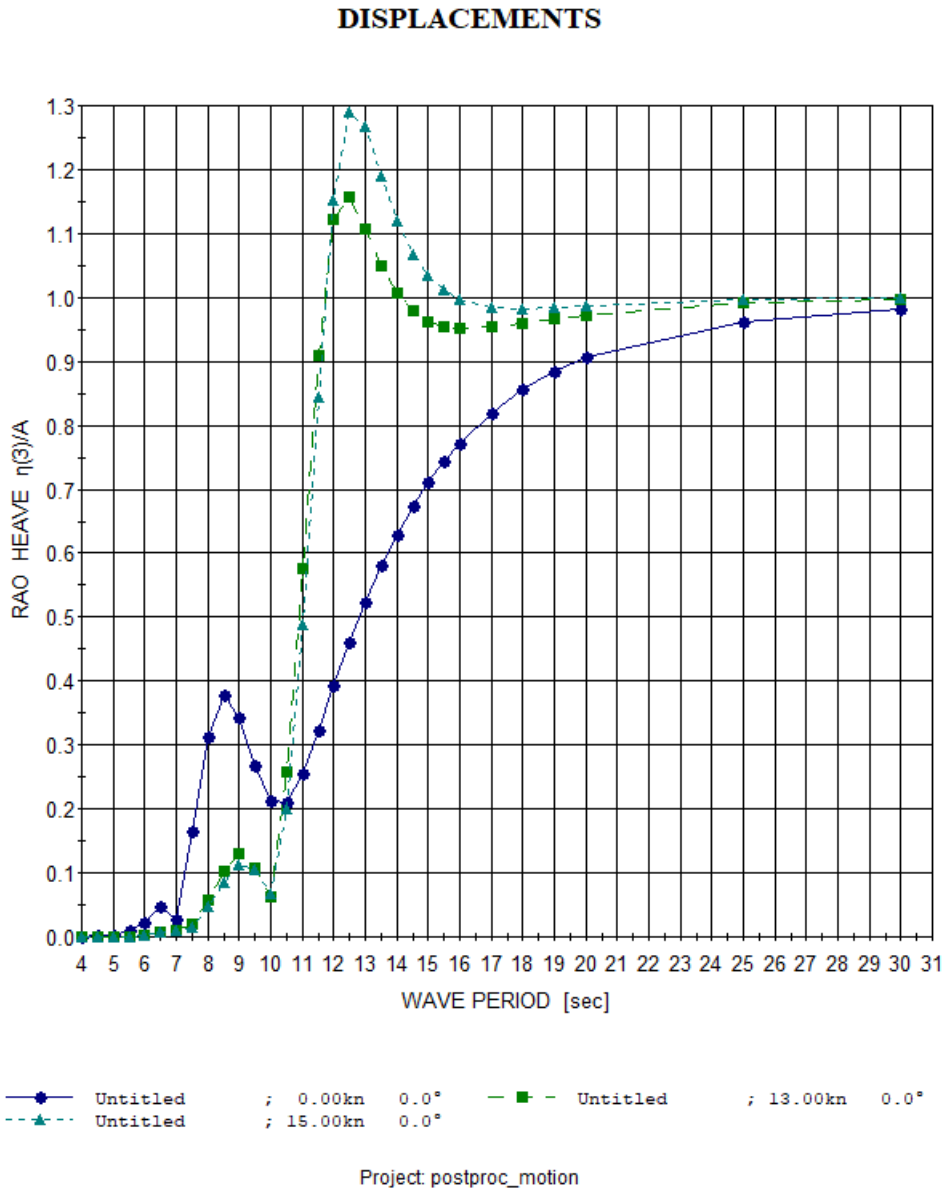


Figure 44: Resulting heave RAOs from ShipX

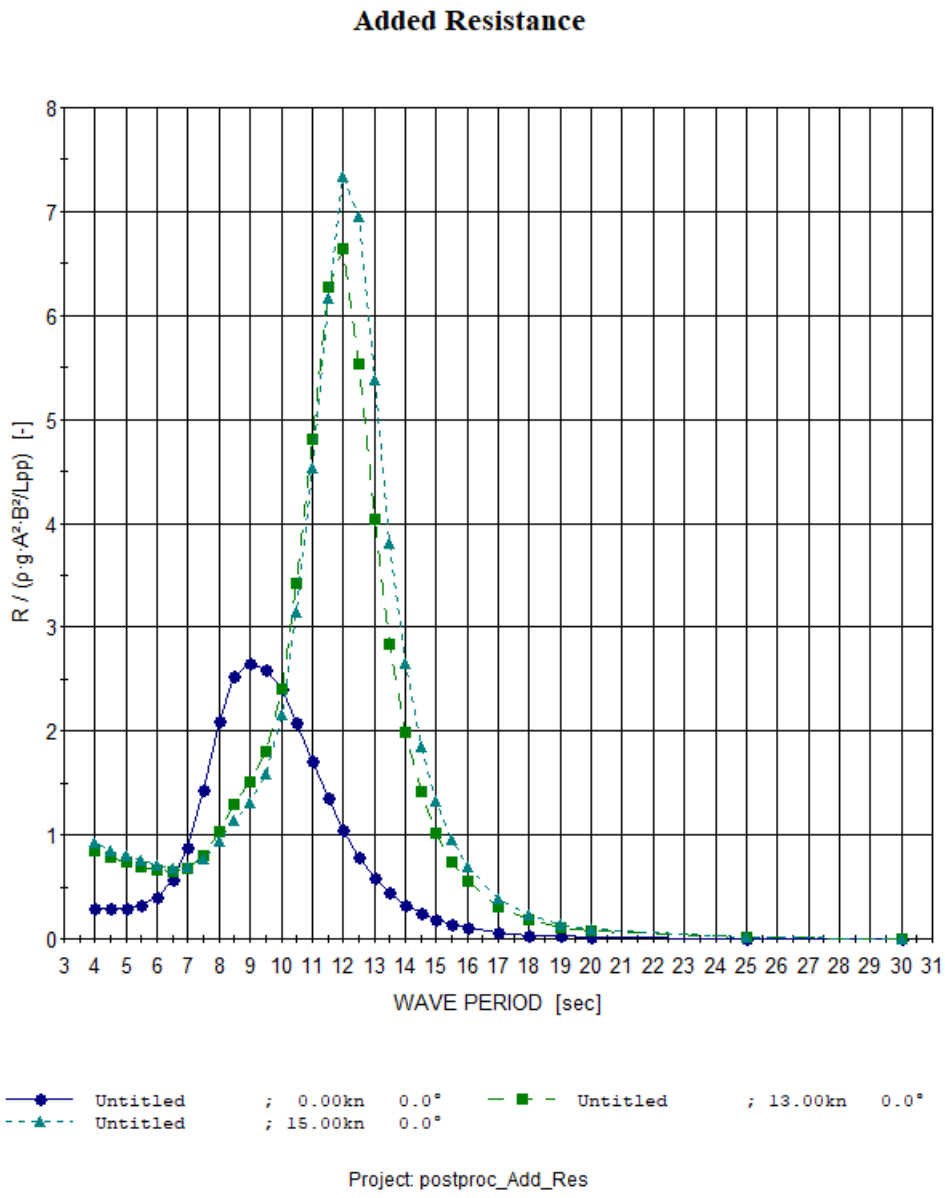


Figure 45: Resulting dimensionless added resistance from ShipX

C Experimental irregular sea state timeseries

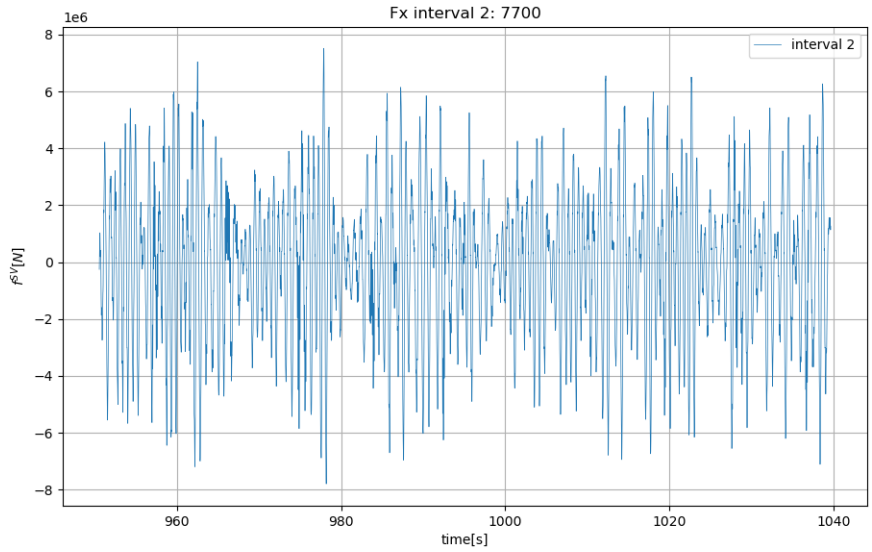


Figure 46: Wave induced force: irregular seaway runs #7700, interval 2

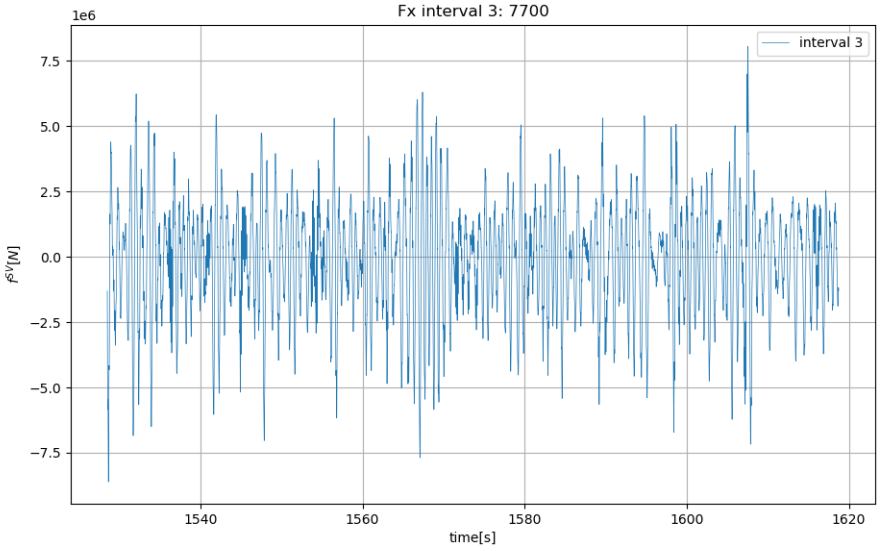


Figure 47: Wave induced force: irregular seaway runs #7700, interval 3

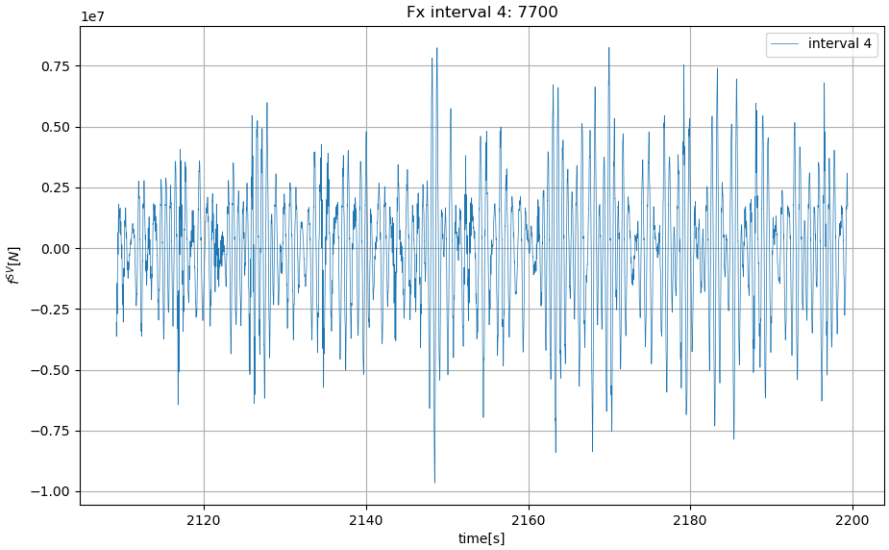


Figure 48: Wave induced force: irregular seaway runs #7700, interval 4

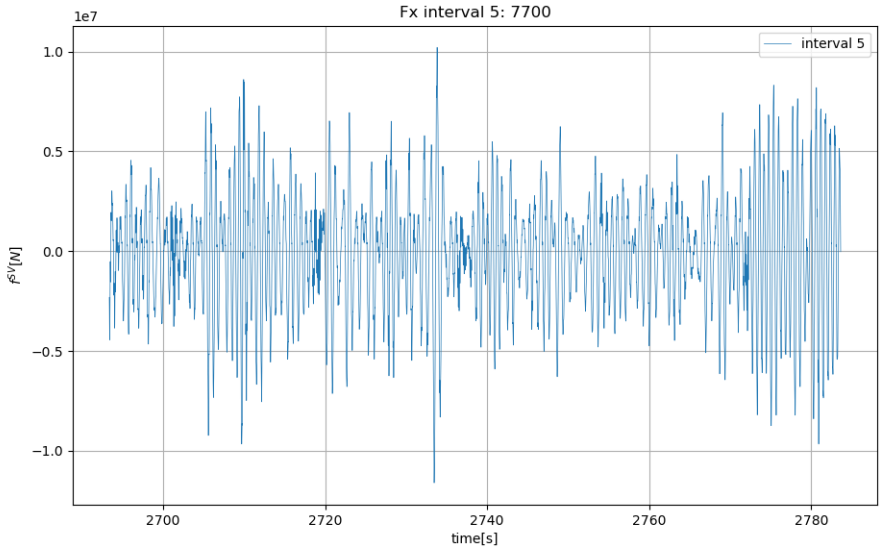


Figure 49: Wave induced force: irregular seaway runs #7700, interval 5

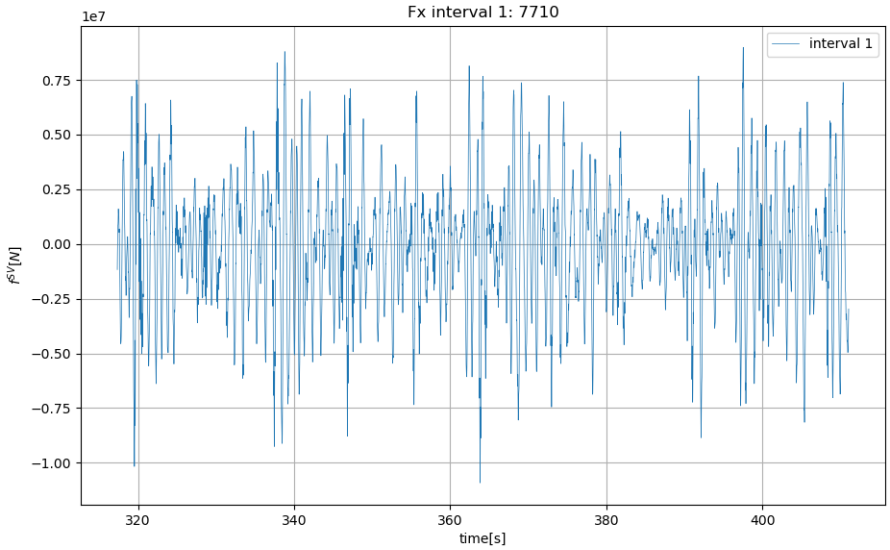


Figure 51: Wave induced force: irregular seaway run #7710, interval 1

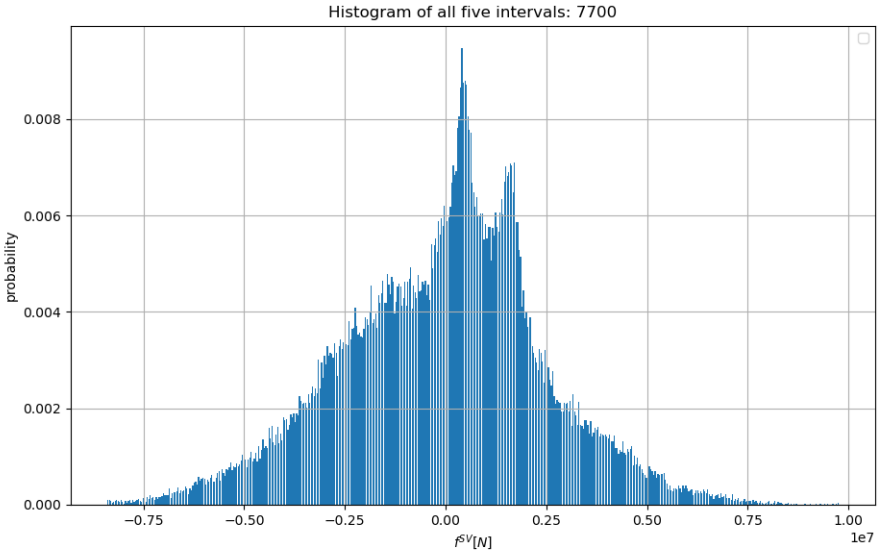


Figure 50: Irregular seaway run #7700, histogram for all five intervals combined

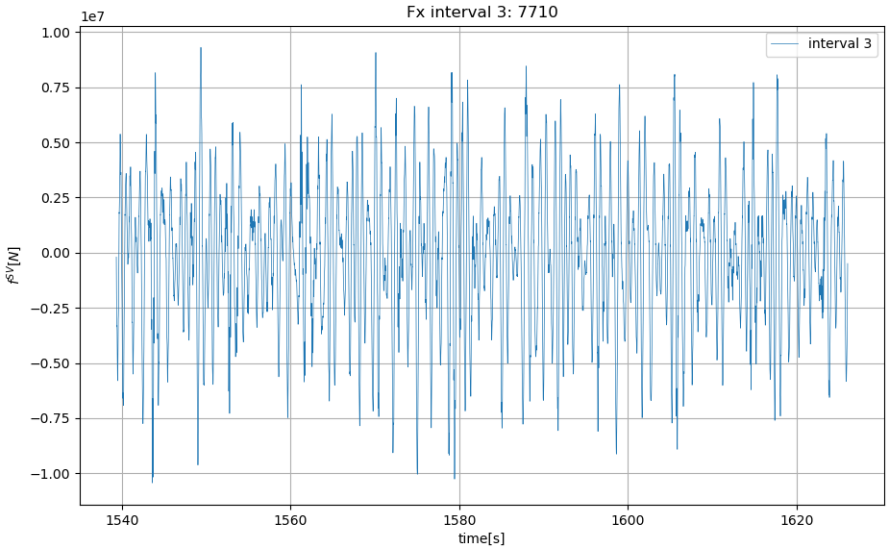


Figure 53: Wave induced force: irregular seaway runs #7710, interval 3

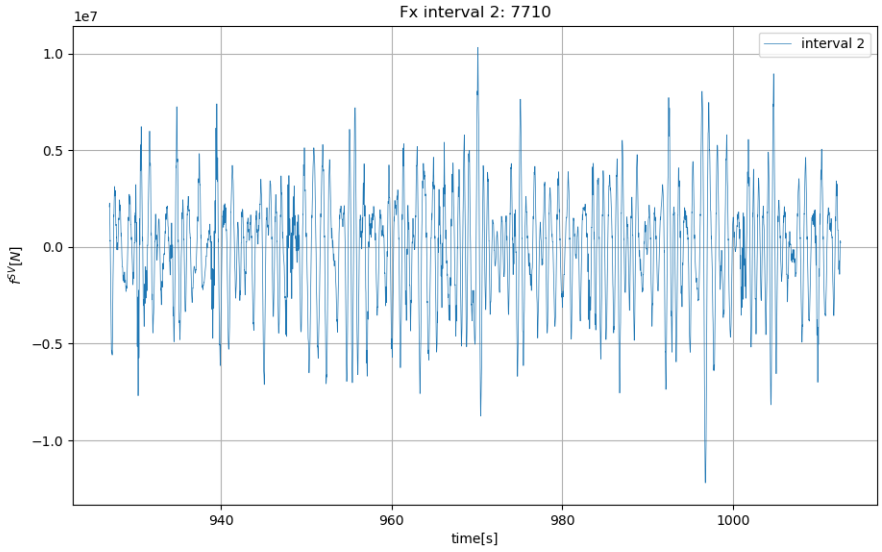


Figure 52: Wave induced force: irregular seaway runs #7710, interval 2

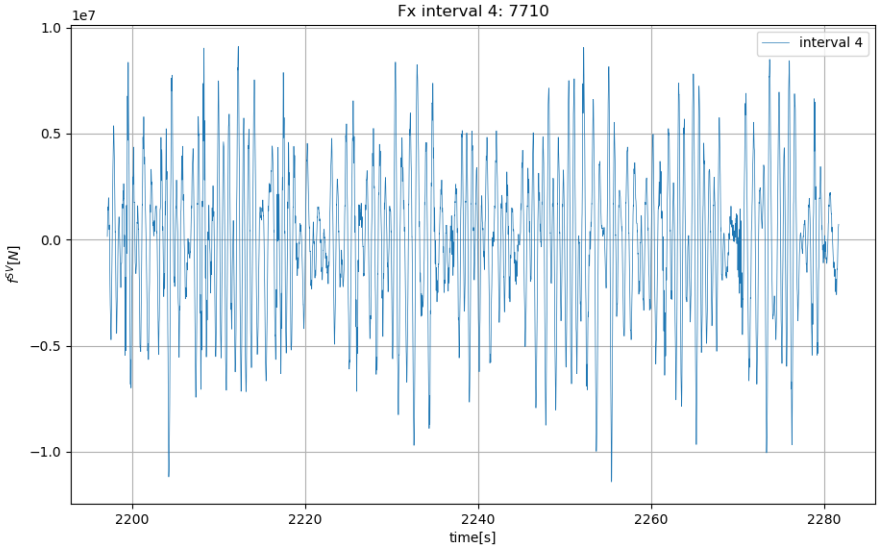


Figure 54: Wave induced force: irregular seaway runs #7710, interval 4

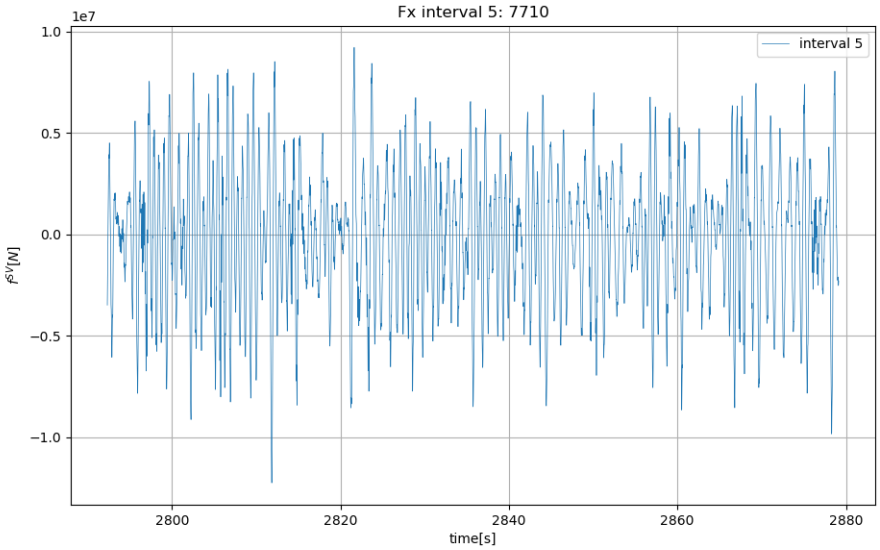


Figure 55: Wave induced force: irregular seaway runs #7710, interval 5

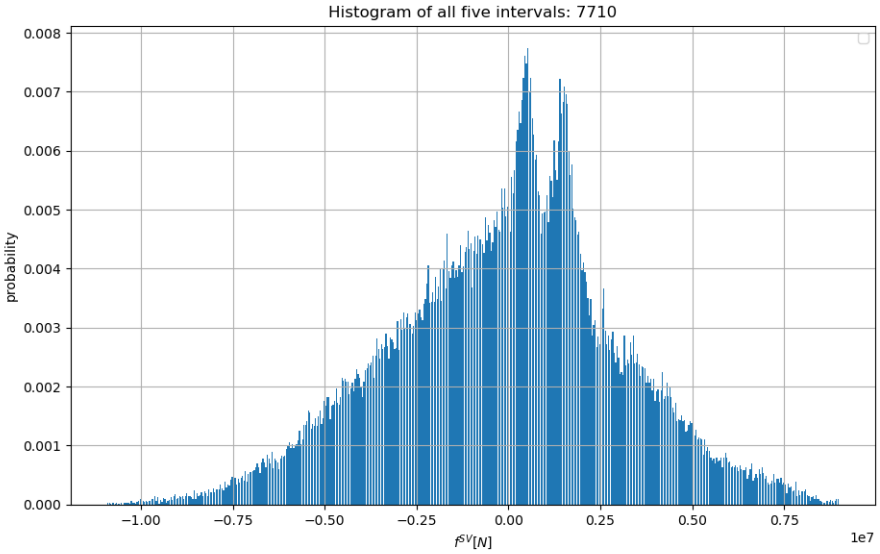


Figure 56: Irregular seaway run #7710, histogram for all five intervals combined

D Python: pre-processing

```
import numpy as np
import os
import matplotlib.pyplot as plt
import scipy.signal as sp
import scipy.fft as sf
import xlsxwriter
from matplotlib.backends.backend_pdf import PdfPages

def OQUS_interval(intStart, intEnd,time_oqus):
    oqus_intStart = round(intStart*5.)/5.       # oqus_intSTart in seconds
    oqus_intEnd = round(intEnd*5.)/5.
    oqus_intStart1 = np.argwhere(time_oqus<oqus_intStart)
    oqus_intStart = oqus_intStart1[-1]
    oqus_intStart = oqus_intStart[0]
    oqus_intEnd1 = np.argwhere(time_oqus>oqus_intEnd)
    oqus_intEnd = oqus_intEnd1[0]
    oqus_intEnd = oqus_intEnd[0]
    return [oqus_intStart, oqus_intEnd]
def ensure_dir(file_path):
    directory = os.path.dirname(file_path)
    if not os.path.exists(directory):
        os.makedirs(directory)
def annot_max(x,y, ax=None):
    xmax = x[np.argmax(y)]
    ymax = y.max()
    text= "x={:.3f}, y={:.3f}".format(xmax, ymax)
    if not ax:
        ax=plt.gca()
    bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
    arrowprops=dict(arrowstyle="->",connectionstyle="angle,angleA=0,angleB=60")
    kw = dict(xycoords='data',textcoords="axes fraction",
              arrowprops=arrowprops, bbox=bbox_props, ha="right", va="top")
    ax.annotate(text, xy=(xmax, ymax), xytext=(0.94,0.96), **kw)
def pqFormel(p,q):
    x1 = -p/2 + np.sqrt(((p**2)/4)-q)
    x2 = -p / 2 - np.sqrt(((p ** 2) / 4) - q)
    return x1,x2
def find_mean_period(array: np.ndarray,time: np.ndarray):
    array_mean = np.mean(array)
    index = np.argwhere(np.diff(np.sign(array-array_mean))>0)
    index = index.transpose()
    mean_period = np.mean(np.diff(time[index]))
    T_stddev = np.std(np.diff(time[index]))
    return mean_period
def find_mean_from_peaks(array: np.ndarray,distance = None,height = None):
    meanArray = np.mean(array)
    array = array-meanArray
```

```
    Fx_interval_amplitude_max, _ = sp.find_peaks(array,distance=distance,   height=(0.0, Non
    Fx_interval_amplitude_min, _ = sp.find_peaks(-array,distance=distance,   height=(0.0, No

    mean = 0.5 * (np.mean(slider_averaging(array[Fx_interval_amplitude_max])) + np.mean(slid
    return mean, Fx_interval_amplitude_min,Fx_interval_amplitude_max

def find_mean_period_BH(array: np.ndarray,time: np.ndarray):
    array_mean =np.mean(array)
    array = array-array_mean
    array = Bingham_filter(array)
    index = np.argwhere(np.diff(np.sign(array))>0)
    index = index.transpose()


    mean_period = np.mean(np.diff(time[index]))
    T_stddev = np.std(np.diff(time[index]))
    return mean_period,index,T_stddev
def Bingham_filter(array: np.ndarray):
    n = len(array)
    for i in range(len(array)-1):
        if i+1 <= n/10:
            array[i] = 0.5*(1-np.cos((10*np.pi*(i+1))/(n)))
        elif i+1>= 9*n/10:
            array[i] = 0.5*(1-np.cos(10*np.pi*((i+1)-(9*n/10))/(n)))
    return array
def FFT(array: np.ndarray,dt):
    array_fft = sf.fft(array)
    array_fft = np.abs(array_fft / len(array_fft))
    n = array_fft.size
    freq_plot = sf.fftfreq(n, dt)
    array_fft = array_fft[:int((len(array_fft) - 1) / 2)]
    freq_plot = freq_plot[:int((len(freq_plot) - 1) / 2)]
    return array_fft,freq_plot
def bpass_gaussian(xsig, dt, flow, fhigh):
    xsig_fft = np.fft.fft(xsig)
    fvec = np.fft.fftfreq(len(xsig), d=dt)
    Nsamples = len(fvec)
    # Create Gaussian window filter:
    M = 50  # Total number of samples in Gaussian ramp function
    gausswin = sp.windows.gaussian(M, std=7)
    lramp = np.array(gausswin[:M // 2])  # Left ramp function (left half of Gaussian)
    rramp = np.array(gausswin[M // 2:])  # Right ramp function (right half of Gaussian)
    i1 = np.argwhere(np.abs(fvec[:Nsamples // 2]) > flow)[0][0]  # Index of low cut frequen
    i2 = np.argwhere(np.abs(fvec[:Nsamples // 2]) < fhigh)[-1][0]  # Index of high cut freq
    lwin = np.zeros(np.max([i1 - M // 2, 0]))  # Left zero-padding
    rwin = np.zeros(np.min([Nsamples // 2 - (i2 + M // 2), Nsamples // 2 - 1]))
# Right zero-padding
    oneband = np.ones(np.max([Nsamples // 2 - len(lwin) - len(rwin) - len(lramp) - len(rram
```

```
# Unity pass-band
    filter = np.concatenate((lwin, lramp, oneband, rramp, rwin))  # Combined window functio
    # Filter signal by multiplication with window function in frequency domain
    # Inverse FFT to obtain filtered time-series
    xsig_bp = np.fft.ifft(xsig_fft * np.concatenate((filter, filter[::-1]))).real
### gives warning
    return xsig_bp, (fvec, filter)
def homemade_Gauss(array: np.ndarray,dt,lcut,hcut ):

    fft = sf.fft(array)
    n = fft.size
    freq = sf.fftfreq(n, dt)
    for i in range(len(fft)):
        if freq[i]<=lcut:
            fft[i] = 0.0
        elif freq[i] >= hcut:
            fft[i] = 0.0
    new_array = sf.ifft(fft)
    return new_array.real
def dispers_eq(x,H):
    return (np.pi*2)/x * np.tanh(2*np.pi*H/x)
def find_wave_length(angl_freq,H,intialguess=0.0):
    xtop = 100000.
    xbot = 0.000001
    x= intialguess#5.9375
    f= lambda x:dispers_eq(x,H)/angl_freq**2 -1
    dfdx = lambda x:-2*9.8066*np.pi/x**(-2)*np.tanh(2*np.pi*H/x) + (np.pi*2*9.8066)/x *(-2*
    while abs(f(x))>1e-4:
        print('x',x)
        x_new=x-f(x)/dfdx(x)
        x=x_new
    return x
def set_interval(array: np.ndarray,intervalstart,intervalend):
    array_mean = np.mean(array)
    index = np.argwhere(np.diff(np.sign(array-array_mean))>0)
    index = index.transpose()
    index= index[0]
    checkstart = abs(index-intervalstart)
    checkend = abs(index-intervalend)
    min = np.argmin(checkstart)
    start = index[min]
    max = np.argmin(checkend)
    end = index[max]

    print('start/end',start,end)
    return [start,end]
def find_wavelength2(angular_wavefreq,H,initguess):
    k=2*np.pi/initguess
    f = lambda k:9.81*k*np.tanh(k*H)-angular_wavefreq*angular_wavefreq
```

```
    dfdx = lambda k:9.81*(np.tanh(k*H)+k/(np.cosh(k*H)**2))
    while abs(f(k))>1e-5:
        k_new = k-f(k)/dfdx(k)
        k=k_new
    return 2*np.pi/k
def slider_averaging(array: np.ndarray):
    return array
def find_amplitude(array: np.ndarray,distance = None):
    array = array- np.mean(array)
    amplitude_max, _ = sp.find_peaks(array,distance=distance,   height=(np.max(array)*0.3, N
    amplitude_min, _ = sp.find_peaks(-array,distance=distance,   height=(np.max(-array)*0.3,
height=(0.5 * np.max(-array), 2 * np.max(-array)))
    amplitude = np.mean(slider_averaging(array[amplitude_max])) - np.mean(slider_averaging(
    return amplitude/2.
def find_calc_interval(carriage_vel: np.ndarray,carriage_pos: np.ndarray):
    carrvel_lp = butter_lowpass_filter(carriage_vel, 1., 200.)
    acc_magnitude = (np.diff(carrvel_lp) / 0.005)
    iacc_peaks, _ = sp.find_peaks(acc_magnitude)
    for i in iacc_peaks:
        index = np.argwhere(iacc_peaks == i)
        if np.abs(acc_magnitude)[i] > 0.04 or np.abs(acc_magnitude)[i] < 0.004:
            iacc_peaks = np.delete(iacc_peaks, index)
        elif carriage_vel[i] < 1.0:
            iacc_peaks = np.delete(iacc_peaks, index)
    for i in range(20):  # how many periods to take away at the beginning and end
        iacc_peaks = np.delete(iacc_peaks, 0)
    peaks_old = np.array([], dtype='int')
    peaks_new = np.array([], dtype='int')
    for i in iacc_peaks:
        if carriage_pos[i] <= 175.:
            peaks_old = np.append(peaks_old, i)
        elif carriage_pos[i] > 175.:
            peaks_new = np.append(peaks_new, i)
    interval = np.array([iacc_peaks[0],iacc_peaks[-1]], dtype='int')
    interval_old = np.array([peaks_old[0],peaks_old[-1]], dtype='int')
    interval_new = np.array([peaks_new[0],peaks_new[-1]], dtype='int')

    return interval,interval_old,interval_new
def butter_lowpass_filter(data, cutoff, fs, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = sp.butter(order, normal_cutoff, btype='low', analog=False)
    y = sp.filtfilt(b, a, data)
    return y
def dimensionless_force(array: np.ndarray,amplitude_wp1,width,Lpp):
    Fx_dimless = np.array([])
    for i in array:
        value = i/(1000*9.81*amplitude_wp1*amplitude_wp1*width*width/Lpp)
        Fx_dimless = np.append(Fx_dimless,value)
```

```
    return Fx_dimless



##automatic plotting
datapath = "./npz2/"
figurepath = "./figure_spectrum/"
testinfolist = []
filelist = os.listdir(datapath)
infiles = [filename for filename in filelist if filename.endswith('.npz')]
Fxamplitudesmultipage = PdfPages(figurepath + '/' + 'Fx_amplitude_multipage.pdf')
heaveamplitudesmultipage = PdfPages(figurepath + '/' + 'heave_amplitude_multipage.pdf')
pitchamplitudesmultipage = PdfPages(figurepath + '/' + 'pitch_amplitude_multipage.pdf')


## open excel worksheet
workbook = xlsxwriter.Workbook('mean_Fx_allruns.xlsx',options={'nan_inf_to_errors': True})
worksheet = workbook.add_worksheet()
worksheet.write(0, 0,'Run')
col_heaveamplitude = 1
worksheet.write(0, col_heaveamplitude,'heave')
col_pitchamplitude = 2
worksheet.write(0, col_pitchamplitude,'pitch')
col_wp1Period = 3
worksheet.write(0, col_wp1Period,'WP1 period')
col_wp3Period = 4
worksheet.write(0, col_wp3Period,'WP3 period')
col_wavelength = 7
worksheet.write(0, col_wavelength,'wavelength')
col_wp1height = 5
worksheet.write(0, col_wp1height,'WP1 height')
col_wp3height = 6
worksheet.write(0, col_wp3height,'WP3 height')
col_heaveRAO = 10
worksheet.write(0, col_heaveRAO,'heave RAO')
col_pitchRAO = 11
worksheet.write(0, col_pitchRAO,'pitch RAO')
col_Fx_mean = 12
worksheet.write(0, col_Fx_mean,'mean Fx')
col_Fx_amplitude = 13
worksheet.write(0, col_Fx_amplitude,'Fx amplitude')
col_Fx_Period = 14
worksheet.write(0, col_Fx_Period,'Fx period')
col_dimlessMeanFx = 15
worksheet.write(0, col_dimlessMeanFx,'dim. mean Fx')
col_encounterPeriod =17
worksheet.write(0, col_encounterPeriod,'encounter period')
col_heavePeriod = 8
```

```
worksheet.write(0, col_heavePeriod,'heave period')
col_pitchPeriod = 9
worksheet.write(0, col_pitchPeriod,'pitch period')
col_dimlessFxAmplitude = 16
worksheet.write(0, col_dimlessFxAmplitude,'dim. Fx ampl.')



## vessel data
waterdepth = 5.8 # 8.5
Lpp = 5.9375
width = 1.009375


##plot all plots
for index_infile in range(len(infiles)):
    infile = infiles[index_infile]


    df = np.load(datapath + infile, allow_pickle=True)
    data = df['data']
    keys = df['keys']
    data_array = data.item()
    datadict = {key: value for key, value in zip(keys, data_array)}
    time = data_array[datadict['DAQ time stamps']]
    time2 = data_array[datadict['DAQ time stamps SR2']]
    FX = data_array[datadict['FX_CENTER']]
    wp1 = data_array[datadict['WP1_BM']]
    wp3 = data_array[datadict['WP3_CAR']]
    wp1_tot = data_array[datadict['WP1_BM']]
    wp3_tot = data_array[datadict['WP3_CAR']]
    wp2_tot = data_array[datadict['WP2_BM']]
    wp4_tot = data_array[datadict['WP4_CAR']]
    wpUS_tot = data_array[datadict['WP_US']]

    ## filter
    # filter carriage speed
    t = time
    dt = 0.005

    ## find calculation interval
    carriage_vel = data_array[datadict['speed_carriage']]
    carriage_pos = data_array[datadict['pos_carriage_analog']]
    interval, interval_old, interval_new = find_calc_interval(carriage_vel, carriage_pos)

    ##compute time period between accelerations
    interval_start = interval[0]
    interval_end = interval[-1]
    interval_length = interval_end - interval_start
```

```
plot_interval = []
for i in range(interval_length):
        plot_interval.append(i)
plot_interval = plot_interval + (interval_start)


## set new intervals for Fx
Fx_interval = FX[interval_start:interval_end]
if len(Fx_interval) % 2 > 0:
    Fx_interval = np.delete(Fx_interval, -1)
time_interval = time[interval_start:interval_end]
if len(time_interval) % 2 > 0:
    time_interval = np.delete(time_interval, -1)
carriage_vel_interval = carriage_vel[interval_start:interval_end]
if len(carriage_vel_interval) % 2 > 0:
    carriage_vel_interval = np.delete(carriage_vel_interval, -1)
wp1 = wp1[interval_start:interval_end]
if len(wp1) % 2 > 0:
    wp1 = np.delete(wp1, -1)
wp3 = wp3[interval_start:interval_end]
if len(wp3) % 2 > 0:
    wp3 = np.delete(wp3, -1)
wp2 = wp2_tot[interval_start:interval_end]
