NTNU – Trondheim
Norwegian University of
Science and Technology

Time-Domain Simulation of Floating, Dynamic Marine Structures using USFOS

Marianne Mellbye Larsen

Master of Science in Engineering and ICT
Submission date: June 2013
Supervisor: Jørgen Amdahl, IMT

Norwegian University of Science and Technology
Department of Marine Technology

NTNU – Trondheim
Norwegian University of
Science and Technology

Master Thesis

Spring 2013

Time-Domain Simulation of Floating, Dynamic Marine Structures using USFOS

Keywords: Time-domain analysis, frequency dependence, memory function, USFOS

Author:
Marianne Mellbye Larsen

Supervisor:
Professor Jørgen Amdahl

Delivered: 10.06.2013 *Availability:* Open

NTNU
Norges teknisk-naturvitenskapelige universitet
Institutt for marin teknikk

MASTER THESIS 2013
for
Stud. techn. Marianne Mellbye Larsen

Time-domain simulation of floating, dynamic marine structures using USFOS

Tidsplansimulering av flytende, dynamiske marine konstruksjoner med USFOS

USFOS is a computer program for simulation of the response to extreme environmental and accidental loads. It was originally developed for fixed offshore structures like jackets and jack-ups. In recent years the program has been modified to account also for floating offshore structures. Floating offshore structures are subjected to diffraction and radiation terms, which are often simulated as added mass and damping terms. In USFOS these masses are at present assumed to be constant. In reality these terms are frequency dependent. In irregular waves the frequency dependence should be accounted for by using memory functions expressed by convolution integrals. The purpose of this work is to investigate the feasibility of including added mass and frequency dependence in USFOS. The work will start with analysis of a single degree of freedom problem using MATLAB

The work is proposed to be carried out in the following steps.

1. Discuss the frequency dependence of the added mass and damping terms for various floating and submerged bodies representative for pontoon type structures. Describe how the convolution integral (retardation function) can be established, based on both the added mass term and the damping term. Compare the retardation functions calculated with the two methods and comment any differences. Describe the challenges related to establishing the retardation function and how the accuracy can be checked by inverse calculation of the added mass and damping term. (Reference is made to PhD thesis of Olav Rognebakke).

2. Develop a MATLAB code for numerical, stepwise incrementation of a single degree of freedom system where the added mass and damping are frequency dependent. Simulate the response of the system for a single excursion, in regular wave and irregular seas. Illustrate the time history of the memory functions. Check the amount of "memory" that is needed to obtain satisfactory results. Investigate also the importance of the memory effect by comparing with results with constant added mass and damping terms.

3. On the basis of the experience obtained with the single degree of freedom system, describe how frequency dependent added mass and damping can be implemented in USFOS. This should include a flow diagram with input/output data defined.

4. To the extent time permits write a subroutine in USFOS to calculate the retardation function for a pontoon of a floating platform. Perform simulations with USFOS for a

simple system and compare the results with those from the MATLAB program.

5. Conclusions and recommendation for further work.

Literature studies of specific topics relevant to the thesis work may be included.

The work scope may prove to be larger than initially anticipated. Subject to approval from the supervisors, topics may be deleted from the list above or reduced in extent.

In the thesis the candidate shall present his personal contribution to the resolution of problems within the scope of the thesis work.

Theories and conclusions should be based on mathematical derivations and/or logic reasoning identifying the various steps in the deduction.

The candidate should utilise the existing possibilities for obtaining relevant literature.

Thesis format
The thesis should be organised in a rational manner to give a clear exposition of results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Telegraphic language should be avoided.

The thesis shall contain the following elements: A text defining the scope, preface, list of contents, summary, main body of thesis, conclusions with recommendations for further work, list of symbols and acronyms, references and (optional) appendices. All figures, tables and equations shall be numerated.

The supervisors may require that the candidate, in an early stage of the work, present a written plan for the completion of the work. The plan should include a budget for the use of computer and laboratory resources, which will be charged to the department. Overruns shall be reported to the supervisors.

The original contribution of the candidate and material taken from other sources shall be clearly defined. Work from other sources shall be properly referenced using an acknowledged referencing system.

The report shall be submitted in two copies:
- Signed by the candidate
- The text defining the scope included
- In bound volume(s)
- Drawings and/or computer prints that cannot be bound should be organised in a separate folder.

Ownership
NTNU has according to the present rules the ownership of the thesis. Any use of the thesis has to be approved by NTNU (or external partner when this applies). The department has the right to use the thesis as if a NTNU employee carried out the work, if nothing else has been agreed in advance.

Thesis supervisor

Prof. Jørgen Amdahl

Deadline: June 10, 2013

Trondheim, January 15, 2013

Jørgen Amdahl

Preface

This master thesis was written during the spring semester 2013, at the Norwegian University of Science and Technology (NTNU). The work is conducted independently, and is partially based on the work done in my project thesis from fall 2012.

The objective is to look into the problem of including frequency-dependence in the added mass and damping terms in USFOS, to allow time-domain simulations of floating structures.

The work has been challenging regarding both the programming and the technical aspects, as transient effects have been mentioned, but disregarded in my hydrodynamic courses at NTNU. The absence of a standardized nomenclature used in the books regarding time-domain effects, has made the learning process more difficult. Slowly building a better understanding of these effects throughout the semester have been a great motivation.

As part of the work with this thesis, a MATLAB function for the convolution of two vectors has been written. However, MATLAB comes with a convolution function implemented in the standard library. This predefined function is only used for verification of correctness of the function written by the author. This is to make a transition to FORTRAN code possible.

Marianne Mellbye Larsen,
Trondheim, June 10, 2013

Abstract

This master thesis deals with calculation of the convolution integrals that constitute the memory functions in the linear time-domain wave response equation. Basic theory needed to calculate the forces acting on floating structures is presented. The main motivation for the thesis is to investigate the feasibility of including frequency dependent added mass and damping in USFOS. USFOS is a computer program originally developed to simulate the response to extreme environmental and accidental loads for fixed offshore structures. The program has been modified to be able to handle also floating offshore structures. Floating structures are however subjected to diffraction and radiation terms often simulated as added mass and damping terms. So far, these terms are assumed to be constant in USFOS. This is not correct, as the added mass and damping terms are frequency dependent in reality. The frequency dependence can be accounted for by adding memory functions expressed as convolution integrals.

The convolution terms are integrated from 0 to t, i.e. from the beginning of time and up to this moment. This is neither possible, due to computer restrictions, nor necessary, due to damping effects. This thesis concentrates on finding the amount of 'memory' that needs to be included in the convolution integral. A single degree of freedom system illustrating a platform oscillating in heave is analysed in MATLAB, both for regular waves and irregular seas. The convolution integral is calculated with different time intervals included in the memory function, and the deviations from the full integral are analysed.

Finally, a brief description is given of how the experiences obtained with the single degree of freedom system can be utilized in the implementation of frequency dependent added mass and damping terms in USFOS.

Sammendrag

Denne masteroppgaven omhandler utregningen av konvolusjonsintegralene som utgjør minne-funksjonene i den lineære bølgeresponsfunksjonen i tids-domenet. Grunnleggende teori for utregning av kreftene som virker på flytende konstruksjoner blir presentert i oppgaven. Hovedmotivasjonen er å utforske muligheten for å inkludere frekvensavhengighet i tilleggsmasse- og dempningsleddene i USFOS. USFOS er et program som opprinnelig ble skrevet for å simulere responsen fra ekstreme miljø- og ulykkeslaster for faste offshore konstruksjoner. Programmet er senere modifisert for å kunne håndtere også flytende konstruksjoner. Flytende konstruksjoner er imidlertid utsatt for diffraksjons- og eksitasjonslaster. Hittil er disse uttrykkene antatt å være konstante i USFOS, men dette er ikke helt korrekt da tilleggsmassen og dempningen i virkeligheten er frekvensavhengige. Denne frekvensavhengigheten kan bli redegjort for ved å inkludere minnefunksjoner uttrykt som konvolusjonsintegraler.