if len(wp2) % 2 > 0:
    wp2 = np.delete(wp2, -1)
wp4 = wp4_tot[interval_start:interval_end]
if len(wp4) % 2 > 0:
    wp4 = np.delete(wp4, -1)
wpUS = wpUS_tot[interval_start:interval_end]
if len(wpUS) % 2 > 0:
        wpUS = np.delete(wpUS, -1)
## calculate encounter frequency
mean_period_wp1 = find_mean_period(wp1,time_interval)
mean_period_wp3 = find_mean_period(wp3, time_interval)

mean_vel = np.mean(carriage_vel_interval)
frequency_wp1 = 2. * np.pi / mean_period_wp1
frequency_wp3 = 2. * np.pi / mean_period_wp3
enc_freq_omega = frequency_wp1 + mean_vel / 9.81 * frequency_wp1 ** 2
enc_freq = 0.5 * enc_freq_omega / np.pi

## gaussian bandpass filter
lowcut = enc_freq * 0.9  # 0.08#*2*np.pi#
highcut = enc_freq * 1.1  # 0.0105#*2*np.pi#
Fx_bp = homemade_Gauss(Fx_interval,dt,lowcut,highcut)


## check interval for main frequency
mean_period_Fx = find_mean_period(Fx_bp, time_interval)
```

```
## lowpass filter
Fx_lp_interval = butter_lowpass_filter(Fx_interval,mean_period_Fx*4.,200.)
wp1 = butter_lowpass_filter(wp1,mean_period_wp1*1.1,200.)

## Fast Fourier Transformation
Fx_fft, Fx_freq = FFT(Fx_interval,dt)

## find amplitude and mean value
amplitude_Fx = find_amplitude(Fx_interval,distance=(200.*0.8*mean_period_Fx))
amplitude_wp1 = find_amplitude(wp1, distance=mean_period_wp1*200.*0.8)
amplitude_wp3 = find_amplitude(wp3,distance=mean_period_wp3*200*0.8)

Fx_mean_FA,_,_ = find_mean_from_peaks(Fx_interval)
Fx_lp_mean_FA,Fx_interval_min,Fx_interval_max = find_mean_from_peaks(Fx_lp_interval, di

Fx_amplitude_max, _ = sp.find_peaks(Fx_lp_interval, distance=(200.*0.8*mean_period_Fx),
Fx_amplitude_min, _ = sp.find_peaks(-Fx_lp_interval, distance=(200.*0.8*mean_period_Fx)



## find wavelength
wavelength = find_wavelength2(frequency_wp1,waterdepth,Lpp)
wp3_wavelength = find_wavelength2(frequency_wp3,waterdepth,Lpp)


## added resistance dimensionless
Fx_interval_mean,_,_ = find_mean_from_peaks(Fx_interval,distance=(200*0.8*mean_period_F
Fx_dimless = dimensionless_force(Fx_interval,amplitude_wp1,width,Lpp)
Fx_dimless_fft, Fx_dimless_freq =FFT(Fx_dimless,dt)
Fx_dimless_mean,_,_ = find_mean_from_peaks(Fx_dimless,distance=(200*0.8*mean_period_Fx)
Fx_dimless_amplitude = find_amplitude(Fx_dimless,distance=(200*0.8*mean_period_Fx))
Fx_dimless_mean_FA,_,_ = find_mean_from_peaks(Fx_dimless)
Fx_amplitude = find_amplitude(Fx_interval,distance=(200*0.8*mean_period_Fx))
Fx_lp_dimless = dimensionless_force(Fx_lp_interval, amplitude_wp1, width, Lpp)
Fx_lp_dimless_mean,_,_ = find_mean_from_peaks(Fx_lp_dimless,distance=(200*0.8*mean_perio
Fx_lp_dimless_mean_FA,_,_ = find_mean_from_peaks(Fx_lp_dimless)



## OQUS values

time_oqus_total = data_array[datadict['DAQ_time_stamps_SR2']]
zpos_total = data_array[datadict['zpos1']]
pitch = data_array[datadict['pitch1']]
roll = data_array[datadict['roll1']]

oqus_intStart = time_interval[0]
oqus_intStart = round(oqus_intStart * 5.) / 5.
```

```
oqus_intEnd = time_interval[-1]
oqus_intEnd = round(oqus_intEnd * 5.) / 5.
oqus_intStart1 = np.argwhere(time_oqus_total < oqus_intStart)
oqus_intStart = oqus_intStart1[-1]
oqus_intStart = oqus_intStart[0]
oqus_intEnd1 = np.argwhere(time_oqus_total > oqus_intEnd)
oqus_intEnd = oqus_intEnd1[0]
oqus_intEnd = oqus_intEnd[0]


time_oqus_old_vec = OQUS_interval(time[interval_old[0]], time[interval_old[-1]],time_oq

time_oqus = time_oqus_total[oqus_intStart:oqus_intEnd]
if len(time_oqus) % 2 > 0:
    time_oqus = np.delete(time_oqus, -1)
zpos = zpos_total[oqus_intStart:oqus_intEnd]
if len(zpos) % 2 > 0:
    zpos = np.delete(zpos, -1)
roll = roll[oqus_intStart:oqus_intEnd]
if len(roll) % 2 > 0:
    roll = np.delete(roll, -1)
pitch = pitch[oqus_intStart:oqus_intEnd]
if len(pitch) % 2 > 0:
    pitch = np.delete(pitch, -1)

time_oqus_old = time_oqus_total[time_oqus_old_vec[0]:time_oqus_old_vec[-1]]
if len(time_oqus_old) % 2 > 0:
    time_oqus_old = np.delete(time_oqus_old, -1)
zpos_old = zpos_total[time_oqus_old_vec[0]:time_oqus_old_vec[-1]]
if len(zpos_old) % 2 > 0:
    zpos_old = np.delete(zpos_old, -1)


zpos_period = find_mean_period(zpos, time_oqus)
zpos1 = zpos - np.mean(zpos)
zpos_amplitude = find_amplitude(zpos1,distance=(50.*0.8*zpos_period))
zpos_dimless_amplitude = zpos_amplitude/amplitude_wp3

zpos_old_period = find_mean_period(zpos_old, time_oqus_old)
zpos1 = zpos - np.mean(zpos)
zpos_old_amplitude = find_amplitude(zpos_old, distance=(50. * 0.8 * zpos_old_period))
zpos_old_dimless_amplitude = zpos_old_amplitude / amplitude_wp3

pitch_period = find_mean_period(pitch, time_oqus)
pitch_lp = butter_lowpass_filter(pitch,pitch_period*1.1,50.)
pitch1 = pitch - np.mean(pitch)
pitch_amplitude = find_amplitude(pitch1,distance=(50.*0.8*pitch_period))

zpos_lp = butter_lowpass_filter(zpos_old,zpos_old_period*1.1,50.)
```

```
zpos_lp_amplitude = find_amplitude(zpos_lp,distance=(50.*0.8*zpos_period))

heave_amplitude_max, _ = sp.find_peaks(zpos-np.mean(zpos), distance=(50.*0.8*zpos_perio
heave_amplitude_min, _ = sp.find_peaks(-zpos+np.mean(zpos), distance=(50.*0.8*zpos_peri
heave_old_amplitude_max, _ = sp.find_peaks(zpos_old - np.mean(zpos_old), distance=(50.
heave_old_amplitude_min, _ = sp.find_peaks(-zpos_old + np.mean(zpos_old), distance=(50.

pitch_amplitude_max, _ = sp.find_peaks(pitch-np.mean(pitch), distance=(50.*0.8*pitch_pe
pitch_amplitude_min, _ = sp.find_peaks(-pitch+np.mean(pitch), distance=(50.*0.8*pitch_p



## write into excel sheet
string_file = str(infile)
string_file = string_file.translate({ord(i): None for i in 'E.npz '})

row = index_infile+1

worksheet.write(row,0,int(string_file))
worksheet.write(row, col_heaveamplitude, 2.*zpos_lp_amplitude)
worksheet.write(row, col_pitchamplitude, 2.*pitch_amplitude)
worksheet.write(row, col_wp1Period, mean_period_wp1)
worksheet.write(row, col_wp3Period, mean_period_wp3)
worksheet.write(row, col_wavelength, wavelength)
worksheet.write(row, col_wp1height, 2*amplitude_wp1)
worksheet.write(row, col_wp3height, 2*amplitude_wp3)
worksheet.write(row, col_heaveRAO, zpos_lp_amplitude/amplitude_wp3)
worksheet.write(row, col_pitchRAO, pitch_amplitude*wp3_wavelength/(amplitude_wp3*180.))
worksheet.write(row,col_Fx_mean,Fx_lp_mean_FA*-1)
worksheet.write(row, col_Fx_amplitude, amplitude_Fx)
worksheet.write(row, col_Fx_Period, mean_period_Fx)
worksheet.write(row, col_dimlessMeanFx, -1*Fx_mean_FA*Lpp/(1000*9.81*4*amplitude_wp3*am
worksheet.write(row,col_encounterPeriod,1./enc_freq)
worksheet.write(row, col_heavePeriod, zpos_old_period)
worksheet.write(row, col_pitchPeriod, pitch_period)
worksheet.write(row, col_dimlessFxAmplitude, Fx_amplitude*Lpp/(1000*9.81*4*amplitude_wp


## plot graphs and save pdf

string_key = 'Fx spectrum'
plot_title = string_file + '_' + string_key
carr_plot_title = string_file + '_Carriage'
Fx_plot_title = string_file + '_Fx timeseries'
Fx_freq_plot_title = string_file + '_Fx relative frequency'
Fx_dimless_plot_title = string_file + '_Fx dimensionless'
ensure_dir(figurepath + string_file + '/')
pp = PdfPages(figurepath + string_file + '/' +'multipage.pdf')
```

```
plt.figure(1)
plt.grid()
plt.plot(time, FX, linewidth=0.7)
plt.title(Fx_plot_title)
plt.xlabel('time ')
plt.ylabel('Fx')
plt.axvline(time[interval[0]], c='red')
plt.axvline(time[interval[1]], c='red')
plt.savefig(figurepath + string_file + '/' + Fx_plot_title + '.pdf')
plt.savefig(pp, format='pdf')
plt.close(1)

plt.figure('Fx interval')
plt.title('Fx timeseries: time interval')
plt.xlabel('time[s]')
plt.ylabel('Fx[N]')
plt.plot(time_interval, Fx_interval, linewidth=0.7, label='w/o')
plt.plot(time_interval, Fx_lp_interval, linewidth=0.7, label='lowpass')
plt.plot(time_interval[Fx_interval_max], Fx_lp_interval[Fx_interval_max], 'o', linewidth
plt.plot(time_interval[Fx_interval_min], Fx_lp_interval[Fx_interval_min], 'o', linewidth
plt.legend(loc='best')
plt.savefig(figurepath + string_file + '/' + 'Fx_interval' + '.pdf')
plt.savefig(pp, format='pdf')
plt.close('Fx interval')

plt.figure(2)
plt.grid()
plt.plot(Fx_freq, Fx_fft, linewidth=0.7)
plt.title(plot_title)
print(plot_title)
plt.xlabel('frequency')
plt.ylabel('spectrum')
plt.xlim([-0.005, 5])
annot_max(Fx_freq, Fx_fft)
plt.savefig(figurepath + string_file + '/' + plot_title + '.pdf')
plt.savefig(pp, format='pdf')
plt.close(2)

plt.figure(3)
plt.grid()
plt.plot(Fx_freq/enc_freq, Fx_fft, linewidth=0.7)
plt.title(Fx_freq_plot_title)
plt.xlabel(r'$\omega / \omega_e$')
plt.ylabel('spectrum')
plt.xlim([-0.005, 5])
plt.savefig(figurepath + string_file + '/' + Fx_freq_plot_title + '.pdf')
plt.savefig(pp, format='pdf')
plt.close(3)
```

```
plt.figure(5)
plt.grid()
plt.plot(Fx_dimless_freq, Fx_dimless_fft, linewidth=0.7)
plt.title(Fx_dimless_plot_title)
plt.xlabel('frequency')
plt.ylabel(r'$\frac{R_{AW}}{\rho g \zeta^{2}B^2/L_{PP}}$')
plt.xlim([-0.005, 5])
plt.savefig(figurepath + string_file + '/' + Fx_dimless_plot_title + '.pdf')
plt.savefig(pp, format='pdf')
plt.close(5)

plt.figure(5)
plt.grid()
plt.plot(time_interval, wp3, linewidth=0.7, label='WP3')
plt.plot(time_interval, wp4, linewidth=0.7, label='WP4')
plt.plot(time_interval, wpUS, linewidth=0.7, label='WP US')
plt.title('wave probes: carriage')
plt.xlabel('time[s]')
plt.ylabel('wave amplitude[m]')
plt.legend(loc='best')
plt.savefig(figurepath + string_file + '/' + 'WP_car.pdf')
plt.close(5)

plt.figure(5)
plt.grid()
plt.plot(time_interval, wp1, linewidth=0.7, label='WP1')
plt.plot(time_interval, wp2, linewidth=0.7, label='WP2')
plt.title('wave probes: wave maker' )
plt.xlabel('time[s]')
plt.ylabel('wave amplitude[m]')
plt.legend(loc='best')
plt.savefig(figurepath + string_file + '/' + 'WP_wm.pdf')
plt.close(5)

fig = plt.figure('WP1')
plt.suptitle('Wave probe 1')
ax1 = fig.add_subplot(2, 1, 1)
ax1.plot(time, wp1_tot, '-', label='WP1')
ax1.axvline(time[interval[0]], c='red')
ax1.axvline(time[interval[1]], c='red')
ax1.set_ylabel('wave amplitude[m]')
ax2 = fig.add_subplot(2, 1, 2)
ax2.plot(time_interval, wp1, '-', label='WP1 interval')
ax2.set_xlabel('time[s]')
ax2.set_ylabel('wave amplitude[m]')
ax1.grid()
ax2.grid()
plt.legend(loc='best')
```

```
plt.savefig(pp, format='pdf')
plt.savefig(figurepath + string_file + '/' +  'WP1_tot.pdf')
plt.close(fig)

fig = plt.figure('WP2')
plt.suptitle('Wave probe 2')
ax1 = fig.add_subplot(2, 1, 1)
ax1.plot(time, wp2_tot, '-', label='WP2')
ax1.axvline(time[interval[0]], c='red')
ax1.axvline(time[interval[1]], c='red')
ax1.set_ylabel('wave amplitude[m]')
ax2 = fig.add_subplot(2, 1, 2)
ax2.plot(time_interval, wp2, '-', label='WP2 interval')
ax2.set_xlabel('time[s]')
plt.legend(loc='best')
ax2.set_ylabel('wave amplitude[m]')
ax1.grid()
ax2.grid()
plt.savefig(pp, format='pdf')
plt.savefig(figurepath + string_file + '/' +'WP2_tot.pdf')
plt.close(fig)


fig = plt.figure('WP3')
plt.suptitle('Wave probe 3')
ax1 = fig.add_subplot(2, 1, 1)
ax1.plot(time, wp3_tot, '-', label='WP3')
ax1.axvline(time[interval[0]], c='red')
ax1.axvline(time[interval[1]], c='red')
ax1.set_ylabel('wave amplitude[m]')
ax2 = fig.add_subplot(2, 1, 2)
ax2.plot(time_interval, wp3, '-', label='WP3 interval')
ax2.set_xlabel('time[s]')
ax2.set_ylabel('wave amplitude[m]')
plt.legend(loc='best')
ax1.grid()
ax2.grid()
plt.savefig(pp, format='pdf')
plt.savefig(figurepath + string_file + '/' +  'WP3_tot.pdf')
plt.close(fig)

fig = plt.figure('WP4')
plt.suptitle('Wave probe 4')
ax1 = fig.add_subplot(2, 1, 1)
ax1.plot(time, wp4_tot, '-', label='WP4')
ax1.axvline(time[interval[0]], c='red')
ax1.axvline(time[interval[1]], c='red')
ax1.set_ylabel('wave amplitude[m]')
ax2 = fig.add_subplot(2, 1, 2)
```

```
ax2.plot(time_interval, wp4, '-', label='WP4 interval')
ax2.set_xlabel('time[s]')
ax2.set_ylabel('wave amplitude[m]')
plt.legend(loc='best')
ax1.grid()
ax2.grid()
plt.savefig(pp, format='pdf')
plt.savefig(figurepath + string_file + '/' +  'WP4_tot.pdf')
plt.close(fig)


plt.figure(5)
plt.grid()
plt.title('Fx_amplitudes '+ string_file)
plt.plot(time_interval, Fx_interval, label='w/o')
plt.plot(time_interval[Fx_amplitude_max], Fx_interval[Fx_amplitude_max], 'o', linewidth
plt.plot(time_interval[Fx_amplitude_min], Fx_interval[Fx_amplitude_min], 'o', linewidth
plt.axhline(np.mean(Fx_interval[Fx_amplitude_max]), c='blue', linestyle='-', linewidth=
plt.axhline(np.mean(Fx_interval[Fx_amplitude_min]), c='blue', linestyle='-', linewidth=
plt.xlabel('time')
plt.ylabel('Fx')
plt.savefig(Fxamplitudesmultipage, format='pdf')
plt.close(5)

plt.figure(6)
plt.grid()
plt.title('heave_amplitudes old tank part '+ string_file)
plt.plot(time_oqus_old, zpos_old, label='w/o')
plt.plot(time_oqus_old[heave_old_amplitude_max], zpos_old[heave_old_amplitude_max], 'o'
plt.plot(time_oqus_old[heave_old_amplitude_min], zpos_old[heave_old_amplitude_min], 'o'
plt.axhline(np.mean(zpos_old[heave_old_amplitude_max]), c='blue', linestyle='-', linewi
plt.axhline(np.mean(zpos_old[heave_old_amplitude_min]), c='blue', linestyle='-', linewi
plt.xlabel('time')
plt.ylabel('heave')
plt.savefig(heaveamplitudesmultipage, format='pdf')
plt.close(6)

plt.figure(7)
plt.grid()
plt.title('pitch_amplitudes '+ string_file)
plt.plot(time_oqus, pitch, label='w/o')
plt.plot(time_oqus[pitch_amplitude_max], pitch[pitch_amplitude_max], 'o', linewidth=0.5
plt.plot(time_oqus[pitch_amplitude_min], pitch[pitch_amplitude_min], 'o', linewidth=0.5
plt.axhline(np.mean(pitch[pitch_amplitude_max]), c='blue', linestyle='-', linewidth=0.5
plt.axhline(np.mean(pitch[pitch_amplitude_min]), c='blue', linestyle='-', linewidth=0.5
plt.xlabel('time')
plt.ylabel('pitch')
plt.savefig(pitchamplitudesmultipage, format='pdf')
plt.close(7)
```

```
        pp.close()
workbook.close()
Fxamplitudesmultipage.close()
heaveamplitudesmultipage.close()
pitchamplitudesmultipage.close()
```