Konvolusjonsleddene skal integreres fra 0 til t, det vil si fra tidenes morgen og opp til dette øyeblikket. Dette er verken mulig, på grunn av databegrensninger, eller nødvendig, på grunn av dempning. Denne oppgaven konsentrerer seg om å finne den nødvendige mengden 'minne' som må inkluderes i konvolusjonsintegralet. Et masse-fjær system med én frihetsgrad som illustrerer en platform som svinger i hiv analyseres i MATLAB, både for regulære bølger og irregulær sjø. Konvolusjonsintegralet regnes ut med ulike tidsintervaller inkludert i minnefunksjonen, og avvikene fra det fulle integralet blir analysert.

Til slutt blir det gitt en kort forklaring av hvordan erfaringene som har blitt oppnådd gjennom analysene av masse-fjær systemet kan benyttes i implementasjonen av frekvensavhengighet for tilleggsmasse- og dempningsleddene i USFOS.

Acknowledgements

This thesis is written under supervision of Professor Jørgen Amdahl at the Department of Marine Technology at NTNU. I am very grateful for the guidance I got from him during this work. His enthusiasm together with his knowledge is truly inspiring when dealing with complex problems. Despite of his tight schedule, he has always found time to give me some pointers.

During the work with this thesis, the help and motivation from my classmates in room C1.077 have been greatly appreciated. They made long days at the office manageable, sometimes even fun. Whenever I felt the problems at hand to be impossible to overcome, they helped motivate me to continue the work.

Special thanks is also given to the rest of my friends, who have had to put up with numerous questions over the last months regarding the more practical aspects of a master thesis, from paper sizes to English grammar.

Lastly, I would like to thank my family for always being there for me and believing in me.

Marianne Mellbye Larsen,
Trondheim, June 10, 2013

Nomenclature

α	Constant in the JONSWAP sea spectrum
$\boldsymbol{\omega}$	Rotation
$\ddot{\eta}_3$	Acceleration in heave direction
$\dot{\eta}_3$	Velocity in heave direction
η_1, η_2, η_3	Oscillatory rigid-body translatory motions; *surge*, *sway* and *heave*
η_4, η_5, η_6	Oscillatory rigid-body angular motions; *roll*, *pitch* and *yaw*
η_j	Wave induced vessel motion response, see η_{1-6}
γ	Constant in the JONSWAP sea spectrum
λ	Wavelength
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	Unit vectors in x- y- and z-direction respectively
$\mathbf{r}$	Distance from the origin to a given position
$\mathbf{s}$	Motion of an arbitrary point on a structure
ω	Circular frequency in radians per second $= \frac{2\pi}{T}$
ω_1	Frequency of first wave component
ω_l	Lower integral limit of ω
ω_p	Constant in the JONSWAP sea spectrum
ω_u	Upper integral limit of ω
ρ	Mass density of sea water $= 1025 \frac{kg}{m^3}$
σ	Constant in the JONSWAP sea spectrum
ε	Phase angle
φ	Velocity potential
ζ	Surface elevation

ζ_a	Wave amplitude
a_c	Normal component of cylinder acceleration
A_n	Amplitude of wave component n
a_n	Wave induced acceleration normal to cylinder axis
a_x	x-component of acceleration
a_z	z-component of acceleration
A_{33}	Added mass in heave
$A_{33}(\infty)$	Infinite frequency limit for added mass in heave
A_{33}^{2D}	2D added mass in heave
$A_{ij}(\omega)$	Frequency dependent added mass component
A_{ij}^{∞}	Infinite frequency limit for added mass component
A_{kj}	Added mass coefficients
A_{WP}	Water-plane area
B	Damping matrix
B_{33}	Damping in heave
$B_{33}(\infty)$	Infinite frequency limit for damping in heave
B_{33}^{2D}	2D damping in heave
$B_{ij}(\omega)$	Frequency dependent damping component
B_{kj}	Damping coefficients
C	Stiffness matrix
C_D	Drag coefficient
C_M	Mass coefficient
C_{33}	Restoring in heave
C_{kj}	Restoring coefficients
D	Cylinder diameter
F	Force
F_3	Force in heave direction
F_k	Force component, where $k = 1, 2, ..., 6$
g	Acceleration of gravity $= 9.81 \frac{m}{s^2}$
H_S	Significant wave height, in this thesis chosen as $H_{1/3}$

$H_{1/3}$	Average of the 1/3 highest waves
h_{33}	Retardation function (impulse response function) in heave
h_{ij}	Retardation function (impulse response function)
i	Wave component number, $i = 1, 2, ..., N$, where N is the total number of wave components
k	Wave number $= \frac{2\pi}{\lambda} = \frac{\omega^2}{g}$
k_i	Return period
L	Length
M	Mass matrix
N	Number of wave components
n	Unit normal vector
p	Pressure
S	Average wetted surface
$S(\omega)$	Wave spectrum
T	Wave period
t	Time
T_r	Return period of irregular sea pattern
T_Z	Zero up-crossing period
u	x-component of velocity
u_n	Wave induced velocity normal to cylinder axis
u_{rel}	Relative velocity between fluid and cylinder normal to cylinder axis
w	z-component of velocity

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans have traveled and worked at sea for thousands of years. Knowledge of the sea and of how floating structures respond to different sea states, are a crucial part of protecting lives and property at sea. Floating structures might have complex geometries and loads, and the sea changes constantly, leaving the calculation of the motion responses to be quite comprehensive.

Many advanced computer programs have been written to simulate motion responses and loads on structures in different sea states. Accidents or extreme environmental conditions might lead to for instance flooding of some compartments, causing the buoyancy of the structure to change. Detailed knowledge of how a structure will be able to cope under such conditions is of high importance in designing safer ships and platforms.

1.1 Motivation

USFOS is a computer program for simulation of the response to extreme environmental and accidental loads. The program was originally developed for fixed offshore structures like jackets and jack-ups. USFOS have been modified in more recent years to be able to also simulate the response of floating offshore structures. One of the new challenges in the simulations due to this modification is connected to the added mass and damping terms. These terms are used to simulate the diffraction and radiation loads. They are at present assumed to be constant, but are in reality dependent of frequency.

For USFOS to be able to account for transient effects and not only steady-state conditions, a memory function must be introduced. The memory function is expressed by convolution integrals, which should be integrated from the beginning of time and up to present time. For practical reasons, this is not possible. This thesis is written as an investigation of the feasibility of including frequency dependence in

the added mass and damping terms in USFOS. The amount of time necessary to include in the memory function will be examined.

1.2 Background

Numerous books and articles have been written about the subject of understanding the motions of floating bodies. Some basic background is given here as a framework for discussions of the wave-induced motions.

1.2.1 Problem simplifications

As stated by Vugts, the general problem of floating structures at sea involves: "(...) the dynamic equilibrium of forces and moments in and on an elastic body moving in the interface of two different media" [9]. The problem is simplified by only considering the external forces on the submerged parts of the vessel. This is a valid simplification for many cases because the density of seawater is a thousand times larger than the density of air [9]. There are situations in which the air environment must be included in the calculations, for instance in manoeuvring a ship with large superstructures in strong winds [9], but these effects are neglected in this thesis.

Another simplification made is that the structures are assumed to be rigid bodies, a simplification that is valid as long as no structural or vibrational problems are dealt with [9].

To further simplify the problem, the water is considered incompressible, inviscid and irrotational. Finally, the problem is considered to be linear, allowing irregular seas to be simulated by superpositioning the contributions for different regular wave components [3].