E Python: holtrop

```
import math
import numpy as np


def holtrop(L_WL,B,T_F,T_A,D,Vol,LCB,C_B,C_M,A_V,A_T,A_BT,h_B,Fn):

    """ INPUT """
    rho = 1025.  # kg/m
    rho_air = 1.225  # kg/m
    g = 9.8066  # m/s
    v = Fn*math.sqrt(g*L_WL)      # design speed
    nu = 10. ** (-6.)  # dynamic viscosity water

    # Fn = v / math.sqrt(g * L_WL)
    Re = (v * L_WL) / nu
    T = 0.5 * (T_F + T_A)
    V_shape = False
    U_shape = False
    C_DA = 0.8  # default

    pram_with_gondola = False

    container_type = False

    # appendages
    rudder_behind_skeg = False  # 0.2-0.5
    rudder_behind_skeg_value = 0.3
    rudder_behind_skeg_S = 0.

    rudder_behind_stern = False
    rudder_behind_stern_S = 0.5
    rudder_behind_stern_value = 0.

    twin_screw_rudder_slender = False
    twin_screw_rudder_slender_S = 1.5
    twin_screw_rudder_slender_value = 0.

    twin_screw_rudder_thick = False
    twin_screw_rudder_thick_value = 2.5
    twin_screw_rudder_thick_S = 0.

    shaft_brackets = False  # 2-4
```

```
shaft_brackets_value = 3.0
shaft_brackets_S = 0.

skeg = False  # 0.5-1
skeg_value = 0.75
skeg_S = 0.

strut_bossing = False  # 2-3
strut_bossing_value = 2.5
strut_bossing_S = 0.

hull_bossing = False
hull_bossing_value = 1.0
hull_bossing_S = 0.

exposed_shafts = False  # buttock 10deg =1 / buttock 20deg = 4
exposed_shafts_value = 1.
exposed_shafts_S = 0.

stabilizer_fins = False
stabilizer_fins_value = 1.8
stabilizer_fins_S = 0.

dome = False
dome_value = 1.7
dome_S = 0.

bilge_keels = False
bilge_keels_value = 0.4
bilge_keels_S = 0.

# bow thrust drag coefficient
C_DTH = 0.003  # 0.003 - 0.012
# bow thrust diameter
d_TH = 2.
# numer of bow thrusts
n_TH = 0.

# surface roughness
k_S = 150.  # micro meter

# longitudinal center of buoyancy
lcb = -LCB / L_WL * 100.

# prismatic coefficient
# C_P = Vol/(L_WL*A_M)
C_P = C_B / C_M

# midship coefficient
```

```
# C_M = 1./(1.+(1.-C_B)**3.5)

# length of run
L_R = L_WL * ((1. - C_P + 0.06 * C_P * lcb) / (4. * C_P - 1.))

# water plane area coefficient
if container_type == True:
      if C_P < 0.62 and C_P > 0.57:
          C_WP = 3.226*(C_P-0.36)
else:
      C_WP = 0.763*(C_P+0.34)

# aftbodyform
if V_shape == True:
    C_stern = -10.
elif U_shape == True:
    C_stern = 10.
elif pram_with_gondola == True:
    C_stern = -25.
else:
    C_stern = 0.

# wetted surface hull
c_23 = 0.453 + 0.4425 * C_B - 0.2862 * C_M - 0.003467 * (B/T) + 0.3696 * C_WP
S = L_WL * (2.*T+B) * math.sqrt(C_M) * (0.615989*c_23 + 0.111439*C_M**3. + 0.000571111

# waterline etrance angle
a = -1. * ((L_WL/B)**0.80856 * (1. - C_WP)**0.30484 * (1. - C_P - 0.0225 * lcb)**0.6367
i_E = 1. + 89. * math.exp(a)

# coefficients

if (T / L_WL) > 0.05:
    c_12 = (T / L_WL) ** 0.2228446
elif (T / L_WL) < 0.05 and (T / L_WL) > 0.02:
    c_12 = 48.2 * (T / L_WL - 0.02) ** 2.078 + 0.479948
elif (T / L_WL) < 0.02:
    c_12 = 0.479948

c_13 = 1. + 0.003 * C_stern

c_14 = 1. + 0.011*C_stern

c_23 = 0.453 + 0.4425 * C_B - 0.2862 * C_M - 0.003467 * T / B + 0.3696 * C_WP

if B/L_WL <= 0.11:
    c_7 = 0.229577*(B/L_WL)**(1./3.)

elif B/L_WL > 0.11 and B/L_WL <= 0.25:
```

```
        c_7 = B/L_WL

    elif B / L_WL > 0.25:
        c_7 = 0.5 - 0.0625*(L_WL/B)

    c_3 = 0.56 * (A_BT**1.5) / (B * T * (0.31 * math.sqrt(A_BT) + T_F - h_B))

    c_1 = 2223105. * c_7**(3.78613) * (T / B)**(1.07961) * (90. - i_E)**(-1.37565)

    c_2 = math.exp(-1.89 * math.sqrt(c_3))

    c_5 = 1. - 0.8 * (A_T/(B * T * C_M))

    c_17 = 6919.3 * C_M**(-1.3346) * (Vol/(L_WL**3.))**2.00977 * (L_WL/B - 2.0)**1.40692

    if C_P <= 0.8:
        c_16 = 8.07981 * C_P - 13.8673 * C_P**2. + 6.984388 * C_P**3.
    else:
        c_16 = 1.73014 - 0.7067 * C_P

    if (L_WL**(3.))/Vol <= 512.:
        c_15 = -1.69385
    elif (L_WL**(3.))/Vol > 512. and (L_WL**(3.))/Vol <= 1726.91:
        c_15 = -1.69385 + (L_WL/(Vol**(3.)) - 8.) / 2.36
    else:
        c_15 = 0.0

    # form factor of hull prediction
    # K_1 = c_13 * (0.93 + c_12 * (B/L_R)**0.92497) * (0.95-C_P)**(-0.521448) * (1 -C_P + 0
old
    K_1 = 0.93 + 0.487118 * c_14 * (B/L_WL)**1.06806 * (T/L_WL)**0.46106 * (L_WL/L_R)**0.12
    #print(((L_WL)/Vol))
    # frictional resistance
    # frictional coefficient
    C_F = 0.075/((math.log10(Re) - 2.)**2.)

    R_f = 0.5 * rho * v**2. * S * C_F

    # bow thrust resistance
    sum_bowthruster_resistance = rho * v ** 2. * math.pi * d_TH ** 2. * C_DTH * n_TH

    # appendage resistance

    appendage_list = [rudder_behind_skeg, rudder_behind_stern, twin_screw_rudder_slender, t
                      shaft_brackets, skeg, strut_bossing, hull_bossing, exposed_shafts, st
                      bilge_keels]
    appendage_matrix = [appendage_list,
                        [rudder_behind_skeg_value, rudder_behind_stern_value, twin_screw_ru
                         twin_screw_rudder_thick_value, shaft_brackets_value, skeg_value, s
```

```
                        hull_bossing_value, exposed_shafts_value, stabilizer_fins_value, d
                        bilge_keels_value]]
appendage_surface = [appendage_list, [rudder_behind_skeg_S, rudder_behind_stern_S, twin
                                      twin_screw_rudder_thick_S, shaft_brackets_S, skeg_
                                      hull_bossing_S, exposed_shafts_S, stabilizer_fins_

sum_appendage_value = 0.
sum_appendage_surface = 0.
# for i in range(len(appendage_list)):
#         if appendage_matrix[i-1][0]==True:
#             sum_appendage_value = sum_appendage_value + appendage_matrix[i-1][1]
#             sum_appendage_surface = sum_appendage_surface + appendage_surface[i-1][1]

K_2 = 0.
# for i in range(len(appendage_list)):
#     if appendage_matrix[i][0]==True:
#         K_2 = K_2+ (appendage_matrix[i-1][1] * appendage_surface[i-1][1])/(sum_appen

R_APP = 0.5 * rho * v ** 2. * K_2 * C_F * sum_appendage_surface + sum_bowthruster_resis

# wave resistance
if L_WL/B <= 12.0:
    labma = 1.446 * C_P - 0.03 * L_WL/B
elif L_WL/B > 12.0:
    labma = 1.446 * C_P - 0.36
d = -0.9
m_1 = 0.0140407 * L_WL/T - 1.75254 * (Vol**(1./3.) / L_WL) - 4.79323 * B/L_WL - c_16
m_3 = -7.2035 * (B / L_WL) ** (0.326869) * (T / B) ** (0.605375)
m_4 = 0.4 * c_15 * math.exp(-0.034 * Fn**(-3.29))

R_Wa = c_1 * c_2 * c_5 * rho * g * Vol * math.exp(m_1 * Fn**d + m_4 * math.cos(labma/(F
R_Wa04 = c_1 * c_2 * c_5 * rho * g * Vol * math.exp(m_1 * 0.4**d +(0.4 * c_15 * math.ex

R_Wb = c_17 * c_2 * c_5 * rho * g * Vol * math.exp(m_3 * Fn ** d + m_4 * math.cos(labma
R_Wb055 = c_17 * c_2 * c_5 * rho * g * Vol * math.exp(m_3 * 0.55 ** d + (0.4 * c_15 * m

R_Wc = R_Wa04 + (20. * Fn - 8.) / 3. * (R_Wb055 - R_Wa04)

if Fn <= 0.4:
    R_W = R_Wa
elif Fn >= 0.55:
    R_W = R_Wb
elif Fn > 0.4 and Fn < 0.55:
    R_W = R_Wc

# resistance of bulbous bow
h_F = C_P * C_M * (B * T / L_WL) * (136. - 316.3 * Fn) * Fn**3.
if h_F < -0.01 * L_WL:
    h_F = -0.01 * L_WL
```

```
h_W = (i_E * v**2.) / (400. * g)
if h_W > 0.01 * L_WL:
    h_W = 0.01 * L_WL

Fr_i = v / (math.sqrt(g * (T_F - h_B - 0.25 * math.sqrt(A_BT) + h_F + h_W)))


P_B = 0.56 * (math.sqrt(A_BT))/(T_F - 1.5 * h_B + h_F)

R_B = 0.11 * rho * g * (math.sqrt(A_BT))**3. * (Fr_i**3.)/(1. + Fr_i**2.) * math.exp(-3

# transom resistance
if A_T == 0.:
    c_6 = 0.
if A_T != 0.:
    Fr_T = v / (math.sqrt((2. * g * A_T)/(B + B * C_WP)))
    if Fr_T <= 5.:
        c_6 = 0.2 * (1. - 0.2 * Fr_T)
    else:
        c_6 = 0.

R_TR = 0.5 * rho * v**2. * c_6 * A_T

# correlation allowance resistance
if T_F/L_WL <= 0.04:
    c_4 = T_F/L_WL
else:
    c_4 = 0.04
C_A = 0.00546 * (L_WL + 100.)**(-0.16) - 0.002 + 0.003 * math.sqrt(L_WL/7.5) * C_B**4.

if k_S <= 150.:
    deltaC_A = 0.
elif k_S > 150.:
    deltaC_A = (0.105 * k_S**(1./3.) - 0.005579)/(L_WL**(1./3.))

R_A = 0.5 * rho * v**2. * (C_A + deltaC_A) * (S + sum_appendage_surface)

# air resistance
R_AA = 0.5 * rho_air * v**2. * C_DA * A_V

# total resistance
print('S', S)

print('K1', K_1)

print('Rf', R_f)
print('RA', R_A)
print('RAA', R_AA)
print('RAPP', R_APP)
```

```
    print('RW', R_W)
    print('RB', R_B)
    print('RTR', R_TR)

    R_total = K_1 * R_f + R_A + R_AA + R_APP + R_W + R_B + R_TR
    C_total = R_total / (0.5 * rho * v**2. * S)
    return R_total, C_total, S


def main():
    print("Hello World!")


    rho = 1025

    V = 15 * 0.5144  # speed from knots in m/s
    Lpp = 190.
    L_aft = 0.
    B = 32.3
    D = 0.
    T = 11.
    disp = 50218
    Vol = disp / rho
    LCB = 0
    C_B = 0.728
    C_M = 1./(1.+(1.-C_B)**3.5)
    vol = disp * rho
    A_V = 0.
    A_T = 0.0001
    A_BT = 0.0001
    h_B = 0.
    Fn = V / math.sqrt(Lpp * 9.8066)
    print('Fn',Fn)
    Rc, Cc, S = holtrop(1.01*Lpp,B,T,T,D,disp,LCB,C_B,C_M,A_V,A_T,A_BT,h_B,Fn)



    if V== 19. * 0.5144:
        RC_ben = 659193.122
        RC_app = 719000.
    if V == 24. * 0.5144:
        RC_ben = 1192302.81
        RC_app = 1482600.

    print('ich-Rc', Rc)
    print('ich-Cc', Cc)
```

```
if __name__ == '__main__':
    main()
```