1.2.2 Definition of motions

A floating structure subjected to waves might have response motions, both translatory and angular. The motions will be oscillatory, with the frequency depending on the sea state [1]. To describe the waves and response motions of the analysed structure, a coordinate system must be chosen. The axis system used in this thesis is defined with its origin and xy-plane at the undisturbed free surface, and the z-axis perpendicular to it. The z-axis is chosen so it will pass through the structures center of gravity when the structure is floating in still water [1].

The translational oscillatory motions are referred to as *surge*, *sway* and *heave*, where heave is the vertical motion, while surge and sway are the horizontal motions. For structures with forward motion, such as ships, surge is defined as the longitudinal motion. The rotational oscillatory motions are *roll*, *pitch* and *yaw*, where yaw

is rotation about the vertical axis, and roll and pitch are rotations about the horizontal axes. The axis system with oscillatory motion definitions is shown in Figure 1.1.

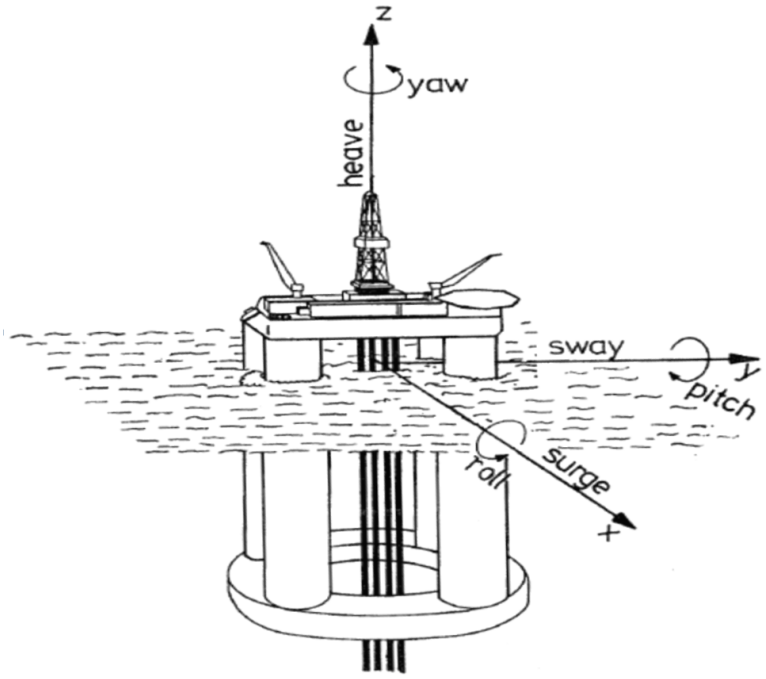

Figure 1.1: Axis system showing translational and rotational oscillatory motions [1]

The motion components are denoted η_1, η_2, ... η_6, where the first three components ($i = 1, 2, 3$) refer to the translatory motions, and the last three components ($i = 4, 5, 6$) refer to the rotations as shown in table 1.1.

Translations		Rotations	
η_1	surge	η_4	roll
η_2	sway	η_5	pitch
η_3	heave	η_6	yaw

Table 1.1: Rigid oscillatory motion components for floating structures

Assuming small motions, the motion of any part of the structure can be written as in equation (1.1). $\boldsymbol{\omega}$ is the rotation, and is given by equation (1.2). $\mathbf{r}$ is the distance from the origin to the given location, and is found in equation (1.3). $\times$ denotes the vector product, while $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are unit vectors along the x-, y- and z-axes, respectively.

$$\mathbf{s} = \eta_1\mathbf{i} + \eta_2\mathbf{j} + \eta_3\mathbf{k} + \boldsymbol{\omega} \times \mathbf{r} \tag{1.1}$$

$$\boldsymbol{\omega} = \eta_4\mathbf{i} + \eta_5\mathbf{j} + \eta_6\mathbf{k} \tag{1.2}$$

$$\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{1.3}$$

The rigid-body motion of any point on the floating structure given in equation (1.1) can hence be written as in equation (1.4).

$$\mathbf{s} = (\eta_1 + z\eta_5 - y\eta_6)\mathbf{i} + (\eta_2 + x\eta_6 - z\eta_4)\mathbf{j} + (\eta_3 + y\eta_4 - x\eta_5)\mathbf{k} \tag{1.4}$$

1.3 Scope of work

The intended work scope proved to be larger than initially anticipated, and was reduced in extent in cooperation with the supervisor. The MATLAB code written in the second step of the proposed work description was reduced to focus on calculating the convolution integrals constituting the memory terms. The investigation of the importance of the memory effects was hence left for further work. The fourth step of the proposed work description was completely omitted in this thesis due to time restrictions.

1.4 Outline of thesis

A brief introduction to the sea environment explaining how to calculate the surface elevation and corresponding vertical velocity component for regular waves are given in chapter 2. Wave spectra are introduced, and the process of using them to simulate irregular seas are explained.

Chapter 3 explains how the response motions are found in the frequency domain assuming steady-state conditions. The frequency dependence of the added mass and damping terms for various floating and submerged bodies representative for pontoon type structures are discussed.

The linear time-domain heave response of a floating structure is presented in chapter 4. How the retardation function in the convolution integral can be established is explained, based on both added mass and damping. The two techniques are tested for a rectangular cross-section oscillating at the surface, and the obtained functions are compared. The challenges related to establishing the retardation function is then described, and a procedure for checking the accuracy by inverse calculation is presented.

Chapter 5 explains the MATLAB program written in order to assess how much time must be included in the memory functions. The results of the analyses are then presented and discussed in chapter 6.

Chapter 7 deals with USFOS, describing how frequency dependent added mass and damping can be implemented based on the experiences obtained with the system tested in MATLAB. A flow diagram is presented, explaining a possible program flow of the subroutine.

Finally, chapter 8 presents the conclusion of this thesis and recommendations for further work.

Chapter 2

The sea environment

To be able to predict a floating structures response to different sea states, a set of equations describing the seas characteristics are needed. The derivation of the equations valid for linear wave theory is found in Newman (1977) [10]. Some important resulting equations are presented in this chapter.

2.1 Regular waves

As mentioned in the introduction, the sea water is assumed to be incompressible and inviscid, with irrotational fluid motion [1]. A velocity potential describes the fluid velocity vector for a water particle located at the point (x, y, z) at a given time t [1]. Assuming infinite water depth, the velocity potential is given as in equation (2.1). Using the same assumptions, the wave profile is given in equation (2.2). ζ_a is the wave amplitude.

$$\varphi = \frac{g\zeta_a}{\omega} e^{kz} \cos(\omega t - kx) \tag{2.1}$$

$$\zeta = \zeta_a \sin(\omega t - kx) \tag{2.2}$$

The dynamic pressure in the fluid due to the wave is given by equation (2.3).

$$p = \rho g \zeta_a e^{kz} \sin(\omega t - kx) \tag{2.3}$$

The x- and z-components of velocity can be found by derivation of the velocity potential in equation (2.1) with respect to x and z respectively. The velocity in x-direction is given in equation (2.4) and in z-direction in equation (2.5). "It should

be noted that the linear theory assumes the velocity potential and fluid velocity to be constant from the mean free-surface to the free-surface level" [1].

$$u = \omega\zeta_a e^{kz} \sin(\omega t - kx) \tag{2.4}$$

$$w = \omega\zeta_a e^{kz} \cos(\omega t - kx) \tag{2.5}$$

Similarly the x- and z-components of acceleration are found by derivation with respect to time of the corresponding velocity component. Derivation of (2.4) gives the acceleration in x-direction given in equation (2.6), while derivation of (2.5) gives the acceleration in z-direction given in equation (2.7).

$$a_x = \omega^2\zeta_a e^{kz} \cos(\omega t - kx) \tag{2.6}$$

$$a_z = -\omega^2\zeta_a e^{kz} \sin(\omega t - kx) \tag{2.7}$$

Equations (2.1) to (2.7) are part of the linear wave theory for propagating waves. They are derived assuming a horizontal sea bottom and a free-surface of infinite horizontal extent [1]. In real life, these assumptions are never completely fulfilled. However, taking into consideration that a floating offshore structure usually operates in deep water and far from shore, the equations might be assumed to give a good estimation of the conditions the structure is subjected to.