F Python: slowly varying drift force

```
import numpy as np
import os
import matplotlib.pyplot as plt
import scipy.signal as sp
import scipy.fft as sf
import xlsxwriter
import scipy.interpolate as si
import openpyxl as open
from matplotlib.backends.backend_pdf import PdfPages
from numpy import random
import cmath
import math
from math import pi
from datetime import datetime




def total_Histogram(array: np.ndarray,numberofbins,):
    histogram_array = np.ndarray([numberofbins])
    bin_edges_array = np.linspace(0.,max(array),numberofbins+1)
    for i,edge in enumerate(histogram_array):
        index = np.argwhere((array >= bin_edges_array[i]) & (array <= bin_edges_array[i+1])
        histogram_array[i] = len(index)
    bin_center_array = bin_center(bin_edges_array)
    width_histogram = 0.8 * (bin_edges_array[0] - bin_edges_array[1])
    return histogram_array,bin_center_array,width_histogram

def FFT(array: np.ndarray,dt):
    array_fft = sf.fft(array)
    array_fft = np.abs(array_fft / len(array_fft))
    n = array_fft.size
    freq_plot = sf.fftfreq(n, dt)
    array_fft = array_fft[:int((len(array_fft) - 1) / 2)]
    freq_plot = freq_plot[:int((len(freq_plot) - 1) / 2)]
    array_fft = np.delete(array_fft,0)
    freq_plot = np.delete(freq_plot,0)

    return array_fft,freq_plot
def bin_center(binEdges):
    bincenters = 0.5 * (binEdges[1:] + binEdges[:-1])
    return bincenters
def FFT(array: np.ndarray,dt):
    array_fft = sf.fft(array)
    array_fft = np.abs(array_fft / len(array_fft))
    n = array_fft.size
    freq_plot = sf.fftfreq(n, dt)
```

```
    array_fft = array_fft[:int((len(array_fft) - 1) / 2)]
    freq_plot = freq_plot[:int((len(freq_plot) - 1) / 2)]
    return array_fft,freq_plot
def ensure_dir(file_path):
    directory = os.path.dirname(file_path)
    if not os.path.exists(directory):
        os.makedirs(directory)
def take_out_calmwater_run(array:np.ndarray):
    newarray = array
    newarray = np.delete(newarray,0)
    return newarray
def change_1and2(array:np.ndarray):
    array_1 = array[0]
    array_2 = array[1]
    array[0] = array_2
    array[1] = array_1
    return array
def make_empty_list():
    array = np.ndarray([])
    array = np.delete(array,0)
    return array
def JONSWAP(Tp, Hs,array_length,):
    array = np.linspace(3.5,18.5,num=array_length)
    increment = 15./array_length
    omega_p = 2*np.pi/Tp
    omega_array = 2*np.pi/array
    Sj_array = np.ndarray([len(array)])
    Ay = 1-0.287*np.log(3.3)
    sigma_a = 0.07
    sigma_b = 0.09
    for i,w in enumerate(omega_array):
        S_PM = 0.3125*Hs**(2)*omega_p**(4)*w**(-5)*np.exp(-1.25*(w/omega_p)**(-4))
        if w <= omega_p:
            S_J = Ay*S_PM*3.3**(np.exp(-0.5*((w-omega_p)/(sigma_a*omega_p))**2))
        elif w > omega_p:
            S_J = Ay*S_PM*3.3**(np.exp(-0.5*((w-omega_p)/(sigma_b*omega_p))**2))
        Sj_array[i] = S_J
    return array,Sj_array
def interpolation(xparr:np.ndarray, yparr:np.ndarray, xarr:np.ndarray ):
    yarr = np.ndarray(len(xarr))
    reversed_array = None
    if xparr[0]>xparr[1]:
        reversed_array = True
        rev_xparr = xparr[::-1]
        xparr = rev_xparr
        rev_yparr = yparr[::-1]
        yparr = rev_yparr
    for i,x in enumerate(xarr):
        yarr[i] = np.interp(x,xparr,yparr)
```

```
    return yarr


def find_irregularSeastate_AmplitudePeriod_Oeyvind(array: np.ndarray, timearray: np.ndarray
    array_mean = np.mean(array)
    index = np.argwhere(np.diff(np.sign(array)) > 0)
    index = index.transpose()
    index = index[0]

    period_array = make_empty_list()
    amplitude_array = make_empty_list()
    amplitude_array_long = np.ndarray([len(array)])
    period_array_long = np.ndarray([len(array)])
    newtime_array = make_empty_list()
    for i in range(1, len(index)):
        period = timearray[index[i]] - timearray[index[i - 1]]
        temp_array = array[index[i - 1]:index[i]]
        amplitude = 0.5 * (max(temp_array) - min(temp_array))
        newtime_array = np.append(newtime_array, timearray[index[i - 1]] + period * 0.5)
        amplitude_array = np.append(amplitude_array, amplitude)
        amplitude_array_long[index[i - 1]:index[i]] = amplitude
        period_array = np.append(period_array, period)
        period_array_long[index[i - 1]:index[i]] = period

    return period_array, amplitude_array, newtime_array, amplitude_array_long, period_array


def find_irregularSeastate_AmplitudePeriod_unsorted(array: np.ndarray, timearray: np.ndarra
    array_mean = np.mean(array)
    index_up = np.argwhere(np.diff(np.sign(array)) > 0)
    index_up = index_up.transpose()
    index_neg = np.argwhere(np.diff(np.sign(array)) < 0)
    index_neg = index_neg.transpose()
    index_up = index_up[0]
    index_neg = index_neg[0]
    index = np.append(index_up,index_neg)
    index = np.sort(index)

    period_array = make_empty_list()
    amplitude_array = make_empty_list()
    amplitude_array_long = np.ndarray([len(array)])
    period_array_long = np.ndarray([len(array)])
    newtime_array = make_empty_list()
    for i in range(1, len(index)):
        period = 2. * (timearray[index[i]] - timearray[index[i - 1]])
        temp_array = array[index[i - 1]:index[i]]
        amplitude = (max(temp_array) - min(temp_array))
        newtime_array = np.append(newtime_array, timearray[index[i - 1]] + period * 0.25)
        amplitude_array = np.append(amplitude_array, amplitude)
```

```
            amplitude_array_long[index[i - 1]:index[i]] = amplitude
            period_array = np.append(period_array, period)
            period_array_long[index[i - 1]:index[i]] = period

    return period_array, amplitude_array, newtime_array, amplitude_array_long, period_array


#data = pd.read_excel("mean_Fx_allruns.xlsx", engine='openpyxl')

# Give the location of the file
loc_shipX = ("ShipX_results.xlsx")
loc = ("mean_Fx_allruns.xlsx")

# To open Workbook
wb = open.load_workbook(loc)
sheet = wb.active
wb_shipX =  open.load_workbook(loc_shipX)
sheet_shipX = wb_shipX.active


# Figure path

#spectrum_parameter = np.array([6.5,1.5])              # Tp, Hs
spectrum_parameter = np.array([7.5,1.5])              # Tp, Hs
amountBINS = 500

figurepath = "./figure_JONSWAP3/"
string_file = 'Tp'+str(spectrum_parameter[0])+'Hs'+str(spectrum_parameter[1])
string_file = string_file.translate({ord(i): None for i in '.␣'})
ensure_dir(figurepath + string_file + '/')


workbook = xlsxwriter.Workbook(figurepath+string_file+ '/'+'JONSWAP_script.xlsx',options={'
worksheet = workbook.add_worksheet()
worksheet.write(0, 0,'Hp')
worksheet.write(2, 0,'Tp')
worksheet.write(1, 0,spectrum_parameter[1])
worksheet.write(3, 0,spectrum_parameter[0])

# create arrays
Fx_mean_15kn = np.array([])
Fx_mean_12kn = np.array([])
heave_RAO_15kn = np.array([])
heave_RAO_12kn = np.array([])
pitch_RAO_15kn = np.array([])
pitch_RAO_12kn = np.array([])
wp1_period_15kn = np.array([])
wp1_period_12kn = np.array([])
wp1_amplitude_15kn = np.array([])
```

```
wp1_amplitude_12kn = np.array([])
wp3_period_15kn = np.array([])
wp3_period_12kn = np.array([])
wp3_amplitude_15kn = np.array([])
wp3_amplitude_12kn = np.array([])
Fx_shipX_15kn = make_empty_list()
periods_shipX = make_empty_list()
wavenumber_shipX = make_empty_list()
added_res_shipX = make_empty_list()
added_res_dim_shipX = make_empty_list()

# write array from experimant data
for i in range(2,sheet.max_row+1):
    if i<22:
        Fx_mean_15kn = np.append(Fx_mean_15kn,abs(sheet.cell(i, 13).value))
        heave_RAO_15kn = np.append(Fx_mean_15kn, sheet.cell(i, 11).value)
        pitch_RAO_15kn = np.append(Fx_mean_15kn, sheet.cell(i, 12).value)
        wp1_period_15kn = np.append(wp1_period_15kn, sheet.cell(i, 4).value)
        wp3_period_15kn = np.append(wp3_period_15kn, sheet.cell(i, 5).value)
        wp1_amplitude_15kn = np.append(wp1_amplitude_15kn, (sheet.cell(i, 6).value)*0.5)
        wp3_amplitude_15kn = np.append(wp3_amplitude_15kn, (sheet.cell(i, 7).value) * 0.5)

    elif i>21:
        Fx_mean_12kn = np.append(Fx_mean_12kn,abs(sheet.cell(i, 13).value))
        heave_RAO_12kn = np.append(Fx_mean_12kn,sheet.cell(i, 11).value)
        pitch_RAO_12kn = np.append(Fx_mean_12kn,sheet.cell(i, 12).value)
        wp1_period_12kn = np.append(wp1_period_12kn, sheet.cell(i, 4).value)
        wp3_period_12kn = np.append(wp3_period_12kn, sheet.cell(i, 5).value)
        wp1_amplitude_12kn = np.append(wp1_amplitude_12kn, (sheet.cell(i, 6).value)*0.5)
        wp3_amplitude_12kn = np.append(wp3_amplitude_12kn, (sheet.cell(i, 7).value) * 0.5)


Fx_mean_15kn = change_1and2(Fx_mean_15kn)
Fx_mean_12kn = change_1and2(Fx_mean_12kn)
calm_water_Fx = [Fx_mean_15kn[0],Fx_mean_12kn[0]] # first entry 15kn; second entry 12kn
Fx_mean_dim_15kn = Fx_mean_15kn/calm_water_Fx[0]
Fx_mean_dim_12kn = Fx_mean_12kn/calm_water_Fx[1]
wp1_period_15kn= np.sort(wp1_period_15kn)
wp1_period_12kn = np.sort(wp1_period_12kn)
wp1_amplitude_15kn[0] = 0.
wp1_amplitude_12kn[0] = 0.
wp3_amplitude_15kn[0] = 0.
wp3_amplitude_12kn[0] = 0.


# write ShipX data into arrays
for i in range(4,sheet.max_row+1):
    periods_shipX = np.append(periods_shipX, sheet_shipX.cell(i, 2).value)
    Fx_shipX_15kn = np.append(Fx_shipX_15kn,sheet_shipX.cell(i,28).value)
```

```
    wavenumber_shipX = np.append(wavenumber_shipX,sheet_shipX.cell(i,3).value)
    added_res_shipX = np.append(added_res_shipX, sheet_shipX.cell(i, 42).value)
    added_res_dim_shipX = np.append(added_res_dim_shipX, sheet_shipX.cell(i, 42).value)

Fx_RAO_shipX_15kn = np.ndarray([len(Fx_shipX_15kn)])
Fx_dimless_shipX = np.ndarray([len(Fx_shipX_15kn)])

wavelength_shipX = 2*pi/wavenumber_shipX
waveamplitude_shipX = np.ndarray([len(periods_shipX)])

for i, length in enumerate(wavelength_shipX):
# wave amplitude shipX
    waveamplitude_shipX[i] = 0.5 * 0.025 * length

for i, f in enumerate(Fx_shipX_15kn):                                    # added resistan
    Fx_dimless_shipX[i] = f * 190.00 / (9.81 * 1000 * 32.22 ** 2)
    added_res_shipX[i] = added_res_shipX[i] * 1000 * 9.81 * waveamplitude_shipX[i] ** 2 * 3

waveamplitude_shipXtoexp = interpolation(periods_shipX, waveamplitude_shipX, wp3_period_15k
added_res_experiment_15kn = np.ndarray([len(Fx_mean_dim_15kn)])

for i, f in enumerate(Fx_mean_dim_15kn):                                 # added resis
    added_res_experiment_15kn[i] = f * 1000 * 9.81 * (wp3_amplitude_15kn[i] * 32) ** 2 * 32

enc_freq_shipX = np.ndarray([len(periods_shipX)])
enc_period_shipX = np.ndarray([len(periods_shipX)])
shipX_omega = np.ndarray([len(periods_shipX)])

for i,T in enumerate(periods_shipX):
    shipX_omega[i] = 2*np.pi/T
    enc_freq_shipX[i] = shipX_omega[i]+wavenumber_shipX[i]*7.72
    enc_period_shipX[i] = 2*np.pi/enc_freq_shipX[i]




# take out calm water run
wCW_Fx_mean_15kn = take_out_calmwater_run(Fx_mean_dim_15kn*32**3)
wCW_Fx_mean_12kn = take_out_calmwater_run(Fx_mean_12kn)
wCW_wp1_period_15kn = take_out_calmwater_run(wp1_period_15kn)
wCW_wp1_period_12kn = take_out_calmwater_run(wp1_period_12kn)
wCW_wp1_amplitude_15kn = take_out_calmwater_run(wp1_amplitude_15kn)
wCW_wp1_amplitude_12kn = take_out_calmwater_run(wp1_amplitude_12kn)
wCW_wp3_period_15kn = take_out_calmwater_run(wp3_period_15kn)
wCW_wp3_period_12kn = take_out_calmwater_run(wp3_period_12kn)
wCW_wp3_amplitude_15kn = take_out_calmwater_run(wp3_amplitude_15kn)
wCW_wp3_amplitude_12kn = take_out_calmwater_run(wp3_amplitude_12kn)
wCW_added_res_experiment_15kn = take_out_calmwater_run(added_res_experiment_15kn)
```

```
# mean heave force RAO
RAO_Fx_15kn = wCW_added_res_experiment_15kn/((wCW_wp3_amplitude_15kn*32)**2)
Fx_RAO_shipX_15kn = added_res_shipX/(waveamplitude_shipX)**2

# create omega array and time array
wp1_omega_15kn = 2*pi/wp1_period_15kn

wCW_wp1_omega_15kn = take_out_calmwater_run(wp1_omega_15kn)
incrementstep = 0.01
omega = np.linspace(wCW_wp1_omega_15kn[0],wCW_wp1_omega_15kn[-1],num=int(wCW_wp1_omega_15kn



time_increment = 0.2
time_increment1 = int(1./time_increment)
time = np.linspace(0.,10800.,num=1+10800*time_increment1)



spectrum_period,spectrum_energy = JONSWAP(spectrum_parameter[0],spectrum_parameter[1],len(o
spectrum_omega = 2*pi/spectrum_period
spectrum_ampl = np.sqrt(2*spectrum_energy)
random_phase1 = random.rand(len(spectrum_omega))*2*pi
random_phase2 = random.rand(len(spectrum_omega))*2*pi
random_phase3 = random.rand(len(spectrum_omega))*2*pi
random_phase4 = random.rand(len(spectrum_omega))*2*pi
random_phase5 = random.rand(len(spectrum_omega))*2*pi
random_phases = np.array([random_phase1,random_phase2,random_phase3,random_phase4,random_ph

irregularSea_JONSWAP = np.ndarray([len(time)])
irregularSea_JONSWAP_inside = np.ndarray([len(spectrum_omega)])
for i,t in enumerate(time):
    for j,w in enumerate(spectrum_omega):
            irregularSea_JONSWAP_inside[j] =spectrum_ampl[j] * np.cos(w*t*random_phase1[j])
    irregularSea_JONSWAP[i] = np.sum(irregularSea_JONSWAP_inside)
JONSWAP_period_loeken, JONSWAP_amplitude_loeken, JONSWAP_time_loeken, JONSWAP_amplitude_lo
JONSWAP_period_oeyvind, JONSWAP_amplitude_oeyvind, JONSWAP_time_oeyvind, JONSWAP_amplitude
JONSWAP_omega_loeken = 2.*np.pi/JONSWAP_period_loeken
JONSWAP_omega_long_loeken = 2.*np.pi/JONSWAP_period_long_loeken
JONSWAP_omega_oeyvind = 2.*np.pi/JONSWAP_period_oeyvind
JONSWAP_omega_long_oeyvind = 2.*np.pi/JONSWAP_period_long_oeyvind

worksheet.write(0, 2, 'amplitude Loeken')
worksheet.write(0, 4, 'amplitude Oeyvind')
worksheet.write(0, 1, 'period Loeken')
worksheet.write(0, 3, 'period Oeyvind')
for i,ampl in enumerate(JONSWAP_amplitude_loeken):
```

```
    worksheet.write(i+1, 2,ampl)
for i,ampl in enumerate(JONSWAP_amplitude_oeyvind):
    worksheet.write(i+1, 4,ampl)
for i,ampl in enumerate(JONSWAP_period_loeken):
    worksheet.write(i+1, 1,ampl)
for i,ampl in enumerate(JONSWAP_period_oeyvind):
    worksheet.write(i+1, 3,ampl)


# interpolate
Fx_mean_interpol_15kn = np.ndarray([len(omega)])
RAO_Fx_interpol_15kn = np.ndarray([len(omega)])
wp3_amplitude_interpol_15kn = np.ndarray([len(omega)])



Fx_mean_interpol_15kn = interpolation(wCW_wp1_omega_15kn, wCW_Fx_mean_15kn,omega)
RAO_Fx_interpol_15kn = interpolation(wCW_wp1_omega_15kn, RAO_Fx_15kn,spectrum_omega)
wp3_amplitude_interpol_15kn = interpolation(wCW_wp1_omega_15kn, wCW_wp3_amplitude_15kn, spec
Fx_RAO_interpol_shipX_15kn = interpolation(shipX_omega,Fx_RAO_shipX_15kn,spectrum_omega)
JONSWAP_Fx_RAO_loeken_15kn = interpolation(wCW_wp1_omega_15kn/np.sqrt(32),RAO_Fx_15kn,JONSW
JONSWAP_Fx_RAO_long_loeken_15kn = interpolation(wCW_wp1_omega_15kn/np.sqrt(32.),RAO_Fx_15kn
JONSWAP_Fx_RAO_oeyvind_15kn = interpolation(wCW_wp1_omega_15kn/np.sqrt(32.),RAO_Fx_15kn,JON
JONSWAP_Fx_RAO_long_oeyvind_15kn = interpolation(wCW_wp1_omega_15kn/np.sqrt(32.),RAO_Fx_15k


# timeseries JONSWAP
JONSWAP_Fx_loeken_15kn = JONSWAP_Fx_RAO_loeken_15kn*JONSWAP_amplitude_loeken**2
JONSWAP_Fx_long_loeken_15kn = JONSWAP_Fx_RAO_long_loeken_15kn*JONSWAP_amplitude_long_loeken
JONSWAP_Fx_oeyvind_15kn = JONSWAP_Fx_RAO_oeyvind_15kn*JONSWAP_amplitude_oeyvind**2
JONSWAP_Fx_long_oeyvind_15kn = JONSWAP_Fx_RAO_long_oeyvind_15kn*JONSWAP_amplitude_long_oeyv
print('creating JONSWAP timeseries',datetime.now())

# setup for Newman approx
Fx_Newman_vec1 = np.ndarray([len(time)])
Fx_Newman_vec_shipX1 = np.ndarray([len(time)])
Fx_Newman_vec2 = np.ndarray([len(time)])
Fx_Newman_vec_shipX2 = np.ndarray([len(time)])
Fx_Newman_vec3 = np.ndarray([len(time)])
Fx_Newman_vec_shipX3 = np.ndarray([len(time)])
Fx_Newman_vec4 = np.ndarray([len(time)])
Fx_Newman_vec_shipX4 = np.ndarray([len(time)])
Fx_Newman_vec5 = np.ndarray([len(time)])
Fx_Newman_vec_shipX5 = np.ndarray([len(time)])
Fx_Newman_vecs_15kn= np.array([Fx_Newman_vec1,Fx_Newman_vec2,Fx_Newman_vec3,Fx_Newman_vec4,
Fx_Newman_vecs_shipX_15kn = np.array([Fx_Newman_vec_shipX1,Fx_Newman_vec_shipX2,Fx_Newman_

Fx_Newman_vec_mean_15kn = np.ndarray([len(time)])
Fx_Newman_vec_shipX_mean_15kn = np.ndarray([len(time)])
```

```
Fx_Newman_vecs_deltas_15kn = np.array([Fx_Newman_vec1,Fx_Newman_vec2,Fx_Newman_vec3,Fx_Newm
Fx_Newman_vecs_shipX_15kn_deltas_15kn = np.array([Fx_Newman_vec_shipX1,Fx_Newman_vec_shipX2


inner_vector= np.zeros([len(spectrum_omega)])
inner_vector_shipX= np.zeros([len(spectrum_omega)])



# Newman


for index,phase in  enumerate(random_phases):
    for j,t in enumerate(time):
        for i in spectrum_omega:
            range_omega = make_empty_list()
            range_omega = np.argwhere(abs((spectrum_omega-i)/i) <= 0.05)
            for m in range_omega:
                inner = (spectrum_ampl[m] * math.sqrt(RAO_Fx_interpol_15kn[m]) * np.cos(spec
                inner_vector[m] = inner
                inner_shipX = (spectrum_ampl[m] * math.sqrt(Fx_RAO_interpol_shipX_15kn[m])
                inner_vector_shipX[m] = inner_shipX
        Fx_Newman_vecs_15kn[index][j] = 2.*np.sum(inner_vector)**2
        Fx_Newman_vecs_shipX_15kn[index][j] = 2.*np.sum(inner_vector_shipX)**2
    print('...realisation',index+1,'out of',len(random_phases),':' ,datetime.now())

for j,newman1 in enumerate(Fx_Newman_vec1):
    Fx_Newman_vec_mean_15kn[j] = 0.2*(Fx_Newman_vecs_15kn[0][j]+Fx_Newman_vecs_15kn[1][j]+F
    Fx_Newman_vec_shipX_mean_15kn[j] = 0.2*(Fx_Newman_vecs_shipX_15kn[0][j]+Fx_Newman_vecs_

for i, delta in enumerate(Fx_Newman_vecs_deltas_15kn):
    for j, newman in enumerate(Fx_Newman_vec1):
        delta[j] = Fx_Newman_vecs_15kn[i][j] - Fx_Newman_vec_mean_15kn[j]
        Fx_Newman_vecs_shipX_15kn_deltas_15kn[i][j] = Fx_Newman_vecs_shipX_15kn[i][j] - Fx_


# calculate histogram
hist_max_15kn = np.max(Fx_Newman_vecs_15kn)
hist_Fx1_15kn, bin_edges_Fx1_15kn = np.histogram(Fx_Newman_vecs_15kn[0], range=(0.,hist_max
hist_Fx2_15kn, bin_edges_Fx2_15kn = np.histogram(Fx_Newman_vecs_15kn[1], range=(0.,hist_max
hist_Fx3_15kn, bin_edges_Fx3_15kn = np.histogram(Fx_Newman_vecs_15kn[2], range=(0.,hist_max
hist_Fx4_15kn, bin_edges_Fx4_15kn = np.histogram(Fx_Newman_vecs_15kn[3], range=(0.,hist_max
hist_Fx5_15kn, bin_edges_Fx5_15kn = np.histogram(Fx_Newman_vecs_15kn[4], range=(0.,hist_max
hist_Fx1_15kn = hist_Fx1_15kn/sum(hist_Fx1_15kn)
hist_Fx2_15kn = hist_Fx2_15kn/sum(hist_Fx2_15kn)
hist_Fx3_15kn = hist_Fx3_15kn/sum(hist_Fx3_15kn)
hist_Fx4_15kn = hist_Fx4_15kn/sum(hist_Fx4_15kn)
hist_Fx5_15kn = hist_Fx5_15kn/sum(hist_Fx5_15kn)
```

```
width_histogram_15kn = 0.8*(bin_edges_Fx1_15kn[0]-bin_edges_Fx1_15kn[1])
bin_center_Fx1_15kn = bin_center(bin_edges_Fx1_15kn)
bin_center_Fx2_15kn = bin_center(bin_edges_Fx2_15kn)
bin_center_Fx3_15kn = bin_center(bin_edges_Fx3_15kn)
bin_center_Fx4_15kn = bin_center(bin_edges_Fx4_15kn)
bin_center_Fx5_15kn = bin_center(bin_edges_Fx5_15kn)


hist_max_shipX_15kn = np.max(Fx_Newman_vecs_shipX_15kn)
hist_Fx_shipX1_15kn, bin_edges_Fx_shipX1_15kn = np.histogram(Fx_Newman_vecs_shipX_15kn[0],
hist_Fx_shipX2_15kn, bin_edges_Fx_shipX2_15kn = np.histogram(Fx_Newman_vecs_shipX_15kn[1],
hist_Fx_shipX3_15kn, bin_edges_Fx_shipX3_15kn = np.histogram(Fx_Newman_vecs_shipX_15kn[2],
hist_Fx_shipX4_15kn, bin_edges_Fx_shipX4_15kn = np.histogram(Fx_Newman_vecs_shipX_15kn[3],
hist_Fx_shipX5_15kn, bin_edges_Fx_shipX5_15kn = np.histogram(Fx_Newman_vecs_shipX_15kn[4],
hist_Fx_shipX1_15kn = hist_Fx_shipX1_15kn/sum(hist_Fx_shipX1_15kn)
hist_Fx_shipX2_15kn = hist_Fx_shipX2_15kn/sum(hist_Fx_shipX2_15kn)
hist_Fx_shipX3_15kn = hist_Fx_shipX3_15kn/sum(hist_Fx_shipX3_15kn)
hist_Fx_shipX4_15kn = hist_Fx_shipX4_15kn/sum(hist_Fx_shipX4_15kn)
hist_Fx_shipX5_15kn = hist_Fx_shipX5_15kn/sum(hist_Fx_shipX5_15kn)
width_histogram_shipX_15kn = 0.8*(bin_edges_Fx_shipX1_15kn[0]-bin_edges_Fx_shipX1_15kn[1])
bin_center_Fx_shipX1_15kn = bin_center(bin_edges_Fx_shipX1_15kn)
bin_center_Fx_shipX2_15kn= bin_center(bin_edges_Fx_shipX2_15kn)
bin_center_Fx_shipX3_15kn= bin_center(bin_edges_Fx_shipX3_15kn)
bin_center_Fx_shipX4_15kn= bin_center(bin_edges_Fx_shipX4_15kn)
bin_center_Fx_shipX5_15kn= bin_center(bin_edges_Fx_shipX5_15kn)


mean_hist_15kn = np.ndarray([len(bin_center_Fx1_15kn)])
hist_delta_15kn = np.ndarray([len(hist_Fx1_15kn)])
for i,bin1 in enumerate(hist_delta_15kn):
    mean_bin = 0.2*(hist_Fx1_15kn[i] + hist_Fx2_15kn[i] + hist_Fx3_15kn[i] + hist_Fx4_15kn[
    mean_hist_15kn[i] = mean_bin
    hist_delta_15kn[i] = np.sqrt(((hist_Fx1_15kn[i]-mean_bin)**2+(hist_Fx2_15kn[i]-mean_bin

mean_hist_shipX = np.ndarray([len(bin_center_Fx_shipX1_15kn)])
hist_delta_shipX_15kn = np.ndarray([len(hist_Fx_shipX1_15kn)])
for i,bin1 in enumerate(hist_delta_shipX_15kn):
    mean_bin = 0.2*(hist_Fx_shipX1_15kn[i] + hist_Fx_shipX2_15kn[i] + hist_Fx_shipX3_15kn[i
    mean_hist_shipX[i] = mean_bin
    hist_delta_shipX_15kn[i] = np.sqrt(((hist_Fx_shipX1_15kn[i]-mean_bin)**2+(hist_Fx_shipX

for i,pip in enumerate(hist_delta_15kn):
    print('histogram delta',i, hist_delta_15kn[i],hist_delta_shipX_15kn[i])




hist_Fx_JONSWAP_loeken_15kn, bin_edges_Fx_JONSWAP_loeken_15kn = np.histogram(JONSWAP_Fx_lo
bin_center_Fx_JONSWAP_loeken_15kn= bin_center(bin_edges_Fx_JONSWAP_loeken_15kn)
```

```
width_histogram_JONSWAP_loeken_15kn = 0.8*(bin_edges_Fx_JONSWAP_loeken_15kn[0]-bin_edges_Fx_
hist_Fx_JONSWAP_loeken_15kn = hist_Fx_JONSWAP_loeken_15kn/sum(hist_Fx_JONSWAP_loeken_15kn)

hist_Fx_JONSWAP_oeyvind_15kn, bin_edges_Fx_JONSWAP_oeyvind_15kn = np.histogram(JONSWAP_Fx_
bin_center_Fx_JONSWAP_oeyvind_15kn= bin_center(bin_edges_Fx_JONSWAP_oeyvind_15kn)
width_histogram_JONSWAP_oeyvind_15kn = 0.8*(bin_edges_Fx_JONSWAP_oeyvind_15kn[0]-bin_edges_F
hist_Fx_JONSWAP_oeyvind_15kn = hist_Fx_JONSWAP_oeyvind_15kn/sum(hist_Fx_JONSWAP_oeyvind_15k

hist_oeyvind_long_15kn, bin_center_oeyvind_long_15kn,width_histogram_oeyvind_long_15kn = to
hist_oeyvind_long_15kn = hist_oeyvind_long_15kn/sum(hist_oeyvind_long_15kn)
hist_loeken_long_15kn, bin_center_loeken_long_15kn,width_histogram_loeken_long_15kn = total_
hist_loeken_long_15kn = hist_loeken_long_15kn/sum(hist_loeken_long_15kn)


# FFT

FFT_oeyvind_15kn,freq_oeyvind_15kn = FFT(JONSWAP_Fx_loeken_15kn,time_increment)
FFT_realisation1_15kn,freq_realisation1_15kn = FFT(Fx_Newman_vecs_15kn[0],time_increment)
FFT_realisation2_15kn,freq_realisation2_15kn = FFT(Fx_Newman_vecs_15kn[1],time_increment)
FFT_realisation3_15kn,freq_realisation3_15kn = FFT(Fx_Newman_vecs_15kn[2],time_increment)
FFT_realisation4_15kn,freq_realisation4_15kn = FFT(Fx_Newman_vecs_15kn[3],time_increment)
FFT_realisation5_15kn,freq_realisation5_15kn = FFT(Fx_Newman_vecs_15kn[4],time_increment)
FFT_mean_15kn,freq_mean_15kn = FFT(Fx_Newman_vec_mean_15kn,time_increment)



# sorting RAOs
JONSWAP_Fx_RAO_loeken_15kn = [x for _, x in sorted(zip(JONSWAP_omega_loeken, JONSWAP_Fx_RA
JONSWAP_Fx_RAO_loeken_15kn = np.array(JONSWAP_Fx_RAO_loeken_15kn)
JONSWAP_omega_loeken = sorted(JONSWAP_omega_loeken)
JONSWAP_omega_loeken = np.array(JONSWAP_omega_loeken)

# plot

pp = PdfPages(figurepath + string_file + '/' + 'Newman-approx_15kn.pdf')
ppp = PdfPages(figurepath + string_file + '/' + 'Newman-approx-shipX_15kn.pdf')
ppfft = PdfPages(figurepath + string_file + '/' + 'FFT_15kn.pdf')
ppmethod2 = PdfPages(figurepath + string_file + '/' + 'method2.pdf')



plt.figure(1)
plt.grid()
plt.title('added␣resistance␣dimensionless')#:  '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(sp
plt.plot(wp1_period_15kn*5.657,Fx_mean_dim_15kn,'-',label='experiment')
plt.plot(periods_shipX,added_res_dim_shipX,'-',label='shipX',color='darkorange')
plt.xlabel('wave␣period[s]␣')
plt.ylabel(r'$R␣/␣\rho␣\cdot␣g␣\cdot␣A^2␣\cdot␣B^2/L_{PP}␣[-]$')
plt.legend(loc='best')
```

```
plt.savefig(pp, format='pdf',bbox_inches="tight")
plt.savefig(ppp, format='pdf',bbox_inches="tight")
plt.savefig(ppfft, format='pdf',bbox_inches="tight")
plt.savefig(ppmethod2, format='pdf',bbox_inches="tight")
plt.close(1)



plt.figure(2)
plt.grid()
plt.title('RAO added resistance')#: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(spectrum_par
plt.plot(wCW_wp1_period_15kn*np.sqrt(32),RAO_Fx_15kn,'-',label='experiment')
plt.plot(periods_shipX,Fx_RAO_shipX_15kn,'-',label='shipX',color='darkorange')
plt.ylabel(r'$f^{SV}/amplitude [N/m^2]$')
plt.xlabel('wave period[s]' )
plt.legend(loc='best')
plt.savefig(pp, format='pdf',bbox_inches="tight")
plt.savefig(ppp, format='pdf',bbox_inches="tight")
plt.savefig(ppfft, format='pdf',bbox_inches="tight")
plt.savefig(ppmethod2, format='pdf',bbox_inches="tight")
plt.close(2)

plt.figure(1)
plt.grid()
plt.title('JONSWAP spectrum: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(spectrum_parameter[
plt.plot(spectrum_omega,spectrum_energy,'-')#,label='experiment')
plt.xlabel('omega[rad/s] ' )
plt.ylabel(r'$S(\omega)$')
plt.legend(loc='best')
plt.savefig(pp, format='pdf',bbox_inches="tight")
plt.savefig(ppp, format='pdf',bbox_inches="tight")
plt.savefig(ppfft, format='pdf',bbox_inches="tight")
plt.savefig(ppmethod2, format='pdf',bbox_inches="tight")
plt.close(1)

fig = plt.figure('mean values experiment')
plt.suptitle('realisations mean value experiment: '+'Tp'+str(spectrum_parameter[0])+'Hs'+st
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vec_mean_15kn, '-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx1_15kn,mean_hist_15kn,label='histogram experiment',width=width_histogra
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel(' amount')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(pp, format='pdf')
```

```
fig.savefig(ppp, format='pdf')
plt.close(fig)

fig = plt.figure('mean values shipX')
plt.suptitle('realisations mean value shipX: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(spec
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vec_shipX_mean_15kn, '-',label='Newman-approx-shipX',color='darkora
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx_shipX1_15kn,mean_hist_shipX,label='histogram shipX',width=width_histog
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel(' amount')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(pp, format='pdf')
fig.savefig(ppp, format='pdf')
plt.close(fig)


fig = plt.figure('realisation1')
plt.suptitle('Newman-approx(rel 1) timeseries and difference to mean value: '+'Tp'+str(spect
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[0],'-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_deltas_15kn[0],'-',label='delta')
ax2.set_xlabel('time[s]' )
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
#fig.savefig(pp, format='pdf')
plt.close(fig)

fig = plt.figure('realisation1 histogram')
plt.suptitle('Newman-approx(rel 1) timeseries and histogram: '+'Tp'+str(spectrum_parameter[
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[0],'-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,label='histogram experiment',width=width_histogram
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
```

```
ax2.legend()
fig.savefig(pp, format='pdf')
plt.close(fig)

fig = plt.figure('realisation2')
plt.suptitle('Newman-approx(rel 2) timeseries and difference to mean value: '+'Tp'+str(spect
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[1], '-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_deltas_15kn[1],'-',label='delta')
ax2.set_xlabel('time[s]' )
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
#fig.savefig(pp, format='pdf')
plt.close(fig)

fig = plt.figure('realisation2 histogram')
plt.suptitle('Newman-approx(rel 2) timeseries and histogram: '+'Tp'+str(spectrum_parameter[0
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[1],'-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx2_15kn,hist_Fx2_15kn,label='histogram experiment',width=width_histogram
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(pp, format='pdf')
plt.close(fig)


fig = plt.figure('realisation3')
plt.suptitle('Newman-approx(rel 3) timeseries and difference to mean value: '+'Tp'+str(spect
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[2], '-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_deltas_15kn[2],'-',label='delta')
ax2.set_xlabel('time[s]' )
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
```

```
ax2.legend()

#fig.savefig(pp, format='pdf')
plt.close(fig)

fig = plt.figure('realisation3 histogram')
plt.suptitle('Newman-approx(rel 3) timeseries and histogram: '+'Tp'+str(spectrum_parameter[0
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[2],'-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx3_15kn,hist_Fx3_15kn,label='histogram experiment',width=width_histogram
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(pp, format='pdf')
plt.close(fig)


fig = plt.figure('realisation4')
plt.suptitle('Newman-approx(rel 4) timeseries and difference to mean value: '+'Tp'+str(spect
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[3], '-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_deltas_15kn[3],'-',label='delta')
ax2.set_xlabel('time[s]' )
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
#fig.savefig(pp, format='pdf')
plt.close(fig)

fig = plt.figure('realisation4 histogram')
plt.suptitle('Newman-approx(rel 4) timeseries and histogram: '+'Tp'+str(spectrum_parameter[0
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[3],'-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx4_15kn,hist_Fx4_15kn,label='histogram experiment',width=width_histogram
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
fig.savefig(pp, format='pdf')
plt.close(fig)
```

```
fig = plt.figure('realisation5')
plt.suptitle('Newman-approx(rel 5) timeseries and difference to mean value: '+'Tp'+str(spect
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[4], '-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_deltas_15kn[4],'-',label='delta')
ax2.set_xlabel('time[s]')
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
#fig.savefig(pp, format='pdf')
plt.close(fig)

fig = plt.figure('realisation5 histogram')
plt.suptitle('Newman-approx(rel 5) timeseries and histogram: '+'Tp'+str(spectrum_parameter[
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_15kn[4],'-',label='Newman-approx')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx5_15kn,hist_Fx5_15kn,label='histogram experiment',width=width_histogram
ax2.set_xlabel(r'$f^{SV}[N]$')
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(pp, format='pdf')
plt.close(fig)


fig = plt.figure('Newman-approx-shipX realisation1')
plt.suptitle('Newman-approx-shipX(rel 1) timeseries and difference to mean value: '+'Tp'+st
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[0], '-',label='Newman-approx-shipX',color='darkoran
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_shipX_15kn_deltas_15kn[0],'-',label='delta',color='darkorange'
ax2.set_xlabel('time[s]')
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
#fig.savefig(ppp, format='pdf')
```

```
plt.close(fig)

fig = plt.figure('Newman-approx-shipX realisation1 histogram')
plt.suptitle('Newman-approx-shipX(rel 1) timeseries and histogram: '+'Tp'+str(spectrum_para
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[0],'-',label='Newman-approx-shipX',color='darkorang
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx_shipX1_15kn,hist_Fx_shipX1_15kn,label='histogram shipX',width=width_h
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(ppp, format='pdf')
plt.close(fig)


fig = plt.figure('Newman-approx-shipX realisation2')
plt.suptitle('Newman-approx-shipX(rel 2) timeseries and difference to mean value: '+'Tp'+st
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[1], '-',label='Newman-approx-shipX',color='darkoran
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_shipX_15kn_deltas_15kn[1],'-',label='delta',color='darkorange'
ax2.set_xlabel('time[s]' )
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
#fig.savefig(ppp, format='pdf')
plt.close(fig)

fig = plt.figure('Newman-approx-shipX realisation2 histogram')
plt.suptitle('Newman-approx-shipX(rel 2) timeseries and histogram: '+'Tp'+str(spectrum_para
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[1],'-',label='Newman-approx-shipX',color='darkorang
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx_shipX2_15kn,hist_Fx_shipX2_15kn,label='histogram shipX',width=width_h
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(ppp, format='pdf')
```

```
plt.close(fig)


fig = plt.figure('Newman-approx-shipX realisation3')
plt.suptitle('Newman-approx-shipX(rel 3) timeseries and difference to mean value: '+'Tp'+st
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[2], '-',label='Newman-approx-shipX',color='darkoran
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_shipX_15kn_deltas_15kn[2],'-',label='delta',color='darkorange'
ax2.set_xlabel('time[s]' )
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
#fig.savefig(ppp, format='pdf')
plt.close(fig)

fig = plt.figure('Newman-approx-shipX realisation3 histogram')
plt.suptitle('Newman-approx-shipX(rel 3) timeseries and histogram: '+'Tp'+str(spectrum_para
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[2],'-',label='Newman-approx-shipX',color='darkorang
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx_shipX3_15kn,hist_Fx_shipX3_15kn,label='histogram shipX',width=width_h
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(ppp, format='pdf')
plt.close(fig)


fig = plt.figure('Newman-approx-shipX realisation4')
plt.suptitle('Newman-approx-shipX(rel 4) timeseries and difference to mean value: '+'Tp'+st
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[3], '-',label='Newman-approx-shipX',color='darkoran
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_shipX_15kn_deltas_15kn[3],'-',label='delta',color='darkorange'
ax2.set_xlabel('time[s]' )
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
```

```
#fig.savefig(ppp, format='pdf')
plt.close(fig)

fig = plt.figure('Newman-approx-shipX realisation4 histogram')
plt.suptitle('Newman-approx-shipX(rel 4) timeseries and histogram: '+'Tp'+str(spectrum_para
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[3],'-',label='Newman-approx-shipX',color='darkorang
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx_shipX4_15kn,hist_Fx_shipX4_15kn,label='histogram shipX',width=width_h
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(ppp, format='pdf')
plt.close(fig)


fig = plt.figure('Newman-approx-shipX realisation5')
plt.suptitle('Newman-approx-shipX(rel 5) timeseries and difference to mean value: '+'Tp'+st
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[4], '-',label='Newman-approx-shipX',color='darkoran
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,Fx_Newman_vecs_shipX_15kn_deltas_15kn[4],'-',label='delta',color='darkorange'
ax2.set_xlabel('time[s]' )
ax2.set_ylabel(r'$ \Delta slowdrift[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
#fig.savefig(ppp, format='pdf')
plt.close(fig)

fig = plt.figure('Newman-approx-shipX realisation5 histogram')
plt.suptitle('Newman-approx-shipX(rel 5) timeseries and histogram: '+'Tp'+str(spectrum_para
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,Fx_Newman_vecs_shipX_15kn[4],'-',label='Newman-approx-shipX',color='darkorang
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx_shipX5_15kn,hist_Fx_shipX5_15kn,label='histogram shipX',width=width_h
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
```

```
fig.savefig(ppp, format='pdf')
plt.close(fig)


fig = plt.figure('fft realisation ')
plt.grid()
plt.suptitle('FFT slowly varying drift force: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(sp
plt.plot(1./freq_realisation1_15kn ,FFT_realisation1_15kn, '-',label='realisation 1')
plt.plot(1./freq_realisation2_15kn ,FFT_realisation2_15kn, '-',label='realisation 2')
plt.plot(1./freq_realisation3_15kn ,FFT_realisation3_15kn, '-',label='realisation 3')
plt.plot(1./freq_realisation4_15kn ,FFT_realisation4_15kn, '-',label='realisation 4')
plt.plot(1./freq_realisation5_15kn ,FFT_realisation5_15kn, '-',label='realisation 5')
plt.legend(loc='best')
plt.xlabel('period[s]')
plt.ylabel(r'$f^{SV}[N]$')
fig.savefig(ppfft, format='pdf')
plt.close(fig)

plt.figure('RAOs')
plt.grid()
plt.title('JONSWAP RAO vs. experiment RAO')
plt.plot(2*np.pi/(wCW_wp1_omega_15kn/5.657),RAO_Fx_15kn, '-',label='RAO experiment')
plt.plot(2*np.pi/JONSWAP_omega_loeken,JONSWAP_Fx_RAO_loeken_15kn, '-',label='RAO Faltinsen/L
plt.xlabel('period[s]' )
plt.ylabel('RAO[N/m ] ' )
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf',bbox_inches="tight")

fig = plt.figure('JONSWAP timeseries ')
plt.grid()
plt.suptitle('slowly varying drift force timeseries: '+'Tp'+str(spectrum_parameter[0])+'Hs'
plt.plot(JONSWAP_time_loeken,JONSWAP_Fx_loeken_15kn, '-',label='Faltinsen/Loeken')
plt.legend(loc='best')
plt.xlabel('time[s]')
plt.ylabel(r'$f^{SV}[N]$')
plt.savefig(ppmethod2, format='pdf')
plt.close(fig)

fig = plt.figure('JONSWAP: method2 timeseries ')
plt.grid()
plt.suptitle('slowly varying drift force timeseries: '+'Tp'+str(spectrum_parameter[0])+'Hs'
#plt.plot(time,JONSWAP_Fx_long_15kn, '-',label='Faltinsen/Loeken')
plt.plot(time,JONSWAP_Fx_long_oeyvind_15kn, '-',label='Oeyvind')
plt.legend(loc='best')
plt.xlabel('time[s]')
plt.ylabel(r'$f^{SV}[N]$')
plt.savefig(ppmethod2, format='pdf')
plt.close(fig)
```

```
fig = plt.figure('method2')
plt.suptitle('Timeseries Faltinsen/Loeken vs Oeyvind: '+'Tp'+str(spectrum_parameter[0])+'Hs
ax1 = fig.add_subplot(2,1,1)
ax1.plot(JONSWAP_time_loeken,JONSWAP_Fx_loeken_15kn,'-',label='method2 Loeken/Faltinsen')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(time,JONSWAP_Fx_long_oeyvind_15kn, '-',label='Oeyvind')
ax2.set_xlabel('time[s]' )
ax2.set_ylabel(r'$f^{SV}[N]$')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
plt.savefig(ppmethod2, format='pdf')
plt.close(fig)



fig = plt.figure('fft ')
plt.suptitle('FFT slowly varying drift force: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(sp
ax1 = fig.add_subplot(2,1,1)
ax1.plot(1./freq_mean_15kn ,FFT_mean_15kn, '-',label='mean')
ax1.set_xlabel('period[s]')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax1.legend(loc='best')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(1./freq_oeyvind_15kn,FFT_oeyvind_15kn,'-',label='method 2',color='darkorange')
ax2.set_xlabel('period[s]' )
ax2.set_ylabel(r'$f^{SV}[N]$')
ax2.legend(loc='best')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(ppfft, format='pdf')
plt.close(fig)

fig = plt.figure('fft ')
plt.suptitle('FFT slowly varying drift force: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(sp
ax1 = fig.add_subplot(2,1,1)
ax1.plot(1./freq_realisation1_15kn ,FFT_realisation1_15kn, '-',label='realisation 1')
ax1.set_xlabel('period[s]')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax1.legend(loc='best')
ax2 = fig.add_subplot(2,1,2)
ax2.plot(1./freq_oeyvind_15kn,FFT_oeyvind_15kn,'-',label='method 2',color='darkorange')
ax2.set_xlabel('period[s]' )
ax2.set_ylabel(r'$f^{SV}[N]$')
ax2.legend(loc='best')
```

```
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
fig.savefig(ppfft, format='pdf')
plt.close(fig)

fig = plt.figure('FaltinsenLoeken histogram')
plt.suptitle('Timeseries Faltinsen/Loeken: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(spectr
ax1 = fig.add_subplot(2,1,1)
ax1.plot(JONSWAP_time_loeken,JONSWAP_Fx_loeken_15kn,'-',label='method2 Loeken/Faltinsen')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,label='histogram meth
ax2.set_xlabel(r'$f^{SV}[N]$')
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
plt.savefig(ppmethod2, format='pdf')
plt.close(fig)

fig = plt.figure('Oeyvind histogram')
plt.suptitle('Timeseries Oeyvind: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(spectrum_param
ax1 = fig.add_subplot(2,1,1)
ax1.plot(JONSWAP_time_oeyvind,JONSWAP_Fx_oeyvind_15kn,'-',label='method2 Oeyvind')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,label='histogram me
ax2.set_xlabel(r'$f^{SV}[N]$')
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
plt.savefig(ppmethod2, format='pdf')
plt.close(fig)

fig = plt.figure('FaltinsenLoeken histogram long')
plt.suptitle('Timeseries Faltinsen/Loeken long: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,JONSWAP_Fx_long_loeken_15kn,'-',label='method2 Loeken/Faltinsen')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_loeken_long_15kn,hist_loeken_long_15kn,width=width_histogram_loeken_long_
ax2.set_xlabel(r'$f^{SV}[N]$')
ax2.set_ylabel('probability')
ax1.grid()
```

```
ax2.grid()
ax1.legend()
ax2.legend()
plt.savefig(ppmethod2, format='pdf')
plt.close(fig)

fig = plt.figure('Oeyvind histogram long')
plt.suptitle('Timeseries Oeyvind long: '+'Tp'+str(spectrum_parameter[0])+'Hs'+str(spectrum_
ax1 = fig.add_subplot(2,1,1)
ax1.plot(time,JONSWAP_Fx_long_oeyvind_15kn,'-',label='method2 Oeyvind')
ax1.set_ylabel(r'$f^{SV}[N]$')
ax2 = fig.add_subplot(2,1,2)
ax2.bar(bin_center_oeyvind_long_15kn,hist_oeyvind_long_15kn,width=width_histogram_oeyvind_lo
ax2.set_xlabel(r'$f^{SV}[N]$' )
ax2.set_ylabel('probability')
ax1.grid()
ax2.grid()
ax1.legend()
ax2.legend()
plt.savefig(ppmethod2, format='pdf')
plt.close(fig)

plt.figure(22)
plt.grid()
plt.title('Histogram Faltinsen/Loeken v Oeyvind: ' +'Tp'+str(spectrum_parameter[0])+'Hs'+str
plt.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,width=width_histogram_
plt.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,width=width_histogram
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$' )
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(22)

plt.figure(33)
plt.grid()
plt.title('Histogram Newman-approx v Faltinsen/Loeken: ' +'Tp'+str(spectrum_parameter[0])+'H
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=width_histogram_15kn,label='Newman-approx')
plt.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,width=width_histogram_
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$' )
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(33)

plt.figure(21)
plt.grid()
plt.title('Histogram Newman-approx v Oeyvind: ' +'Tp'+str(spectrum_parameter[0])+'Hs'+str(sp
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=width_histogram_15kn,label='Newman-approx')
plt.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,width=width_histogram
```

```
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$' )
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(21)

plt.figure(332)
plt.grid()
plt.title('Histogram Newman-approx v Faltinsen/Loeken v Oeyvind:' +'Tp'+str(spectrum_parame
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=width_histogram_15kn,label='Newman-approx')
plt.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,width=width_histogram_
plt.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,width=width_histogram
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$' )
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(332)

plt.figure(33)
plt.grid()
plt.title('Histogram Newman-approx various realisations:' +'Tp'+str(spectrum_parameter[0])+
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=width_histogram_15kn,label='realisation 1')
plt.bar(bin_center_Fx2_15kn,hist_Fx2_15kn,width=width_histogram_15kn,label='realisation 2')
plt.bar(bin_center_Fx3_15kn,hist_Fx3_15kn,width=width_histogram_15kn,label='realisation 3')
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$' )
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(33)


plt.figure(122)
plt.grid()
plt.title('Histogram Faltinsen/Loeken v Oeyvind:' +'Tp'+str(spectrum_parameter[0])+'Hs'+str
plt.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,width=0.6*width_histogr
plt.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,width=0.6*width_histo
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$' )
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(122)

plt.figure(133)
plt.grid()
plt.title('Histogram Newman-approx v Faltinsen/Loeken:' +'Tp'+str(spectrum_parameter[0])+'H
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=0.6*width_histogram_15kn,label='Newman-appr
plt.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,width=0.6*width_histogr
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$' )
```

```
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(133)

plt.figure(121)
plt.grid()
plt.title('Histogram␣Newman-approx␣v␣Oeyvind:' +'Tp'+str(spectrum_parameter[0])+'Hs'+str(sp
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=0.6*width_histogram_15kn,label='Newman-appr
plt.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,width=0.6*width_histo
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$')
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(121)

plt.figure(1332)
plt.grid()
plt.title('Histogram␣Newman-approx␣v␣Faltinsen/Loeken␣v␣Oeyvind:' +'Tp'+str(spectrum_parame
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=0.6*width_histogram_15kn,label='Newman-appr
plt.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,width=0.6*width_histogr
plt.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,width=0.6*width_histo
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$')
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(1332)

plt.figure(133)
plt.grid()
plt.title('Histogram␣Newman-approx␣various␣realisations:' +'Tp'+str(spectrum_parameter[0])+
plt.bar(bin_center_Fx1_15kn[1:],hist_Fx1_15kn[1:],width=width_histogram_15kn,label='realisat
plt.bar(bin_center_Fx2_15kn[1:],hist_Fx2_15kn[1:],width=width_histogram_15kn,bottom=hist_Fx1
plt.bar(bin_center_Fx3_15kn[1:],hist_Fx3_15kn[1:],width=width_histogram_15kn,bottom=hist_Fx1
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$')
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(133)

plt.figure(233)
plt.grid()
plt.title('Histogram␣Newman-approx␣various␣realisations:' +'Tp'+str(spectrum_parameter[0])+
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=width_histogram_15kn,label='realisation␣1',
plt.bar(bin_center_Fx2_15kn,hist_Fx2_15kn,width=width_histogram_15kn,label='realisation␣2',
plt.bar(bin_center_Fx3_15kn,hist_Fx3_15kn,width=width_histogram_15kn,label='realisation␣3',
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$')
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
```

```
plt.close(233)

plt.figure(522)
plt.grid()
plt.title('Histogram␣Faltinsen/Loeken␣v␣Oeyvind:' +'Tp'+str(spectrum_parameter[0])+'Hs'+str
plt.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,width=width_histogram_
plt.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,alpha=0.4,width=widt
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$')
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(522)

plt.figure(335)
plt.grid()
plt.title('Histogram␣Newman-approx␣v␣Faltinsen/Loeken:' +'Tp'+str(spectrum_parameter[0])+'H
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=width_histogram_15kn,label='Newman-approx')
plt.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,alpha=0.4,width=width_
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$')
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(335)

plt.figure(215)
plt.grid()
plt.title('Histogram␣Newman-approx␣v␣Oeyvind:' +'Tp'+str(spectrum_parameter[0])+'Hs'+str(sp
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=width_histogram_15kn,label='Newman-approx')
plt.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,alpha=0.4,width=widt
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$')
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(215)

plt.figure(3532)
plt.grid()
plt.title('Histogram␣Newman-approx␣v␣Faltinsen/Loeken␣v␣Oeyvind:' +'Tp'+str(spectrum_parame
plt.bar(bin_center_Fx1_15kn,hist_Fx1_15kn,width=width_histogram_15kn,label='Newman-approx')
plt.bar(bin_center_Fx_JONSWAP_loeken_15kn,hist_Fx_JONSWAP_loeken_15kn,alpha=0.4,width=width_
plt.bar(bin_center_Fx_JONSWAP_oeyvind_15kn,hist_Fx_JONSWAP_oeyvind_15kn,alpha=0.4,width=widt
plt.ylabel('probability')
plt.xlabel(r'$f^{SV}[N]$')
plt.legend(loc='best')
plt.savefig(ppmethod2, format='pdf')
plt.close(3532)
```

```
pp.close()
ppp.close()
ppfft.close()
ppmethod2.close()
workbook.close()
print('...plot:␣DONE')
```