2.2 Irregular seas

The sea state a floating structure is exposed to do usually not look like a regular sine wave with constant amplitude propagating in one direction. The sea state is far more complex. However, due to linear superposition, irregular sea states can be simulated by adding together waves of different amplitudes, wavelengths and propagation directions [3].

This means that the surface elevation might be modeled by numerous longcrested waves, with different amplitudes, frequencies and phases. If considering a two-dimensional model, all the waves propagate along the same axis. The surface elevation might then be expressed as in equation (2.8), where N is the total number of wave components [3].

$$\zeta(x,t) = \sum_{n=1}^{N} A_n \sin(\omega_n t - k_n x + \varepsilon_n) \tag{2.8}$$

The phase angles, ε_n, are randomly chosen numbers between 0 and 2π and are constant with time. A_n, ω_n and k_n are the wave components *amplitude*, *frequency* and *wave number* respectively. The wave amplitude A_n can be expressed by a wave spectrum $S(\omega)$ as in equation (2.9) [3].

$$\frac{1}{2}A_n^2 = S(\omega_n)\Delta\omega \tag{2.9}$$

A wave spectrum describes the wave conditions after a constant velocity wind has been blowing for a long time. A specific ocean wave spectrum will typically be far more complex, and might for instance have two different peaks, one generated by waves coming in from far away, and one due to local winds creating more local waves [11]. However, a general wave spectrum is used to simulate irregular seas for analyses. The two most common standard wave spectra available are the Pierson-Moskowitz spectrum given in equation (2.10) [12] and the JONSWAP spectrum given in equation (2.11) [11].

$$S(\omega) = \frac{H_S^2 T_Z}{8\pi^2}\left(\frac{\omega T_Z}{2\pi}\right)^5 \exp\left[\frac{-1}{\pi}\left(\frac{\omega T_Z}{2\pi}\right)^{-4}\right] \tag{2.10}$$

$$\begin{aligned} S(\omega) &= \frac{\alpha g^2}{\omega^5}\exp\left[-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4\right]\gamma^r \\ r &= \exp\left[-\frac{(\omega-\omega_p)^2}{2\sigma^2\omega_p^2}\right] \end{aligned} \tag{2.11}$$

The main difference between the two spectra is that the Pierson-Moskowitz spectrum assumes a fully developed sea, while Hasselmann et al. [13] realized that this is never really the case, and introduced an extra factor γ^r to the Pierson-Moskowitz spectrum creating the JONSWAP spectrum. The constants α, ω_p, γ and σ might be found in for instance Stewart (2008) [11]. The significant wave height H_S is usually defined as $H_{1/3}$, the average of the 1/3 largest wave heights [2]. This wave height is said by Huss [2] to correspond approximately to the wave height perceived by the human eye when observing irregular seas, as the lower components get filtered out.

For a regular wave, the wave period is easy to identify. For irregular seas however, it is harder to define a constant wave period, as the resulting wave elevation is irregular. T_Z is defined to be the mean zero up-crossing period; the mean time interval between wave crests rising up past the still-water level.

For the analyses conducted in the work with this thesis, the Bretschneider spectrum shown in Figure 2.1 is used. The spectrum is also known as an ISSC spectrum and is a modified Pierson-Moskowitz spectrum [14]. The spectrum is based on the equation given in (2.12).

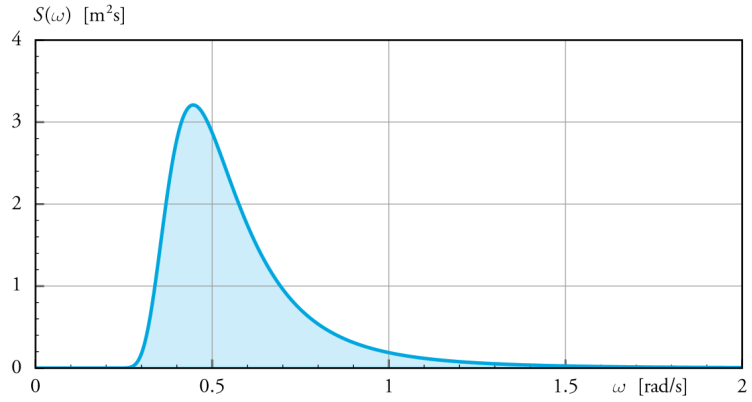

Figure 2.1: Bretschneider wave spectrum with $H_S = 4m$ and $T_Z = 10s$ [2]

$$S(H_S, T_Z, \omega) = \frac{H_S^2 T_Z}{8\pi^2} \left(\frac{2\pi}{\omega T_Z}\right)^5 \exp\left[-\frac{1}{\pi}\left(\frac{2\pi}{\omega T_Z}\right)^4\right] \tag{2.12}$$

For each wave component, the amplitude is found by equation (2.13), where the upper limit ω_u and the lower limit ω_l are defined in (2.14). This is another way to write equation (2.9).

$$A_n = \sqrt{2 \int_{\omega_l}^{\omega_u} S(\omega)\, d\omega} \tag{2.13}$$

$$\begin{aligned} \omega_l &= \omega_n - 0.5\Delta\omega \\ \omega_u &= \omega_n + 0.5\Delta\omega \end{aligned} \tag{2.14}$$

Huss(2010) [2] explains the wave spectrum and its features as follows:

> "The shape of the wave spectrum reflects the character of the irregular sea. A spectrum with large area represents a severe sea state with large waves, a spectrum that is spread out over a wide span of frequencies represent a very chaotic sea state with a mixture of short and long waves while a narrow spectrum represents a rather regular sea state (such as swell) where most of the energy is concentrated to waves with almost the same frequency." [2]

When generating an irregular sea state for simulation analyses, one technique is to divide the wave spectrum into N columns of width $\Delta\omega$, as shown in the upper half of Figure 2.2. The lower half of the figure shows how the wave components are summed up to give the resulting irregular sea. Another way to visualize the

transition from the wave spectrum in the frequency domain to the irregular wave elevation in the time domain is given in Figure 2.3.

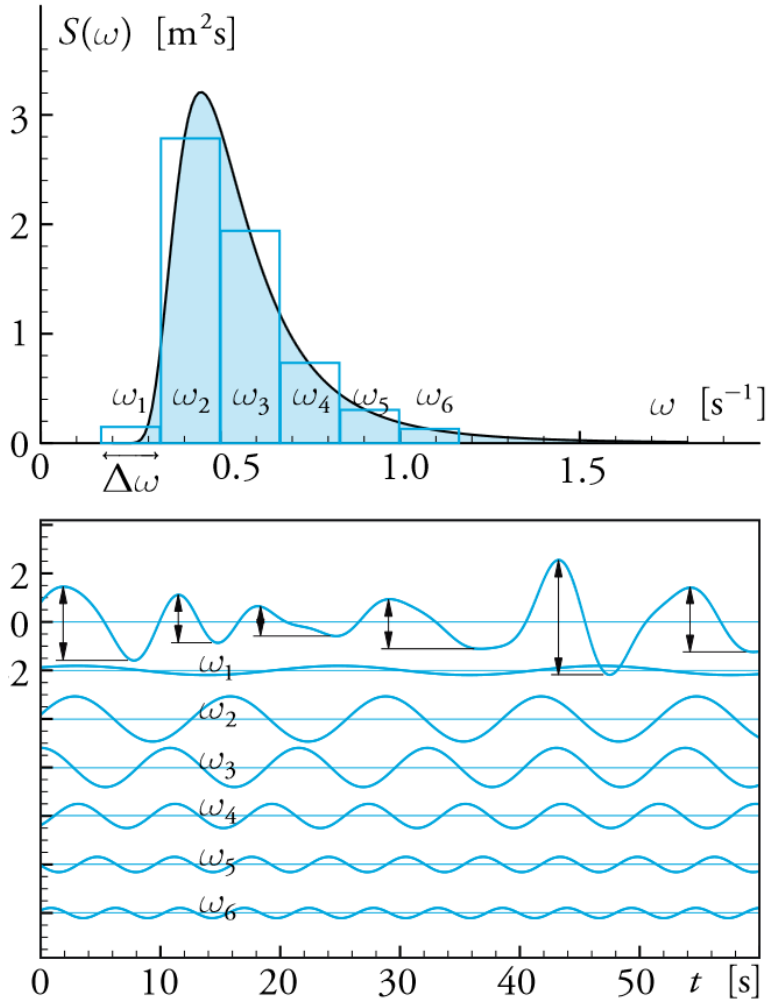

Figure 2.2: How a simplified irregular sea state can be simulated from a wave spectrum. The arrows indicate the quasi-random wave heights [2]

2.2.1 The return period

The return period is the total time span a unique wave pattern is generated before it is repeated. The ability to produce long return periods T_r is an indicator of the quality of the chosen wave generator. The simulated irregular sea will repeat itself

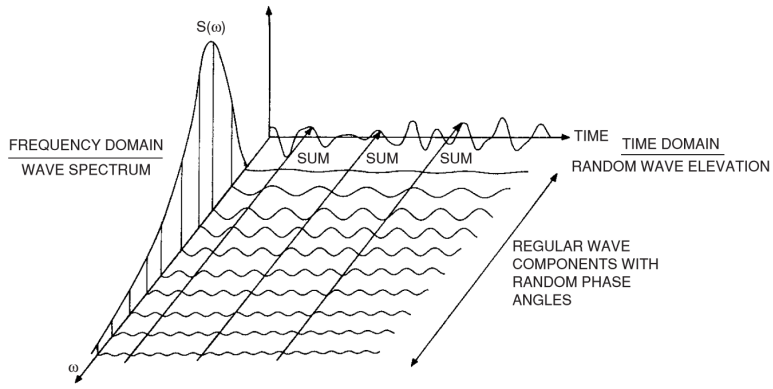

Figure 2.3: The connection between the frequency-domain and time-domain representation of waves [3]

when all the wave components have run an integer number of full periods at the same time, i.e. when the condition given in equation (2.15) is satisfied [2]. ω_1 is the frequency of the first component, i is the component number, $\Delta\omega$ is the frequency step between the wave components, T_r is the return period of the irregular sea pattern and k_i is an integer number [2].

$$\left[\frac{\omega_1}{2\pi} + (i-1)\frac{\Delta\omega}{2\pi}\right] T_r = k_i \tag{2.15}$$

For the Bretschneider spectrum, with $\omega_1 = 0.2$ and the total number of wave components $N = 100$, the frequency step $\Delta\omega = 0.02$. It can be shown that for $\omega_1 > 0$, condition (2.15) is satisfied if (2.16) is satisfied [2].

$$k_2 = k_1 \left(\frac{\Delta\omega}{\omega_1} + 1\right) \tag{2.16}$$

For $N = 100$, $k_1 = 10$ and $k_2 = 11$. The return period T_r will then be equal to 314.2 seconds. This is illustrated in Figure 2.4. It can be seen from equation (2.16) that a way of extending the return period is to make the ratio $\frac{\Delta\omega}{\omega_1}$ more irrational [2].

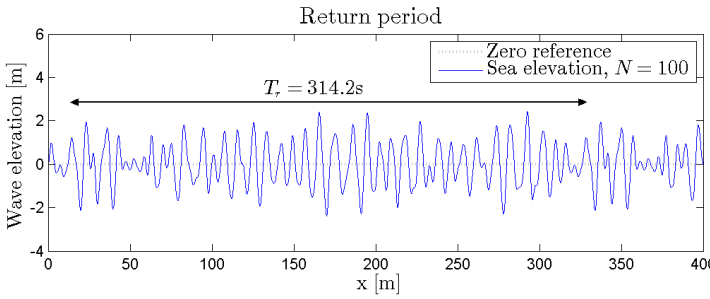

Figure 2.4: The return period for $N = 100$ wave components in a Bretschneider spectrum with $\omega_1 = 0.2$

Chapter 3

Frequency-domain motion response

The same way regular waves can be added together to simulate irregular seas, a structures steady-state response in irregular seas is the sum of the response to each wave component [1]. It is hence sufficient to analyze a structure in incident regular sinusoidal waves. In a frequency-domain analysis, steady-state conditions are assumed, meaning no transient effects due to initial conditions are considered.

The problem of finding the motion response to a given sea state might be divided in two sub-problems as shown in Figure 3.1. One sub-problem consists of the wave exciting forces on a structure retained from moving, as shown in the left part of the figure. The other sub-problem, shown in the middle of Figure 3.1, consists of the fluid reactive forces induced by a structure forced to oscillate in still water [9]. Due to the assumed linearity with respect to the wave amplitude, the loads from the first and second sub-problem can be added together to obtain the total hydrodynamic forces acting on a structure oscillating in regular waves [5].

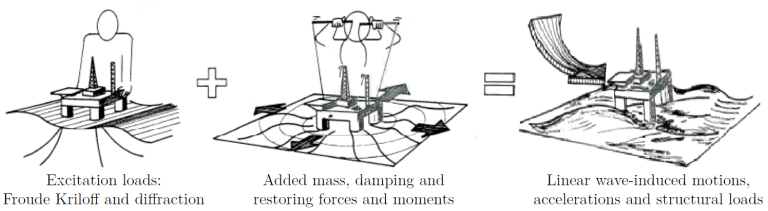

Figure 3.1: Motion response as the sum of two sub-problems [4]

3.1 Added mass, damping and restoring

One of the sub-problems, shown in the middle of Figure 3.1, consists of the forces on the structure from the fluid when the structure is forced to oscillate with frequency ω_i in still water. The oscillation frequency is the same as the frequency of the incoming waves in the other sub-problem. The hydrodynamic loads developed in this sub-problem are referred to as *added mass*, *damping* and *restoring* loads [1]. The oscillating structure will generate outgoing waves, and by integration of the oscillating fluid pressure forces over the structures surface, the forces and moments acting on the structure are found [1].

The hydrodynamic added mass and damping loads due to harmonic motion mode η_j can be written as in equation (3.1), where $k = 1, 2, \cdots, 6$. F_1, F_2 and F_3 are the force components in the x-, y- and z-direction, and F_4, F_5 and F_6 are the moment components about the corresponding axes.

$$F_k = -A_{kj}\frac{d^2\eta_j}{dt^2} - B_{kj}\frac{d\eta_j}{dt} \tag{3.1}$$

A_{kj} and B_{kj} are the added mass and damping coefficients, where k denotes the force-direction, and j denotes the direction of movement [5]. Six possible values for each of k and j gives a total of 36 added mass coefficients and 36 damping coefficients. Many of these coefficients are zero for a structure due to symmetry or no forward speed [1].