G Python: irregular sea

```
import numpy as np
import os
import matplotlib.pyplot as plt
import scipy.signal as sp
import scipy.fft as sf
import xlsxwriter
from scipy.signal import butter, lfilter, freqz, filtfilt


def make_empty_list():
    array = np.ndarray([])
    array = np.delete(array,0)
    return array
def JONSWAP(Tp, Hs,array_length,):
    array = np.linspace(3.5,18.5,num=array_length)
    increment = 15./array_length
    omega_p = 2*np.pi/Tp
    omega_array = 2*np.pi/array
    Sj_array = np.ndarray([len(array)])
    gamma = 3.3
    Ay = 1-0.287*np.log(gamma)
    sigma_a = 0.07
    sigma_b = 0.09
    for i,w in enumerate(omega_array):
        S_PM = 0.3125*Hs**(2)*omega_p**(4)*w**(-5)*np.exp(-1.25*(w/omega_p)**(-4))
        if w <= omega_p:
            S_J = Ay*S_PM*gamma**(np.exp(-0.5*((w-omega_p)/(sigma_a*omega_p))**2))
        elif w > omega_p:
            S_J = Ay*S_PM*gamma**(np.exp(-0.5*((w-omega_p)/(sigma_b*omega_p))**2))
        Sj_array[i] = S_J
    return array,Sj_array
def pqFormel(p,q):
    x1 = -p/2 + np.sqrt(((p**2)/4)-q)
    x2 = -p / 2 - np.sqrt(((p ** 2) / 4) - q)
    return x1,x2
def combining_energy(period,energy,period2,amplitude2):
    for i, ii in enumerate(period2):
        inside = False
        for j, jj in enumerate(period):
            if round(ii * 1000) == round(jj * 1000):
                energy[j] = energy[j] + (0.5 * amplitude2[i] ** 2)
                inside = True
        if inside == False:
            period = np.append(period, period2[i])
            energy = np.append(energy, (0.5 * amplitude2[i] ** 2))
    return period,energy
def bin_center(binEdges):
```

```
    bincenters = 0.5 * (binEdges[1:] + binEdges[:-1])
    return bincenters
def bpass_gaussian(xsig, dt, flow, fhigh, M = 50):
    xsig_fft = np.fft.fft(xsig)
    fvec = np.fft.fftfreq(len(xsig), d=dt)
    Nsamples = len(fvec)
    # Create Gaussian window filter:
#    M = 50  # Total number of samples in Gaussian ramp function
    gausswin = sp.windows.gaussian(M, std=7)
    lramp = np.array(gausswin[:M // 2])  # Left ramp function (left half of Gaussian)
    rramp = np.array(gausswin[M // 2:])  # Right ramp function (right half of Gaussian)
    i1 = np.argwhere(np.abs(fvec[:Nsamples // 2]) > flow)[0][0]  # Index of low cut frequen
    i2 = np.argwhere(np.abs(fvec[:Nsamples // 2]) < fhigh)[-1][0]  # Index of high cut freq
    lwin = np.zeros(np.max([i1 - M // 2, 0]))  # Left zero-padding
    rwin = np.zeros(np.min([Nsamples // 2 - (i2 + M // 2), Nsamples // 2 - 1]))
# Right zero-padding
    oneband = np.ones(np.max([Nsamples // 2 - len(lwin) - len(rwin) - len(lramp) - len(rram
# Unity pass-band
    filter = np.concatenate((lwin, lramp, oneband, rramp, rwin))  # Combined window functio
    # Filter signal by multiplication with window function in frequency domain
    # Inverse FFT to obtain filtered time-series
    xsig_bp = np.fft.ifft(xsig_fft * np.concatenate((filter, filter[::-1]))).real
### gives warning
    return xsig_bp, (fvec, filter)
def homemade_Gauss(array: np.ndarray,lowperiod,highperiod ,dt):
    lcut = 1./highperiod
    hcut = 1./lowperiod
    fft = sf.fft(array)
#    fft = np.abs(fft / len(fft))
    n = fft.size
    freq = sf.fftfreq(n, dt)
    for i in range(len(fft)):
        if freq[i]<=lcut:
            fft[i] = 0.0
        elif freq[i] >= hcut:
            fft[i] = 0.0
    new_array = sf.ifft(fft)
    return new_array.real
def find_mean_period_old(array: np.ndarray,time: np.ndarray):
    array_mean = np.mean(array)
    index = np.argwhere(np.diff(np.sign(array-array_mean))>0)
    index = index.transpose()
    m = (array[index+1]-array[index])/(time[index+1]-time[index])
    m=m[0]
    array_index =array[index]
    array_index = array_index[0]
    time_index = time[index]
    time_index = time_index[0]
    time_cor = (array_index-array_mean)/m
```

```
    time_index = time_index+time_cor
    mean_period = np.mean(np.diff(time_index))
    return mean_period

def find_irregularSeastate_AmplitudePeriod_Oeyvind(array: np.ndarray,timearray: np.ndarray)
    array_mean = np.mean(array)
    index = np.argwhere(np.diff(np.sign(array-array_mean)) > 0)
    index = index.transpose()
    index_neg = np.argwhere(np.diff(np.sign(array)) < 0)
    index_neg = index_neg.transpose()
    period_array = make_empty_list()
    amplitude_array = make_empty_list()
    index = index[0]
    for i in range(1,len(index)):
        period = timearray[index[i]]-timearray[index[i-1]]

        temp_array = array[index[i-1]:index[i]]
        amplitude = 0.5*(max(temp_array)-min(temp_array))
        amplitude_array = np.append(amplitude_array, amplitude)
        period_array = np.append(period_array, period)

    return period_array,amplitude_array

def find_irregularSeastate_AmplitudePeriod_unsorted(array: np.ndarray,timearray: np.ndarray
    array_mean = np.mean(array)
    index_up = np.argwhere(np.diff(np.sign(array)) > 0)
    index_up = index_up.transpose()
    index_neg = np.argwhere(np.diff(np.sign(array)) < 0)
    index_neg = index_neg.transpose()
    period_array = make_empty_list()
    amplitude_array = make_empty_list()
    index_up = index_up[0]
    index_neg = index_neg[0]
    if len(index_up) < len(index_neg):
        index_neg = np.delete(index_neg,0)
    if len(index_up) > len(index_neg):
        index_up = np.delete(index_up, 0)
    if index_up[0]< index_neg[0]:
        index1 = index_up
        index2 = index_neg
    if index_up[0]> index_neg[0]:
        index2 = index_up
        index1 = index_neg
    #print(timearray)

    for i in range(0,len(index1)):
        period =  (timearray[index2[i]]-timearray[index1[i]])*2.
        temp_array = array[index1[i]:index2[i]]
        amplitude = (max(temp_array)-min(temp_array))
```

```
        amplitude_array = np.append(amplitude_array, amplitude)
        period_array = np.append(period_array, period)

    index1 = np.delete(index1,0)
    index2 = np.delete(index2,-1)

    for i in range(0,len(index2)):
        period = 2* (timearray[index1[i]]-timearray[index2[i]])
        temp_array = array[index2[i]:index1[i]]
        amplitude = (max(temp_array)-min(temp_array))
        amplitude_array = np.append(amplitude_array, amplitude)
        period_array = np.append(period_array, period)

    return period_array,amplitude_array


def make_empty_list():
    array = np.ndarray([])
    array = np.delete(array,0)
    return array
def butter_bandpass(data, lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b,a = butter(order, [low, high], btype='band')
    y = sp.filtfilt(b,a, data)
    return y
def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    sos = butter(order, [low, high],output='sos', btype='band')
    y = sp.sosfiltfilt(sos, data)
    return y
def butter_highpass(cutoff, fs, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='high', analog=False)
    return b, a
def butter_highpass_filter(data, cutoff, fs, order=5):
    b, a = butter_highpass(cutoff, fs, order=order)
    y = filtfilt(b, a, data)
    return y
def OQUS_interval(intStart, intEnd,time_oqus):
    oqus_intStart = round(intStart*5.)/5.       # intSTart in seconds
    oqus_intEnd = round(intEnd*5.)/5.
    oqus_intStart1 = np.argwhere(time_oqus<oqus_intStart)
    oqus_intStart = oqus_intStart1[-1]
    oqus_intStart = oqus_intStart[0]
```

```
    oqus_intEnd1 = np.argwhere(time_oqus>oqus_intEnd)
    oqus_intEnd = oqus_intEnd1[0]
    oqus_intEnd = oqus_intEnd[0]
    return [oqus_intStart, oqus_intEnd]
def find_mean_period(array: np.ndarray,time: np.ndarray):
    array_mean = np.mean(array)
    index = np.argwhere(np.diff(np.sign(array-array_mean))>0)
    index = index.transpose()
    mean_period = np.mean(np.diff(time[index]))
    T_stddev = np.std(np.diff(time[index]))
    return mean_period
def butter_lowpass_filter(data, cutoff, fs, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = filtfilt(b, a, data)
    return y
def find_amplitude(array: np.ndarray,distance = None):
    array = array- np.mean(array)
    amplitude_max, _ = sp.find_peaks(array,distance=distance,  height=(np.max(array)*0.2, N
    amplitude_min, _ = sp.find_peaks(-array,distance=distance,  height=(np.max(-array)*0.2,
    amplitude = np.abs(np.mean(slider_averaging(array[amplitude_max]))) + np.abs(np.mean(sli
    return amplitude/2.
def find_calc_interval(carriage_vel: np.ndarray,carriage_pos: np.ndarray):
    carrvel_lp = butter_lowpass_filter(carriage_vel, 1., 200.)
    acc_magnitude = (np.diff(carrvel_lp) / 0.005)
    iacc_peaks, _ = sp.find_peaks(acc_magnitude)
    for i in iacc_peaks:
        index = np.argwhere(iacc_peaks == i)
        if np.abs(acc_magnitude)[i] > 0.04 or np.abs(acc_magnitude)[i] < 0.004:
            iacc_peaks = np.delete(iacc_peaks, index)
        elif carriage_vel[i] < 1.0:
            iacc_peaks = np.delete(iacc_peaks, index)

    peaks_old = np.array([], dtype='int')
    peaks_new = np.array([], dtype='int')
    for i in iacc_peaks:

        if carriage_pos[i] <= 175.:
            peaks_old = np.append(peaks_old, i)
        elif carriage_pos[i] > 175.:
            peaks_new = np.append(peaks_new, i)
    interval = np.array([iacc_peaks[0],iacc_peaks[-1]], dtype='int')
    interval_old = np.array([peaks_old[0],peaks_old[-1]], dtype='int')
    interval_new = np.array([peaks_new[0],peaks_new[-1]], dtype='int')
    return interval,interval_old,interval_new
def set_interval(array: np.ndarray,intervalstart,intervalend):
    array_mean = np.mean(array)
    index = np.argwhere(np.diff(np.sign(array-array_mean))>0)
```

```
    index = index.transpose()
    index= index[0]
    checkstart = abs(index-intervalstart)
    checkend = abs(index-intervalend)
    min = np.argmin(checkstart)
    start = index[min]
    max = np.argmin(checkend)
    end = index[max]


    return [start,end]
def FFT(array: np.ndarray,dt):
    array_fft = sf.fft(array)
    array_fft = np.abs(array_fft / len(array_fft))
    n = array_fft.size
    freq_plot = sf.fftfreq(n, dt)
    array_fft = array_fft[:int((len(array_fft) - 1) / 2)]
    freq_plot = freq_plot[:int((len(freq_plot) - 1) / 2)]
    return array_fft,freq_plot

def moving_average(x, w):
 #assert np.issubdtype(w, np.integer) and w > 0
 return np.convolve(x, np.ones(w), 'valid') / w


def find_mean_from_peaks(array: np.ndarray,distance = None):
    meanArray = np.mean(array)
    array = array-meanArray
    Fx_interval_amplitude_max, _ = sp.find_peaks(array,distance=distance,  height=(0.0, Non
    Fx_interval_amplitude_min, _ = sp.find_peaks(-array,distance=distance,  height=(0.0, No
    mean = 0.5 * (np.mean(slider_averaging(array[Fx_interval_amplitude_max])) + np.mean(slid
    return mean, Fx_interval_amplitude_min,Fx_interval_amplitude_max


def slider_averaging(array: np.ndarray):
    new_array = np.array([])
    N = len(array)
    for i in range(N):
        if i>1 and i<(N-2):
            value = (array[i]*5+array[i-1]*3+array[i+1]*3+array[i-2]*1+array[i+2]*1)/13.
            new_array = np.append(new_array,value)
        elif i==0:
            value = (array[i] * 5  + array[i + 1] * 3 + array[i + 2] * 1) / 9.
            new_array = np.append(new_array, value)
        elif i==1:
            value = (array[i] * 5  + array[i + 1] * 3 + array[i - 1] * 3 + array[i + 2] * 1
            new_array = np.append(new_array, value)
        elif i==N-1:
            value = (array[i] * 5  + array[i - 1] * 3  + array[i - 2] * 1) / 9.
```

```
            new_array = np.append(new_array, value)
        elif i == N-2:
            value = (array[i] * 5 + array[i - 1] * 3+ array[i + 1] * 3  + array[i - 2] * 1)
            new_array = np.append(new_array, value)
    return new_array

datapath = "./irregularSea/"
## data from file
filename = "E7700.npz"
run_name = filename.translate({ord(i): None for i in 'E.npz '})

datafile = os.path.join(datapath,filename)
## load file
df = np.load(datafile,allow_pickle=True)
data = df['data']
keys = df['keys']
data_array = data.item()
datadict = {key: value for key, value in zip(keys, data_array)}

wp3_tot = data_array[datadict['WP3_CAR']]
wp1 = data_array[datadict['WP1_BM']]
wpUS =data_array[datadict['WP_US']]
Fx = data_array[datadict['FX_CENTER']]
time = data_array[datadict['DAQ_time_stamps']]
time_oqus = data_array[datadict['DAQ_time_stamps_SR2']]
zpos = data_array[datadict['zpos1']]
pitch = data_array[datadict['pitch1']]
roll = data_array[datadict['roll1']]
calmwater_res_exp = 31.182
calmwater_res_holtrop = 18.095


# take out calmwater resistance
Fx = Fx+calmwater_res_exp

## set calculation interval
carriage_vel = data_array[datadict['speed_carriage']]
carriage_pos = data_array[datadict['pos_carriage_analog']]
interval,interval_old,interval_new = find_calc_interval(carriage_vel,carriage_pos)
interval_7700 = np.array([[40040,57842],[70170,87997],[99464,117508],[130596,148644],[16184
interval_7710 = np.array([[36699,55439],[70074,87224],[100184,117527],[130652,147556],[1605
if filename=="E7700.npz":
    interval = interval_7700
    Hs = 1.5
    Tp = 7.5
elif filename=="E7710.npz":
    interval = interval_7710
    Hs = 2.5
    Tp = 8.5
```

```
# Fx interval
Fx_interval1 = Fx[interval[0][0]:interval[0][1]]
Fx_interval2 = Fx[interval[1][0]:interval[1][1]]
Fx_interval3 = Fx[interval[2][0]:interval[2][1]]
Fx_interval4 = Fx[interval[3][0]:interval[3][1]]
Fx_interval5 = Fx[interval[4][0]:interval[4][1]]

# time interval
time_interval1 = time[interval[0][0]:interval[0][1]]
time_interval2 = time[interval[1][0]:interval[1][1]]
time_interval3 = time[interval[2][0]:interval[2][1]]
time_interval4 = time[interval[3][0]:interval[3][1]]
time_interval5 = time[interval[4][0]:interval[4][1]]

# wp3 interval
wp3_interval1 = wp3_tot[interval[0][0]:interval[0][1]]
wp3_interval2 = wp3_tot[interval[1][0]:interval[1][1]]
wp3_interval3 = wp3_tot[interval[2][0]:interval[2][1]]
wp3_interval4 = wp3_tot[interval[3][0]:interval[3][1]]
wp3_interval5 = wp3_tot[interval[4][0]:interval[4][1]]

# OQUS interval
oqus_interval1 = OQUS_interval(time_interval1[0],time_interval1[-1],time_oqus)
time_oqus_interval1 = time_oqus[oqus_interval1[0]:oqus_interval1[-1]]
zpos_interval1 = zpos[oqus_interval1[0]:oqus_interval1[-1]]
pitch_interval1 = pitch[oqus_interval1[0]:oqus_interval1[-1]]

oqus_interval2 = OQUS_interval(time_interval2[0],time_interval2[-1],time_oqus)
time_oqus_interval2 = time_oqus[oqus_interval2[0]:oqus_interval2[-1]]
zpos_interval2 = zpos[oqus_interval2[0]:oqus_interval2[-1]]
pitch_interval2 = pitch[oqus_interval2[0]:oqus_interval2[-1]]

oqus_interval3 = OQUS_interval(time_interval3[0],time_interval3[-1],time_oqus)
time_oqus_interval3 = time_oqus[oqus_interval3[0]:oqus_interval3[-1]]
zpos_interval3 = zpos[oqus_interval3[0]:oqus_interval3[-1]]
pitch_interval3 = pitch[oqus_interval3[0]:oqus_interval3[-1]]

oqus_interval4 = OQUS_interval(time_interval4[0],time_interval4[-1],time_oqus)
time_oqus_interval4 = time_oqus[oqus_interval4[0]:oqus_interval4[-1]]
zpos_interval4 = zpos[oqus_interval4[0]:oqus_interval4[-1]]
pitch_interval4 = pitch[oqus_interval4[0]:oqus_interval4[-1]]

oqus_interval5 = OQUS_interval(time_interval5[0],time_interval5[-1],time_oqus)
time_oqus_interval5 = time_oqus[oqus_interval5[0]:oqus_interval5[-1]]
zpos_interval5 = zpos[oqus_interval5[0]:oqus_interval5[-1]]
pitch_interval5 = pitch[oqus_interval5[0]:oqus_interval5[-1]]

# FFT
Fx_fft1, Fx_freq1 = FFT(Fx_interval1,0.005)
```

```
Fx_fft2, Fx_freq2 = FFT(Fx_interval2,0.005)
Fx_fft3, Fx_freq3 = FFT(Fx_interval3,0.005)
Fx_fft4, Fx_freq4 = FFT(Fx_interval4,0.005)
Fx_fft5, Fx_freq5 = FFT(Fx_interval5,0.005)


# spectrum
spectrum_period,spectrum_energy = JONSWAP(Tp,Hs,50)
spectrum_omega = 2*np.pi/spectrum_period
spectrum_ampl = np.sqrt(2*spectrum_energy)

wp3_period11, wp3_amplitude11 = find_irregularSeastate_AmplitudePeriod_unsorted(wp3_interva
wp3_period22, wp3_amplitude22 = find_irregularSeastate_AmplitudePeriod_unsorted(wp3_interva
wp3_period33, wp3_amplitude33 = find_irregularSeastate_AmplitudePeriod_unsorted(wp3_interva
wp3_period44, wp3_amplitude44 = find_irregularSeastate_AmplitudePeriod_unsorted(wp3_interva
wp3_period55, wp3_amplitude55 = find_irregularSeastate_AmplitudePeriod_unsorted(wp3_interva

wp3_period_loeken=wp3_period11
wp3_amplitude_loeken=wp3_amplitude11
wp3_energy_loeken = 0.5*wp3_amplitude_loeken**2
wp3_period_loeken,wp3_energy_loeken = combining_energy(wp3_period_loeken,wp3_energy_loeken,
wp3_period_loeken,wp3_energy_loeken = combining_energy(wp3_period_loeken,wp3_energy_loeken,
wp3_period_loeken,wp3_energy_loeken = combining_energy(wp3_period_loeken,wp3_energy_loeken,
wp3_period_loeken,wp3_energy_loeken = combining_energy(wp3_period_loeken,wp3_energy_loeken,

wp3_amplitude_loeken = [x for _, x in sorted(zip(wp3_period_loeken, wp3_amplitude_loeken))]
wp3_amplitude_loeken = np.array(wp3_amplitude_loeken)
wp3_period_loeken = sorted(wp3_period_loeken)
wp3_period_loeken = np.array(wp3_period_loeken)

wp3_amplitude11 = [x for _, x in sorted(zip(wp3_period11, wp3_amplitude11))]
wp3_amplitude11 = np.array(wp3_amplitude11)
wp3_period11 = sorted(wp3_period11)
wp3_period11 = np.array(wp3_period11)

wp3_amplitude22 = [x for _, x in sorted(zip(wp3_period22, wp3_amplitude22))]
wp3_amplitude22 = np.array(wp3_amplitude22)
wp3_period22 = sorted(wp3_period22)
wp3_period22 = np.array(wp3_period22)

wp3_amplitude33 = [x for _, x in sorted(zip(wp3_period33, wp3_amplitude33))]
wp3_amplitude33 = np.array(wp3_amplitude33)
wp3_period33 = sorted(wp3_period33)
wp3_period33 = np.array(wp3_period33)

wp3_amplitude4 = [x for _, x in sorted(zip(wp3_period44, wp3_amplitude44))]
wp3_amplitude44 = np.array(wp3_amplitude44)
wp3_period44 = sorted(wp3_period44)
wp3_period44 = np.array(wp3_period44)
```

```
wp3_amplitude55 = [x for _, x in sorted(zip(wp3_period55, wp3_amplitude55))]
wp3_amplitude55 = np.array(wp3_amplitude55)
wp3_period55 = sorted(wp3_period55)
wp3_period55 = np.array(wp3_period55)

#sort
wp3_period1, wp3_amplitude1 = find_irregularSeastate_AmplitudePeriod_Oeyvind(wp3_interval1,
wp3_period2, wp3_amplitude2 = find_irregularSeastate_AmplitudePeriod_Oeyvind(wp3_interval2,
wp3_period3, wp3_amplitude3 = find_irregularSeastate_AmplitudePeriod_Oeyvind(wp3_interval3,
wp3_period4, wp3_amplitude4 = find_irregularSeastate_AmplitudePeriod_Oeyvind(wp3_interval4,
wp3_period5, wp3_amplitude5 = find_irregularSeastate_AmplitudePeriod_Oeyvind(wp3_interval5,
wp3_period=wp3_period1
wp3_amplitude=wp3_amplitude1
wp3_energy = 0.5*wp3_amplitude**2
print(len(wp3_period1),len(wp3_period2),len(wp3_period3),len(wp3_period4),len(wp3_period5))


wp3_period,wp3_energy = combining_energy(wp3_period,wp3_energy,wp3_period2,wp3_amplitude2)
wp3_period,wp3_energy = combining_energy(wp3_period,wp3_energy,wp3_period3,wp3_amplitude3)
wp3_period,wp3_energy = combining_energy(wp3_period,wp3_energy,wp3_period4,wp3_amplitude4)
wp3_period,wp3_energy = combining_energy(wp3_period,wp3_energy,wp3_period5,wp3_amplitude5)




wp3_energyALL = [x for _, x in sorted(zip(wp3_period, wp3_energy))]
wp3_energyALL = np.array(wp3_energyALL)
wp3_periodALL = sorted(wp3_period)
wp3_periodALL = np.array(wp3_periodALL)

wp3_amplitude1 = [x for _, x in sorted(zip(wp3_period1, wp3_amplitude1))]
wp3_amplitude1 = np.array(wp3_amplitude1)
wp3_period1 = sorted(wp3_period1)
wp3_period1 = np.array(wp3_period1)

wp3_amplitude2 = [x for _, x in sorted(zip(wp3_period2, wp3_amplitude2))]
wp3_amplitude2 = np.array(wp3_amplitude2)
wp3_period2 = sorted(wp3_period2)
wp3_period2 = np.array(wp3_period2)

wp3_amplitude3 = [x for _, x in sorted(zip(wp3_period3, wp3_amplitude3))]
wp3_amplitude3 = np.array(wp3_amplitude3)
wp3_period3 = sorted(wp3_period3)
wp3_period3 = np.array(wp3_period3)

wp3_amplitude4 = [x for _, x in sorted(zip(wp3_period4, wp3_amplitude4))]
wp3_amplitude4 = np.array(wp3_amplitude4)
wp3_period4 = sorted(wp3_period4)
```

```
wp3_period4 = np.array(wp3_period4)

wp3_amplitude5 = [x for _, x in sorted(zip(wp3_period5, wp3_amplitude5))]
wp3_amplitude5 = np.array(wp3_amplitude5)
wp3_period5 = sorted(wp3_period5)
wp3_period5 = np.array(wp3_period5)


# wave period from encounter period
omega_e = 2*np.pi/wp3_periodALL
incident_period_exp = np.ndarray([len(omega_e)])
for i,we in enumerate(omega_e):
    temp_w = 0.
    temp_w1,temp_w2 = pqFormel(9.81/1.363,-1.*we*9.81/1.363)
    if temp_w1<0:
        temp_w = temp_w2
    elif temp_w2<0:
        temp_w = temp_w1
    incident_period_exp[i] = 2*np.pi/temp_w

omega_e_loeken = 2*np.pi/wp3_period_loeken
incident_period_exp_loeken = np.ndarray([len(omega_e_loeken)])
for i,we in enumerate(omega_e_loeken):
    temp_w = 0.
    temp_w1,temp_w2 = pqFormel(9.81/1.363,-1.*we*9.81/1.363)
    if temp_w1<0:
        temp_w = temp_w2
    elif temp_w2<0:
        temp_w = temp_w1
    incident_period_exp_loeken[i] = 2*np.pi/temp_w


# historgram
amountBINS = 500

hist_interval1, bin_edges_interval1 = np.histogram(Fx_interval1*32.**3., range=(min(Fx_inte
bin_center_interval1 = bin_center(bin_edges_interval1)
width_histogram_interval1 = 0.8*(bin_edges_interval1[0]-bin_edges_interval1[1])
hist_interval2, bin_edges_interval2 = np.histogram(Fx_interval2*32.**3., range=(min(Fx_inte
bin_center_interval2 = bin_center(bin_edges_interval2)
hist_interval3, bin_edges_interval3 = np.histogram(Fx_interval3*32.**3., range=(min(Fx_inte
bin_center_interval3 = bin_center(bin_edges_interval3)
hist_interval4, bin_edges_interval4 = np.histogram(Fx_interval4*32.**3., range=(min(Fx_inte
bin_center_interval4 = bin_center(bin_edges_interval4)
hist_interval5, bin_edges_interval5 = np.histogram(Fx_interval5*32.**3., range=(min(Fx_inte
bin_center_interval5 = bin_center(bin_edges_interval5)
hist_interval = hist_interval1+hist_interval2+hist_interval3+hist_interval4+hist_interval5
hist_interval = hist_interval/sum(hist_interval)
```

```
hist_interval1_abs, bin_edges_interval1_abs = np.histogram(abs(Fx_interval1*32.**3.), range
bin_center_interval1_abs = bin_center(bin_edges_interval1_abs)
width_histogram_interval1_abs = 0.8*(bin_edges_interval1_abs[0]-bin_edges_interval1_abs[1])
hist_interval2_abs, bin_edges_interval2_abs = np.histogram(abs(Fx_interval2*32.**3.), range
hist_interval3_abs, bin_edges_interval3_abs = np.histogram(abs(Fx_interval3*32.**3.), range
hist_interval4_abs, bin_edges_interval4_abs = np.histogram(abs(Fx_interval4*32.**3.), range
hist_interval5_abs, bin_edges_interval5_abs = np.histogram(abs(Fx_interval5*32.**3.), range
hist_interval_abs = hist_interval1_abs+hist_interval2_abs+hist_interval3_abs+hist_interval4_
hist_interval_abs = hist_interval_abs/sum(hist_interval_abs)



# output
print('mean interval 1',np.mean(Fx_interval1))
print('mean interval 2',np.mean(Fx_interval2))
print('mean interval 3',np.mean(Fx_interval3))
print('mean interval 4',np.mean(Fx_interval4))
print('mean interval 5',np.mean(Fx_interval5))
for i in range(0,100):
    print(i,wp3_period4[i],wp3_period44[i])


#plot
print('...plot...')
plt.figure('total interval')
plt.grid()
plt.plot(time,Fx, label='total resistance')
plt.xlabel('time[s]')
plt.ylabel('total resistance[N]')
plt.legend(loc='best')



plt.figure('spectrum')
plt.grid()
plt.plot(incident_period_exp*5.657,wp3_energyALL*32**2, label='wp3 all')
plt.plot(incident_period_exp[4:-4]*5.657,moving_average(wp3_energyALL*32**2,9), label='wp3
plt.plot(spectrum_period,spectrum_energy, label='JONSWAP')
plt.xlabel('period[s]')
plt.ylabel('S(T)')
plt.legend(loc='best')

plt.figure('spectrum moving average')
plt.grid()
plt.plot(moving_average(incident_period_exp*5.657,10),moving_average(wp3_energyALL*32**2,10
plt.plot(moving_average(incident_period_exp_loeken*5.657,10),moving_average(wp3_energy_loek
plt.plot(spectrum_period,spectrum_energy, label='JONSWAP')
plt.xlabel('period[s]')
```

```
plt.ylabel('wave amplitude')
plt.legend(loc='best')

plt.figure('spectrum interval1 move average')
plt.grid()
plt.plot(wp3_energyALL*32**2, label='wp3 interval1 oeyvind ')
plt.plot(moving_average(wp3_energyALL*32**2,10), label='wp3 interval1 move average')
plt.xlabel('period[s]')
plt.ylabel('spectrum energy')
plt.legend(loc='best')

plt.figure('interval Loeken')
plt.grid()
plt.plot(wp3_period11,wp3_amplitude11, label='wp3 interval1  ')
plt.plot(wp3_period22,wp3_amplitude22, label='wp3 interval2  ')
plt.plot(wp3_period33,wp3_amplitude33, label='wp3 interval3  ')
plt.plot(wp3_period44,wp3_amplitude44, label='wp3 interval4  ')
plt.plot(wp3_period55,wp3_amplitude55, label='wp3 interval5  ')
plt.xlabel('period[s]')
plt.ylabel('amplitude')
plt.legend(loc='best')

plt.figure('interval Oeyvind')
plt.grid()
plt.plot(wp3_period1,wp3_amplitude1, label='wp3 interval1  ')
plt.plot(wp3_period2,wp3_amplitude2, label='wp3 interval2  ')
plt.plot(wp3_period3,wp3_amplitude3, label='wp3 interval3  ')
plt.plot(wp3_period4,wp3_amplitude4, label='wp3 interval4  ')
plt.plot(wp3_period5,wp3_amplitude5, label='wp3 interval5  ')
plt.xlabel('period[s]')
plt.ylabel('amplitude')
plt.legend(loc='best')



plt.figure('Fx interval 1')
plt.title('Fx interval 1: '+run_name)
plt.grid()
plt.plot(time_interval1,Fx_interval1*32.**3.,linewidth=0.5,label='interval 1')
plt.legend(loc='best')
plt.xlabel('time[s]')
plt.ylabel(r'$f^{SV}[N]$')

plt.figure('Fx interval 2')
plt.title('Fx interval 2: '+run_name)
plt.grid()
plt.plot(time_interval2,Fx_interval2*32.**3.,linewidth=0.5,label='interval 2')
plt.legend(loc='best')
plt.xlabel('time[s]')
```

```
plt.ylabel(r'$f^{SV}[N]$')

plt.figure('Fx interval 3')
plt.title('Fx interval 3: '+run_name)
plt.grid()
plt.plot(time_interval3,Fx_interval3*32.**3.,linewidth=0.5,label='interval 3')
plt.legend(loc='best')
plt.xlabel('time[s]')
plt.ylabel(r'$f^{SV}[N]$')

plt.figure('Fx interval 4')
plt.title('Fx interval 4: '+run_name)
plt.grid()
plt.plot(time_interval4,Fx_interval4*32.**3.,linewidth=0.5,label='interval 4')
plt.legend(loc='best')
plt.xlabel('time[s]')
plt.ylabel(r'$f^{SV}[N]$')


plt.figure('Fx interval 5')
plt.title('Fx interval 5: '+run_name)
plt.grid()
plt.plot(time_interval5,Fx_interval5*32.**3.,linewidth=0.5,label='interval 5')
plt.legend(loc='best')
plt.xlabel('time[s]')
plt.ylabel(r'$f^{SV}[N]$')


plt.figure('histogram')
plt.title('Histogram of all five intervals: '+run_name)
plt.grid()
plt.bar(bin_center_interval1,hist_interval,width=width_histogram_interval1)
plt.legend(loc='best')
plt.xlabel(r'$f^{SV}[N]$')
plt.ylabel('probability')

plt.figure('histogram abs')
plt.title('Histogram of all five intervals absolute values: '+run_name)
plt.grid()
plt.bar(bin_center_interval1_abs,hist_interval_abs,width=width_histogram_interval1_abs)
plt.legend(loc='best')
plt.xlabel(r'$f^{SV}[N]$')
plt.ylabel('probability')

plt.figure('Fx interval 1: FFT')
plt.title('Fx interval 1 FFT: '+run_name)
plt.grid()
plt.plot(Fx_freq1,Fx_fft1,linewidth=0.5,label='interval 1')
plt.legend(loc='best')
```

```
plt.xlabel('frequency[1/s]')
plt.ylabel('surge force[N]')
plt.xlim([-0.05, 20])

plt.figure('Fx interval 2: FFT')
plt.title('Fx interval 2 FFT: '+run_name)
plt.grid()
plt.plot(Fx_freq2,Fx_fft2,linewidth=0.5,label='interval 2')
plt.legend(loc='best')
plt.xlabel('frequency[1/s]')
plt.ylabel('surge force[N]')
plt.xlim([-0.05, 20])

plt.figure('Fx interval 3: FFT')
plt.title('Fx interval 3 FFT: '+run_name)
plt.grid()
plt.plot(Fx_freq3,Fx_fft3,linewidth=0.5,label='interval 3')
plt.legend(loc='best')
plt.xlabel('frequency[1/s]')
plt.ylabel('surge force[N]')
plt.xlim([-0.05, 20])

plt.figure('Fx interval 4: FFT')
plt.title('Fx interval 4 FFT: '+run_name)
plt.grid()
plt.plot(Fx_freq4,Fx_fft4,linewidth=0.5,label='interval 4')
plt.legend(loc='best')
plt.xlabel('frequency[1/s]')
plt.ylabel('surge force[N]')
plt.xlim([-0.05, 20])

plt.figure('Fx interval 5: FFT')
plt.title('Fx interval 5 FFT: '+run_name)
plt.grid()
plt.plot(Fx_freq5,Fx_fft5,linewidth=0.5,label='interval 5')
plt.legend(loc='best')
plt.xlabel('frequency[1/s]')
plt.ylabel('surge force[N]')
plt.xlim([-0.05, 20])


plt.show()
```