The terms *added mass* and *damping* might be misleading, in fact added mass terms do not have to have the dimension of mass. For instance A_{44} has the dimension of an inertia moment, and other terms might have the dimension of mass multiplied by length [1]. The names *added mass* and *damping* simply refers to the similarities with the terms in the equation for a simple oscillating single degree of freedom system. Equation (3.2) shows such a system, where M, B and C refer to the systems mass matrix, damping matrix and stiffness matrix respectively.

$$M\ddot{x} + B\dot{x} + Cx = 0 \tag{3.2}$$

The hydrodynamic added mass and damping terms multiplied by respectively acceleration and velocity will give forces, but they are not physical properties of a structure, like the structures mass is [5]. The hydrodynamic added mass is the force acting on the structure caused by the pressure field of the fluid being forced to oscillate by the moving structure. It is hence hydrodynamic pressure induced forces. The two terms in equation (3.1) can be understood as [5]:

- $-A_{kj}\ddot{\eta}_j$ = force in k-direction because of acceleration in j-direction
- $-B_{kj}\dot{\eta}_j$ = force in k-direction because of velocity in j-direction

A freely floating structure will have restoring forces following from hydrostatic and mass considerations. The restoring loads are caused by the changes of the displaced volume and hence changes in the buoyancy, due to rigid body motions [4]. The force and moment components may be written as in equation (3.3) [1].

$$F_k = -C_{kj}\eta_j \tag{3.3}$$

"The restoring coefficients can be obtained estimating the variation of the buoyancy loads due to the rigid motions" [4]. Considering an uncoupled heave motion for a structure with the xy- plane as a symmetry plane for the submerged volume, the restoring coefficient is given in equation (3.4) [1]. A_{WP} is the water-plane area.

$$C_{33} = \rho g A_{WP} \tag{3.4}$$

A positive restoring coefficient counteracts the motion, and hence helps bringing the structure back towards its initial position. Vice versa, a negative restoring coefficient will work destabilizing on the system [4]. "The restoring loads are important in fixing the natural periods of the body motions (...)" [4].

3.1.1 Strip theory

Finding the hydrodynamic coefficients for three-dimensional structures are not straightforward, and requires use of numerical techniques and computer programs [5]. Model testing is also possible. However, taking advantage of the structures geometry is useful. The underwater parts of both ships and platform pontoons and columns are slender structures, with one dimension dominating the other two. Dividing the underwater part of the structure in strips like in Figure 3.2 makes it possible to consider each strip in a two-dimensional stream in the cross-sectional plane [5]. This implies that the variation of the flow in the cross-sectional plane is much larger than the variation of the flow in the longitudinal direction [1]. This is not true at the ends of the structure, so the strip theory can only be applied for structures where the end effects are relatively small.

Integration of the strip components over the structures length as shown in equation (3.5), or summation of the results for all the strips, gives the hydrodynamic coefficients. This approach is referred to as strip theory.

$$A_{33} = \int_L A_{33}^{(2D)}(x)\,dx \tag{3.5a}$$

$$B_{33} = \int_L B_{33}^{(2D)}(x)\,dx \tag{3.5b}$$

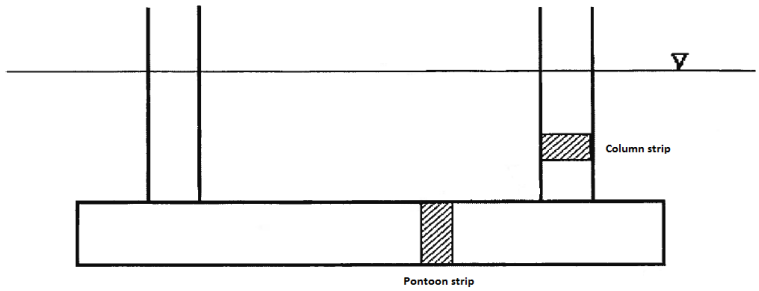

Figure 3.2: Underwater part of platform divided in strips [5]

Reduction of the problem from three to two dimensions makes it possible to find the solution analytically for some special very simple cross-sections like a circular cylinder [5]. In a general case, complicated body shapes makes numerical methods like source technique or conformal mapping necessary to estimate the two-dimensional added mass and damping coefficients.

Strip theory does have certain limitations. End effects become important if the length to beam ratio is too low, making strip theory a valid approach for finding hydrodynamic coefficients only for elongated bodies [3]. Also, strip theory is best applied to high frequency waves.

3.1.2 Frequency dependence of added mass and damping

As explained in the previous section, the added mass and damping terms depend on the cross-section of the submerged part of the structure. The added mass and damping components may also be strongly influenced by the oscillation frequency ω [1]. This is because the frequency of the waves affects the structures capability of generating waves. This might be shown by considering the far-field solution of the velocity potential [1], but this will not be done here.

In addition, the added mass and damping coefficients depend on the motion mode, meaning that for instance the added mass in heave and the added mass in roll are not necessarily equal. Figure 3.3 illustrates this. The left part of the figure shows added mass coefficients, and the right part shows damping coefficients. By comparing the two upper plots with the two lower plots, all for a circular cross-section oscillating at the free surface, it is easily seen that different motion modes yields different damping coefficients.

Again considering Figure 3.3, it is seen from the right column of plots that $B \to 0$ as $\omega \to 0$ and $\omega \to \infty$. This is because no free-surface waves are generated at these frequencies. "The reason is that the approximate free-surface conditions in these two cases state that there cannot be both a horizontal and a vertical velocity

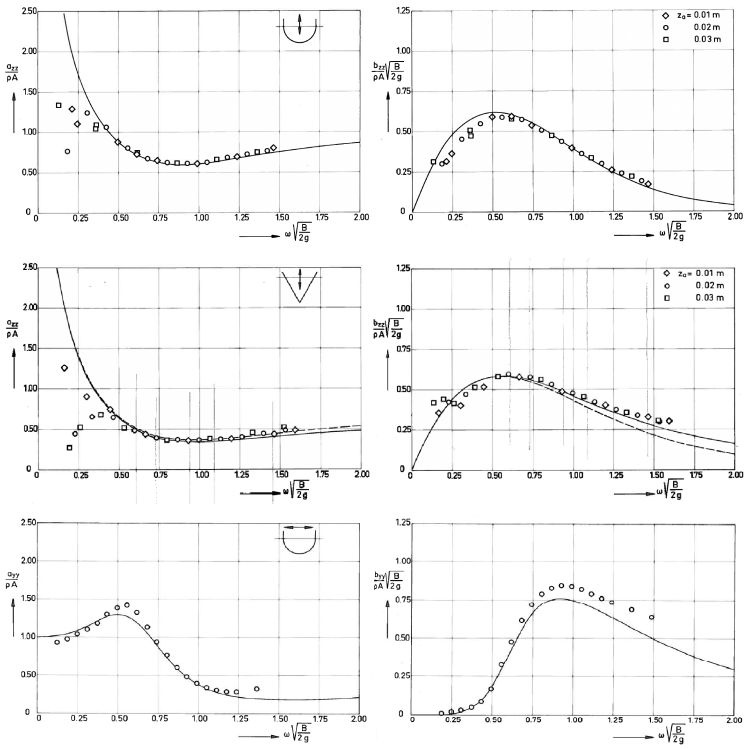

Figure 3.3: Top: Added mass (left) and damping (right) of a circular cylinder oscillating in heave. Middle: Added mass (left) and damping (right) of a pointed cross-section oscillating in heave. Bottom: Added mass (left) and damping (right) of a circular cylinder oscillating in sway.[6]

component on the free-surface. Both are necessary everywhere for there to be any propagating waves" [1]. In theory, $A_{33} \to \infty$ as $\omega \to 0$ for any two-dimensional surface piercing body in deep water. "But as $\omega \to 0$, finite water depth effects and 3D effects become important and make the added mass to be finite in reality" [4].

3.2 Excitation forces

The other sub-problem consists of the forces and moments working on the structure from the fluid when the structure is retained from moving under incident regular sinusoidal waves of frequency ω_i [5]. These loads are due to the unsteady fluid pressure, and are referred to as wave excitation loads. They consist of *Froude-Kriloff* and *diffraction* forces and moments.

The Froude-Kriloff loads are caused by the pressure field in the incident waves when they are undisturbed by the structure [1]. The diffraction forces are the forces that occur because the structures presence will change the undisturbed pressure field. The diffraction forces are found by solving a boundary value problem for the velocity potential, similar to the method used for finding the added mass and damping coefficients [1].

Different books operate with different names for the forces in this sub-problem. Faltinsen [1] operates with the names presented above, that *excitation loads* consists of *Froude-Kriloff* and *diffraction* forces and moments, while Newman [10] defines the names otherwise. Using Newmans notations, *diffraction* forces consists of *Froude-Kriloff* and *scattering* loads. Faltinsens way of naming the forces will be used in this thesis.

The forces on a relatively small structure can be written as a sum of components, as given in equation (3.6) [1]. Each component are given by equation (3.7). To be considered as a small volume structure, the characteristic length must be small compared to the wavelength λ. For instance, the diameter D of a vertical cylinder must be less than $\frac{\lambda}{5}$ to be considered small volume [1].

$$F = F_1 i + F_2 j + F_3 k \tag{3.6}$$

$$F_i = -\iint_S p n_i ds + A_{i1} a_1 + A_{i2} a_2 + A_{i3} a_3 \tag{3.7}$$

The first term in equation (3.7) are the *Froude-Kriloff* forces. p is the pressure in the undisturbed wave field, and n the unit normal vector to the structures surface and positive into the fluid. "The integral is over the average wetted surface of the body" [1]. The three last terms represents the *diffraction* forces.

Since the integral in equation (3.7) is to be taken over the wetted surface of the body, the junctions between the columns and pontoons must be respected, meaning that the Froude-Kriloff forces must be integrated directly [1].

3.2.1 Morison's equation

Wave forces acting on marine structures may be defined as either inertia or mass forces, or viscous forces. Inertia and mass forces are found from potential theory assuming inviscid fluid. Viscous forces are far more difficult to predict, as flow separation will occur and vortices will change the pressure field around the body and hence influence the hydrodynamic forces [7]. Empirical formulas are therefore often applied.

Faltinsen(1990) states that: "Morison's equation is often used to calculate wave loads on circular cylindrical structural members of fixed offshore structures when

viscous forces matter" [1]. Morison's equation gives the force dF on a strip of length dz of a rigid circular cylinder like a platform leg, and can be written as in equation (3.8) [7].

$$dF = \rho\frac{\pi D^2}{4}dzC_M a_n + \frac{\rho}{2}C_D D dz|u_n|u_n \tag{3.8}$$

In the case of a vertical cylinder, the force is positive in the wave propagation direction as seen in Figure 3.4. ρ is the mass density of the water, D is the cylinder diameter, and u_n and a_n are the undisturbed wave induced velocity and acceleration components normal to the cylinder axis at the midpoint of the strip [7]. C_M and C_D are the mass and drag coefficients, and are dependent of several parameters such as Reynolds number, the roughness number, and the Keulegan-Carpenter number. C_M and C_D must be empirically determined [1].

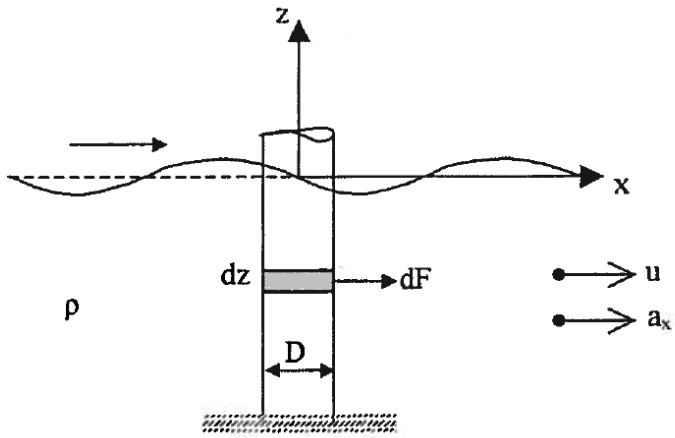

Figure 3.4: Force dF on strip of length dz of vertical cylinder [5]

In the analysis of floating offshore structures, the cylinder is allowed to move, and equation (3.8) must be modified to the expression given in equation (3.9). The velocity used in the equation is now the relative velocity between the fluid and the cylinder. a_c is the normal component of the cylinder acceleration.

$$dF = \rho\frac{\pi D^2}{4}dzC_M a_n + \rho\frac{\pi D^2}{4}dz(C_M - 1)a_c + \frac{\rho}{2}C_D D dz|u_{rel}|u_{rel} \tag{3.9}$$

Figure 3.5 indicates which kind of forces that dominate for different cylinder diameters compared with wavelengths and wave heights. "Morison's equation is normally assumed to be valid for a wavelength/diameter ratio above 5. Below this limit the presence of the cylinder will change the wave potential significantly, which

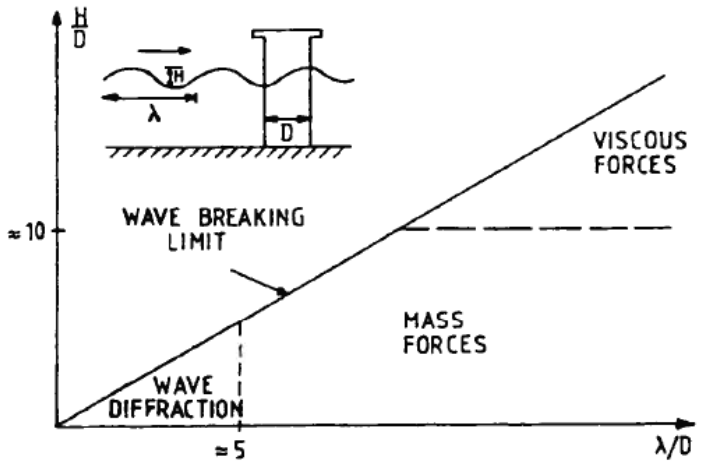

Figure 3.5: Relative importance of mass, viscous drag and diffraction force on vertical cylinder [7]

means that wave diffraction forces become important and Morison's equation should not be used" [7].

Another limitation to Morison's equation is that it only gives a good estimation of the forces in the case of a uniform flow. For orbital flow, for instance encountered by a cylinder subjected to waves, Morison's equation does not give a good representation of the forces as a function of time [1]. Morison's equation is not perfect, but it is the best prediction known today of the forces caused by the complicated flow picture that occurs for separated flow around marine structures [1].

3.3 The equations of motion

"When the hydrodynamic forces have been found it is straightforward to set up the equations of rigid body motions. This follows by using the equations of linear and angular momentum. [1]. "Obtaining the hydrodynamic forces is by no means trivial" [3]. The steady-state rigid-body sinusoidal motions of a floating body are given as in equation (3.10) [3].

$$\sum_{k=1}^{6}[(M_{jk}+A_{jk})\ddot{\eta}_k + B_{jk}\dot{\eta}_k + C_{jk}\eta_k] = F_j e^{i\omega_e t}, \quad j = 1, ..., 6 \tag{3.10}$$

Chapter 4

Time-domain motion response

When analysing a structure in irregular seas, the structure is subjected to many different excitation frequencies. Because added mass and damping depend on the frequency of the waves, as discussed in section 3.1.2, the system in equation (3.10) cannot be used directly [3]. When only the steady-state solution is of interest, this problem is circumvented by finding the response to each regular wave component, and then adding together the structures motion responses for all the components [3].

In some cases, the transient effects might be too important to be neglected. Faltinsen [3] lists waves generated by passing ships, coupling between nonlinear sloshing in a ship tank and ship motions, and wet-deck slamming on a catamaran in waves as examples of situations where transient effects are crucial.

For these transient effects to be considered, a memory function must be included in the equations of motion to represent the time-domain effects [3].

4.1 The memory function

Cummins (1962) [15] and Ogilvie (1964) [16] discussed how the equation of motion must be reformulated to be valid in the time domain. Ogilvie [16] states that: "In a sense, we find that the existence of the free surface causes the physical system to have a 'memory': What happens at one instant of time affects the system for all later times" [16].