H Results: ShipX

			surge		sway	
Frequency	Period	wavenumber	amplitude	phase	amplitude	phase
0,209	30	0,00447	2,11E+06	89,75	6,50E+00	65,88
0,251	25	0,00644	2,94E+06	89,63	2,88E+00	-99,67
0,314	20	0,01006	4,25E+06	89,4	4,18E+01	114,58
0,331	19	0,01115	4,57E+06	89,32	2,64E+01	-148,96
0,349	18	0,01242	4,91E+06	89,24	2,65E+01	158,29
0,37	17	0,01392	5,24E+06	89,13	4,03E+01	-163,51
0,393	16	0,01572	5,54E+06	89,01	1,92E+01	-11,13
0,405	15,5	0,01675	5,67E+06	88,93	3,71E+00	37,29
0,419	15	0,01789	5,77E+06	88,85	4,98E+01	71,84
0,433	14,5	0,01914	5,83E+06	88,75	1,76E+01	-104,65
0,449	14	0,02053	5,83E+06	88,65	8,12E+01	67,1
0,465	13,5	0,02208	5,77E+06	88,53	2,39E+01	87,53
0,483	13	0,02381	5,61E+06	88,4	1,90E+01	119,99
0,503	12,5	0,02576	5,32E+06	88,25	5,34E+01	90,64
0,524	12	0,02795	4,89E+06	88,08	9,74E+01	-109,15
0,546	11,5	0,03043	4,28E+06	87,89	5,81E+01	174,69
0,571	11	0,03326	3,45E+06	87,7	7,75E+01	-168,54
0,598	10,5	0,0365	2,40E+06	87,54	3,37E+01	-171,54
0,628	10	0,04024	1,15E+06	87,73	9,26E+01	41,67
0,661	9,5	0,04459	2,14E+05	-99,68	6,60E+01	-133,87
0,698	9	0,04968	1,51E+06	-95,23	8,68E+01	-89,37
0,739	8,5	0,0557	2,41E+06	-95,66	2,02E+02	143,95
0,785	8	0,06288	2,53E+06	-96,66	1,57E+02	-96,11
0,838	7,5	0,07154	1,58E+06	-98,71	4,83E+01	-37,35
0,898	7	0,08213	9,75E+04	122,64	1,41E+02	10,7
0,967	6,5	0,09525	1,01E+06	82,97	2,33E+02	132,53
1,047	6	0,11179	6,18E+04	46,66	3,23E+01	95,04
1,142	5,5	0,13303	5,85E+05	-92,49	9,01E+01	-33,07
1,257	5	0,16097	4,44E+05	77,19	7,27E+01	66,41
1,396	4,5	0,19873	2,11E+05	-96,5	7,35E+01	24,25
1,571	4	0,25152	1,62E+05	-156,24	2,56E+02	-70,49

0 knots

heave		roll		pitch		ya
amplitude	phase	amplitude	phase	amplitude	phase	amplitude
4,48E+07	3,52	3,52E+00	142,13	6,78E+08	137,55	6,43E+02
4,19E+07	4,45	1,69E+00	-61,14	7,99E+08	128,13	3,25E+02
3,64E+07	5,58	3,12E+01	119,14	1,01E+09	118,23	3,73E+03
3,48E+07	5,79	2,70E+01	-135,65	1,07E+09	116,28	2,33E+03
3,28E+07	5,96	1,23E+01	-165,19	1,13E+09	114,33	2,55E+03
3,04E+07	6,07	4,84E+01	-152,57	1,18E+09	112,39	3,97E+03
2,75E+07	6,06	3,32E+01	29,28	1,24E+09	110,41	2,00E+03
2,59E+07	5,98	1,55E+01	46,85	1,26E+09	109,4	4,23E+02
2,40E+07	5,82	6,11E+01	86,66	1,28E+09	108,35	5,33E+03
2,20E+07	5,55	1,54E+01	-95,84	1,30E+09	107,26	1,58E+03
1,98E+07	5,12	6,22E+01	82,4	1,30E+09	106,09	7,00E+03
1,74E+07	4,48	2,65E+01	71,72	1,30E+09	104,83	1,98E+03
1,48E+07	3,47	6,47E-01	15,63	1,28E+09	103,42	2,02E+03
1,20E+07	1,88	2,90E+01	150,94	1,24E+09	101,82	5,51E+03
9,01E+06	-0,89	9,31E+01	-89,95	1,18E+09	99,91	8,99E+03
5,91E+06	-6,5	5,48E+01	-135,95	1,10E+09	97,55	4,94E+03
2,90E+06	-23,38	9,12E+01	-141,07	9,74E+08	94,46	7,58E+03
1,58E+06	-107,79	2,72E+01	-107,5	8,14E+08	90,13	2,71E+03
3,73E+06	-153,59	2,81E+01	116,11	6,18E+08	83,3	9,42E+03
5,80E+06	-164,97	5,02E+01	-131,22	3,96E+08	70,03	4,73E+03
6,94E+06	-171,89	8,22E+01	-35,81	2,02E+08	32,03	8,82E+03
6,73E+06	-178,86	2,46E+02	-156,19	2,17E+08	-42,43	2,03E+04
5,01E+06	171,19	2,39E+02	-40,77	3,38E+08	-75,08	1,45E+04
2,19E+06	147,56	6,67E+01	28,37	3,59E+08	-95	4,52E+03
1,36E+06	24,15	1,78E+02	61,22	2,36E+08	-120,27	1,42E+04
2,36E+06	-16,54	4,07E+02	178,21	7,63E+07	150,8	2,40E+04
8,92E+05	-40,38	9,82E+01	129	1,39E+08	55,22	2,44E+03
8,43E+05	115,81	2,16E+02	15,18	2,28E+07	-47,22	8,85E+03
5,34E+05	-66,73	1,83E+02	95,67	4,21E+07	176,42	4,23E+03
5,35E+05	94,82	1,36E+02	46,4	1,56E+07	-26,24	6,06E+03
4,51E+04	-73,58	6,57E+02	-51,78	3,08E+07	-17,28	2,28E+04

w	surge		sway		heave	
phase	amplitude	phase	amplitude	phase	amplitude	phase
62,16	2,11E+06	89,75	3,55E+00	79,95	4,48E+07	4,01
-107,74	2,94E+06	89,63	1,15E+01	119,93	4,20E+07	5,03
117	4,25E+06	89,4	2,51E+01	-150,67	3,65E+07	6,08
-149,42	4,57E+06	89,32	4,28E+01	-35,81	3,48E+07	6,19
158,16	4,91E+06	89,24	5,68E+01	-68,95	3,28E+07	6,19
-156,59	5,24E+06	89,13	2,17E+01	-64,23	3,04E+07	6,02
-11,03	5,54E+06	89,01	8,40E+01	-0,9	2,74E+07	5,54
111,24	5,67E+06	88,93	5,54E+01	127,06	2,57E+07	5,12
71,03	5,77E+06	88,85	4,84E+01	-150,01	2,38E+07	4,53
-109,61	5,83E+06	88,75	4,26E+01	127,22	2,18E+07	3,69
62,33	5,83E+06	88,65	2,61E+01	-46,61	1,95E+07	2,51
72,86	5,77E+06	88,53	9,77E+01	86,53	1,69E+07	0,81
120,91	5,61E+06	88,4	1,18E+02	8,29	1,42E+07	-1,72
92,6	5,32E+06	88,25	1,42E+01	-40,08	1,12E+07	-5,72
-108,86	4,89E+06	88,08	1,31E+02	-136,28	8,11E+06	-12,77
175,92	4,28E+06	87,89	3,84E+01	2,28	5,10E+06	-27,96
-171,65	3,45E+06	87,7	5,64E+01	157,74	3,00E+06	-69,77
-154,22	2,40E+06	87,54	4,07E+01	119,84	3,74E+06	-127,65
38,45	1,15E+06	87,73	6,14E+01	-53,76	5,84E+06	-152,4
-129,59	2,14E+05	-99,68	1,78E+02	153,78	7,48E+06	-164,82
-92,29	1,51E+06	-95,23	1,65E+02	-39,44	7,97E+06	-174,63
142,33	2,41E+06	-95,66	1,03E+02	73,88	6,97E+06	174,3
-100,27	2,53E+06	-96,66	1,71E+02	31,95	4,63E+06	156,79
-32,55	1,58E+06	-98,71	5,34E+01	45,24	2,09E+06	110,65
21,43	9,75E+04	122,64	7,37E+01	25,25	2,04E+06	23,45
120,44	1,01E+06	82,97	4,92E+01	-9	2,13E+06	-17,34
81,36	6,18E+04	46,66	4,69E+01	-53,24	8,92E+05	-48,54
-30,42	5,85E+05	-92,49	5,67E+01	134,7	3,36E+05	152,66
54,47	4,44E+05	77,19	3,18E+02	171,38	2,77E+05	83,39
31,68	2,11E+05	-96,5	4,53E+01	-21,98	2,26E+05	-124,98
-73,79	1,62E+05	-156,24	6,54E+01	-79,13	4,40E+05	-45,08

nots

roll		pitch		yaw		sur
amplitude	phase	amplitude	phase	amplitude	phase	amplitude
2,70E+00	108,9	6,79E+08	123,11	2,93E+02	97,99	2,11E+06
8,50E+00	130,28	8,18E+08	116,39	9,46E+02	126,82	2,94E+06
2,13E+01	-174,04	1,04E+09	110,24	2,39E+03	-140,87	4,25E+06
4,71E+01	-27,87	1,10E+09	109,1	4,55E+03	-26,42	4,57E+06
6,21E+01	-60,46	1,16E+09	107,98	5,04E+03	-59,55	4,91E+06
2,93E+01	-78,8	1,22E+09	106,85	2,05E+03	-61,59	5,24E+06
7,22E+01	-3,9	1,27E+09	105,63	7,41E+03	5,37	5,54E+06
4,92E+01	148,45	1,29E+09	104,96	4,70E+03	138,07	5,67E+06
2,48E+01	-105,76	1,31E+09	104,24	4,18E+03	-139,02	5,77E+06
3,53E+01	158,42	1,32E+09	103,42	3,65E+03	124,15	5,83E+06
2,50E+01	-16	1,33E+09	102,48	3,40E+03	-57,28	5,83E+06
1,18E+02	119,7	1,32E+09	101,37	8,79E+03	97,81	5,77E+06
1,38E+02	50,54	1,29E+09	100,02	1,05E+04	11,5	5,61E+06
5,06E+01	-90,98	1,25E+09	98,34	1,77E+03	-18,29	5,32E+06
1,53E+02	-87,05	1,18E+09	96,17	1,22E+04	-132,36	4,89E+06
4,89E+01	24,2	1,07E+09	93,29	3,43E+03	29,13	4,28E+06
7,83E+01	-113,75	9,36E+08	89,23	1,56E+02	156,04	3,45E+06
6,13E+01	-166,03	7,61E+08	83,09	3,23E+03	127,78	2,40E+06
9,89E+01	-3,67	5,54E+08	72,6	6,32E+03	-49,35	1,15E+06
3,25E+02	-159,61	3,46E+08	50,48	1,82E+04	157,9	2,14E+05
3,13E+02	22,62	2,33E+08	-1,4	1,92E+04	-34,55	1,51E+06
1,79E+02	121,99	3,01E+08	-52,25	1,04E+04	80,45	2,41E+06
3,44E+02	72,9	3,68E+08	-78,71	1,58E+04	34,49	2,53E+06
1,73E+02	95,01	3,22E+08	-101,57	6,70E+03	28,71	1,58E+06
1,94E+02	55,1	1,83E+08	-137,42	6,49E+03	21,69	9,75E+04
1,04E+02	20,31	8,79E+07	140	3,97E+03	-8,69	1,01E+06
1,13E+02	-25,52	8,12E+07	60,71	3,84E+03	-57,12	6,18E+04
1,34E+02	122,36	3,17E+07	30,93	4,90E+03	137,63	5,85E+05
8,76E+02	-169,99	4,56E+07	-111,93	2,96E+04	172,57	4,44E+05
7,04E+01	-6,36	4,95E+07	66,28	4,65E+03	-24,21	2,11E+05
1,93E+02	-73,32	3,76E+07	101,15	6,13E+03	-76,82	1,62E+05

15 knots

ge	sway		heave		roll	
phase	amplitude	phase	amplitude	phase	amplitude	phase
89,75	8,74E+00	-47	4,48E+07	4,12	1,06E+01	-29,87
89,63	1,94E+01	-37,65	4,20E+07	5,15	1,92E+01	-54,7
89,4	3,15E+01	173,4	3,65E+07	6,13	1,81E+01	162,63
89,32	1,40E+01	157,96	3,48E+07	6,21	1,75E+01	-176,52
89,24	7,44E+00	-26,77	3,28E+07	6,15	8,92E+00	2,59
89,13	4,67E+01	-103,26	3,04E+07	5,87	3,35E+01	-56,72
89,01	7,75E+01	76,91	2,74E+07	5,24	7,15E+01	97,14
88,93	9,79E+00	-49,76	2,56E+07	4,72	3,21E+01	-33,1
88,85	4,57E+01	105,27	2,37E+07	3,99	6,63E+01	117,64
88,75	7,39E+01	-38,9	2,16E+07	2,98	5,52E+01	-2,01
88,65	7,15E+01	-149,56	1,93E+07	1,56	4,35E+01	-99,74
88,53	1,05E+02	-81,27	1,67E+07	-0,46	6,34E+01	-44,38
88,4	3,67E+01	4,4	1,39E+07	-3,46	7,44E+01	34,52
88,25	1,11E+02	165,56	1,09E+07	-8,21	1,52E+02	-138,71
88,08	1,45E+02	135,33	7,77E+06	-16,7	1,17E+02	-176,91
87,89	7,44E+01	-139,42	4,85E+06	-35,3	1,40E+02	-69,1
87,7	9,56E+01	84,69	3,20E+06	-81,93	1,57E+02	125,31
87,54	3,40E+01	-16,64	4,32E+06	-131,87	4,90E+01	9,42
87,73	2,59E+02	-73,21	6,36E+06	-153,68	5,03E+02	-33,53
-99,68	5,92E+01	-23,74	7,83E+06	-165,75	8,10E+01	56,53
-95,23	1,50E+01	69,65	8,08E+06	-175,78	1,63E+01	-140,5
-95,66	4,72E+01	134,04	6,84E+06	172,57	1,19E+02	160,82
-96,66	9,50E+01	-174,45	4,34E+06	153,31	2,16E+02	-131,28
-98,71	5,49E+01	-26,03	1,98E+06	100,19	1,36E+02	3,18
122,64	2,04E+02	-140,43	2,14E+06	20,56	4,66E+02	-116,01
82,97	1,83E+02	-0,4	2,00E+06	-16,15	4,29E+02	31,47
46,66	1,01E+02	114,62	8,11E+05	-43,52	2,32E+02	132,95
-92,49	2,22E+02	-95,1	2,97E+05	179,13	5,63E+02	-84,23
77,19	3,34E+02	-177,26	4,16E+05	89,5	7,89E+02	-171,75
-96,5	4,73E+02	15,75	4,49E+05	-118,99	1,16E+03	19,48
-156,24	3,59E+01	-107,66	5,21E+05	-47,44	1,48E+02	-111,8

pitch		yaw		15 knots added res
amplitude	phase	amplitude	phase	dimensionless
6,85E+08	119,91	8,81E+02	-23,03	4,40E-03
8,25E+08	113,91	1,82E+03	-23,75	1,44E-02
1,05E+09	108,66	3,22E+03	-175,72	8,81E-02
1,11E+09	107,71	1,64E+03	166,52	1,39E-01
1,17E+09	106,78	5,31E+02	14,88	2,28E-01
1,23E+09	105,81	4,37E+03	-87,03	3,90E-01
1,28E+09	104,75	7,25E+03	87,94	6,99E-01
1,30E+09	104,15	5,98E+02	-81,44	9,53E-01
1,32E+09	103,48	4,51E+03	114,1	1,32E+00
1,33E+09	102,7	7,14E+03	-29,62	1,86E+00
1,33E+09	101,79	7,14E+03	-143,72	2,65E+00
1,32E+09	100,69	9,39E+03	-73,48	3,81E+00
1,29E+09	99,31	3,56E+03	14,6	5,38E+00
1,24E+09	97,57	9,36E+03	162,19	6,95E+00
1,17E+09	95,3	1,33E+04	141,65	7,34E+00
1,06E+09	92,22	7,19E+03	-145,63	6,16E+00
9,18E+08	87,84	8,59E+03	87,03	4,52E+00
7,37E+08	81,11	2,98E+03	19,32	3,15E+00
5,29E+08	69,35	1,67E+04	-78,13	2,16E+00
3,27E+08	43,98	7,61E+03	-3,24	1,59E+00
2,44E+08	-10,37	3,44E+03	123,99	1,32E+00
3,22E+08	-55,42	3,69E+03	106,77	1,14E+00
3,73E+08	-79,87	9,58E+03	176,69	9,39E-01
3,05E+08	-102,61	5,31E+03	-30,73	7,64E-01
1,60E+08	-141,48	1,82E+04	-135,53	6,86E-01
8,37E+07	134,08	1,70E+04	2,07	6,84E-01
6,65E+07	67,15	9,76E+03	116,88	7,11E-01
3,83E+07	46,57	2,08E+04	-91,94	7,51E-01
5,66E+07	-101,36	3,12E+04	-175,87	7,99E-01
7,03E+07	67,24	4,38E+04	17,67	8,54E-01
4,79E+07	100,82	3,40E+03	-106,47	9,24E-01

12 knots added res dimensionless
3,57E-03
1,14E-02
6,64E-02
1,04E-01
1,67E-01
2,81E-01
4,91E-01
6,60E-01
8,97E-01
1,24E+00
1,72E+00
2,43E+00
3,45E+00
4,79E+00
6,06E+00
6,18E+00
4,94E+00
3,53E+00
2,52E+00
1,91E+00
1,61E+00
1,38E+00
1,10E+00
8,28E-01
6,82E-01
6,40E-01
6,42E-01
6,66E-01
7,01E-01
7,45E-01
8,00E-01

I Results: pre-processing

Run	heave	pitch	WP1 perio	WP3 perio	WP1 heigh	WP3 heigh	wavelengtl	heave peri
4000	0,002963	0,187916	0,94875	0,266472	0,006886	0,007856	1,405377	0,173721
4010	0,002475	0,198052	0,866198	0,435389	0,004444	0,033081	1,171449	0,262367
4020	0,001947	0,187355	1,060513	0,585315	0,045557	0,056129	1,755986	0,203063
4030	0,002204	0,233591	1,237463	0,729696	0,072183	0,075897	2,390856	0,215534
4040	0,002177	0,287852	1,379918	0,84875	0,08018	0,079851	2,973006	0,263314
4050	0,002278	0,261544	1,504546	0,955632	0,09229	0,093055	3,534271	0,492088
4060	0,001672	0,690143	1,626863	1,062532	0,112193	0,111626	4,132292	0,323623
4070	0,003319	1,677291	1,749896	1,171268	0,127375	0,12567	4,780941	0,485217
4080	0,022163	2,278147	1,802391	1,217574	0,133143	0,130677	5,072089	1,039524
4090	0,041877	3,101419	1,858296	1,267692	0,145675	0,139473	5,391598	1,1095
4100	0,062913	3,696838	1,91093	1,315318	0,150866	0,148467	5,701333	1,275294
4110	0,072612	3,968932	1,929535	1,332097	0,155062	0,148476	5,81288	1,292121
4120	0,075262	4,058119	1,942381	1,34371	0,157906	0,151886	5,890525	1,343125
4130	0,084352	4,262922	1,961786	1,360164	0,155849	0,150879	6,008798	1,360625
4140	0,09822	4,51484	2,000854	1,395678	0,153131	0,151224	6,250465	1,396129
4150	0,126284	5,22374	2,048375	1,439914	0,162654	0,156663	6,550815	1,439333
4160	0,140124	5,783305	2,083846	1,471696	0,168349	0,164942	6,779566	1,471724
4170	0,158	6,501036	2,136053	1,518818	0,175323	0,173278	7,123309	1,517931
4180	0,180001	7,549501	2,231081	1,605962	0,191169	0,189428	7,770455	1,606667
4190	0,188359	8,148263	2,318056	1,686225	0,206267	0,205709	8,386687	1,686923
7500	0,002073	0,174216	0,39097	2,736	0,002127	0,006093	0,238658	0,231362
7510	0,002308	0,345941	1,237611	0,794291	0,07258	0,06747	2,391429	0,357006
7520	0,002998	0,338391	1,504324	1,031667	0,093	0,08699	3,533232	0,4975
7540	0,003379	1,923341	1,749844	1,252167	0,126112	0,111164	4,780656	0,518103
7550	0,032803	2,56776	1,802581	1,303023	0,133032	0,126952	5,073154	1,223674
7560	0,051688	3,346487	1,858167	1,351463	0,14468	0,13354	5,390851	1,265532
7570	0,07512	4,108329	1,910862	1,403671	0,149412	0,143409	5,700926	1,403256
7590	0,086504	4,455176	1,942281	1,42987	0,155171	0,144645	5,889917	1,429048
7610	0,103435	5,039271	2,000818	1,485467	0,15139	0,147832	6,250244	1,4855
7620	0,116942	5,782228	2,048333	1,530486	0,16134	0,152216	6,550548	1,529474
7640	0,13524	6,689962	2,136442	1,613162	0,1773	0,167392	7,125906	1,612222
7650	0,146219	7,11247	2,2313	1,701591	0,190058	0,183168	7,771978	1,7

pitch perio	heave RAO	pitch RAO	mean Fx	Fx amplitu	Fx period	dim. mean	dim. Fx am	encounter
0,200667	0,37717	0,014733	31,33782	46,51723	0,530095	306,5268	447,7633	0,495309
0,204612	0,074812	0,009844	33,02806	47,00215	0,435285	18,61907	25,51456	0,432512
0,190839	0,03469	0,009919	33,25549	259,2598	0,586189	8,65606	48,88673	0,583024
0,188367	0,029041	0,014215	41,76498	168,7115	0,728391	5,914992	17,39922	0,727117
0,236313	0,027266	0,022525	44,30867	190,4089	0,8481	5,594246	17,74018	0,846877
0,24724	0,024482	0,022264	44,60182	177,5868	0,958391	4,206519	12,18313	0,953887
0,756396	0,014977	0,060544	63,6322	216,8818	1,065449	3,949577	10,33997	1,060621
1,1088	0,02641	0,15882	93,07833	318,9999	1,172606	5,665067	11,99934	1,169451
1,12	0,169604	0,224177	134,1609	336,7858	1,221471	5,904952	11,71619	1,216285
1,230588	0,300252	0,309967	171,8053	338,5105	1,263333	5,78998	10,33767	1,266397
1,274462	0,423749	0,373661	167,1893	362,4273	1,318095	5,103755	9,767619	1,313793
1,332064	0,489046	0,411438	170,9308	368,4249	1,337339	5,561824	9,928063	1,330593
1,343871	0,495517	0,418441	167,5257	373,893	1,34121	5,408853	9,628099	1,342206
1,360984	0,55907	0,453396	159,6988	382,7015	1,361475	4,246461	9,986937	1,359772
1,395333	0,649499	0,50444	144,2147	387,4928	1,395509	3,485528	10,06592	1,395211
1,439649	0,806083	0,599658	120,8072	439,7361	1,433707	2,525621	10,64356	1,438448
1,472281	0,84953	0,658714	97,5593	467,3632	1,478304	2,275596	10,20516	1,470815
1,518519	0,911828	0,750701	68,36442	461,6303	1,511364	2,087454	9,13347	1,518577
1,606154	0,950235	0,89158	29,65331	431,4793	1,597981	0,721495	7,143309	1,605907
1,686531	0,91566	0,976921	-8,23477	472,2162	1,688163	0,121556	6,629244	1,686243
0,159545	0,340272	1,849389	19,80774	33,50754	0,136883	317,0979	536,1317	0,140777
0,306137	0,034203	0,028059	30,24151	131,9084	0,794255	5,603464	17,2141	0,79261
0,308375	0,034459	0,035912	35,15581	158,5367	1,033935	4,119281	12,44566	1,029023
1,148866	0,030394	0,235307	84,09431	245,2364	1,25073	5,112117	11,7893	1,252489
1,191183	0,258391	0,297876	114,1256	252,6535	1,298895	4,722733	9,312658	1,301059
1,335422	0,387056	0,39701	144,3739	277,976	1,353293	5,158552	9,260007	1,352437
1,4035	0,523816	0,489593	141,8964	331,5188	1,407405	2,624526	9,575985	1,401309
1,429744	0,598043	0,546224	130,2098	346,574	1,427821	2,475108	9,840505	1,430519
1,4856	0,699679	0,652443	93,68895	350,0697	1,483933	2,170552	9,515888	1,485081
1,530411	0,768263	0,77181	64,72426	337,4203	1,525822	1,771828	8,651288	1,529495
1,613529	0,807927	0,902114	28,96432	359,2351	1,614044	0,890345	7,616207	1,612126
1,701515	0,798278	0,975207	-18,101	407,0257	1,698561	-0,08342	7,206931	1,701452

period