Cummins [15] presented a new mathematical expression as a representation of a ships response to waves. The linear time-domain response in heave decoupled from all other motions, are given in equation (4.1) [3].

$$(M + A_{33}(\infty))\ddot{\eta}_3 + B_{33}(\infty)\dot{\eta}_3 + C_{33}\eta_3 + \int_0^t h_{33}(\tau)\dot{\eta}_3(t-\tau)\,d\tau = F_3(t) \tag{4.1}$$

By way of comparison, the steady-state response given in equation (3.10) will for heave motion decoupled from all other motions look like equation (4.2).

$$(M + A_{33})\ddot{\eta}_3 + B_{33}\dot{\eta}_3 + C_{33}\eta_3 = F_3(t) \tag{4.2}$$

The structures mass M and restoring C_{33} are the same in the two equations [3]. The differences are an additional integral term in equation (4.1), and that equation (4.1) uses the mean infinite-frequency added mass and damping coefficients [3]. This means that the added mass and damping coefficients in equation (4.1) are independent of frequency. The integral term is referred to as a convolution integral. Cummins [15] states that: "The response is given as a convolution integral over the past history of the exciting force with the impulse response function appearing as the kernel" [15].

$h_{33}(t)$ is the impulse response function, or the retardation function, and section 4.2 explains how this function might be found.

4.2 Finding the retardation function

The retardation function $h(t)$ may be calculated based on either the added mass or the damping coefficient, as shown in equation (4.3). The two different approaches will be further investigated in the rest of this section.

$$h_{ij}(t) = -\frac{2}{\pi}\int_0^\infty \omega(A_{ij}(\omega) - A_{ij}(\infty))\sin(\omega t)\,d\omega \tag{4.3a}$$

$$= \frac{2}{\pi}\int_0^\infty (B_{ij}(\omega) - B_{ij}(\infty))\cos(\omega t)\,d\omega \tag{4.3b}$$

4.2.1 The added mass related approach

Calculation of the retardation function using equation (4.3a) requires information of A_{ij} to be known for all frequencies [3]. The added mass coefficient strongly depends on the geometry of the cross-section of the submerged body. Figure 4.1 shows the

added mass coefficients in heave for a rectangular cross-section for three different beam to draft ratios. The upper lines are for $B/D = 8$, the middle lines are for $B/D = 4$, and the lower lines are for $B/D = 2$.

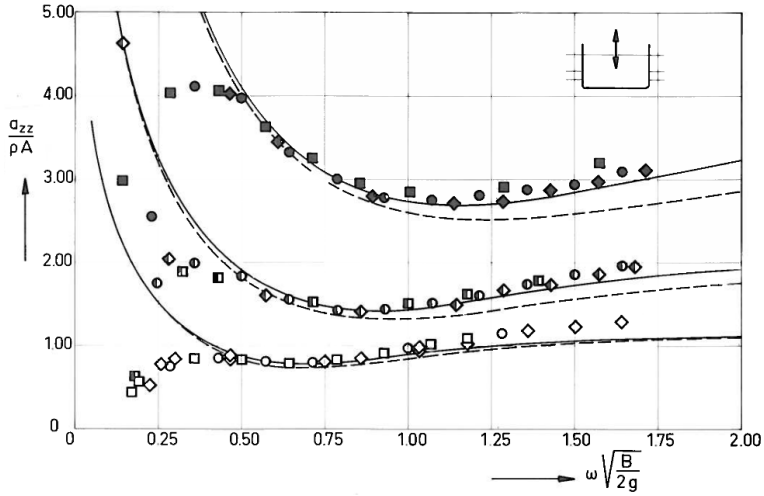

Figure 4.1: Added mass coefficients in heave as function of frequency [6]

As seen in Figure 4.1, low frequencies gives large added mass coefficients. The figure does not give any clear information as to what happens to the added mass coefficient as $\omega \to 0$, but the figure indicates that the values grows exponentially. The figure also indicates that the added mass coefficients might stabilize at a fixed value for frequencies larger than a certain value. For instance for a beam to draft ratio of 2, the lower line in Figure 4.1, the added mass coefficient appears to stabilize at approximately 1.1 for frequencies larger than approximately 1.9. As will be discussed further in 4.2.3, this is not sufficient information of the added mass coefficient for all frequencies to be used as a basis for calculation of the retardation function.

4.2.2 The damping related approach

The other possible approach for calculation of the retardation function is based on the damping of the structure, and is given by equation (4.3b). This approach will, analogous to the added mass approach, require information of B_{ij} to be known for all frequencies [3]. Like for the added mass coefficients, there are a strong dependency to the geometry of the cross-section of the submerged body.

Figure 4.2 presents damping coefficients for a rectangular cross-section oscillating in heave. The upper lines represents a beam to draft ratio $B/D = 8$, the middle lines represents $B/D = 4$ and the lower lines represents $B/D = 2$. The main difference between this figure, and the figure for the added mass coefficients 4.1, is

that unlike the added mass coefficients, the damping coefficients are defined also for low frequencies. The damping coefficients go to zero for really small and for large frequencies. This is because these frequencies causes the structure to not generate any waves, as mentioned in section 3.1.2.

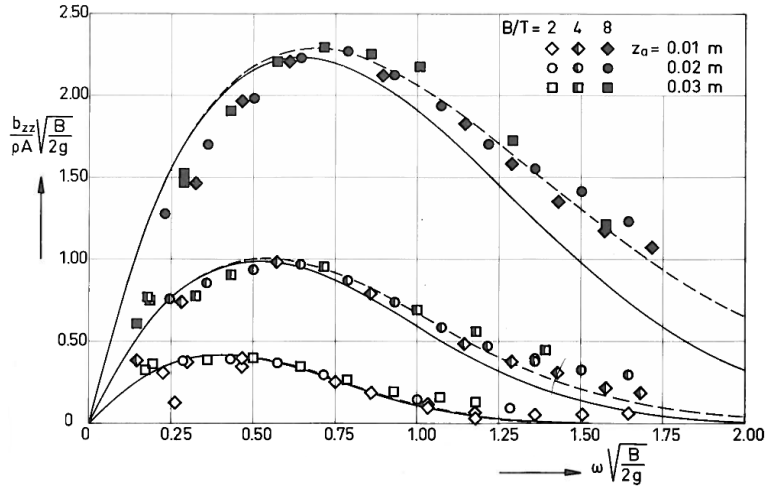

Figure 4.2: Damping coefficients in heave as function of frequency [6]

4.2.3 Challenges related to the two approaches

The main challenge in connection with calculating the retardation function is connected to finding sufficiently detailed information of either the added mass or damping coefficients at all frequencies. Rognebakke (2002) [17] describes this process as crucial. Both approaches where tested in this thesis, using the procedure explained in detail for damping in chapter 5.2.2. Both attempts at finding the retardation function are based on the information given respectively in Figure 4.1 and 4.2. A rectangular cross-section with a beam to draft ratio of 2 are assumed. The resulting functions are shown in Figure 4.3 and 4.4.

It is easily seen that the two approaches in this case does not produce the same resulting function. To decide which, if any, is the correct one, comparison is made to the impulse response function presented by Faltinsen(2005) [3]. His function is given in Figure 4.5. By inspecting the three functions, it seems to be a reasonable accuracy in Figure 4.3, the one calculated utilizing the damping-related equation. The function found by the added mass approach, Figure 4.4, is incorrect. This is probably due to the inadequate information of the added mass coefficients for large and very small frequencies. The approximated function used for the line representing a beam to draft ratio of 2 in Figure 4.1, assumed a value of 5 for the added mass coefficient at $\omega = 0$. This is a large understatement, in fact Kotic and

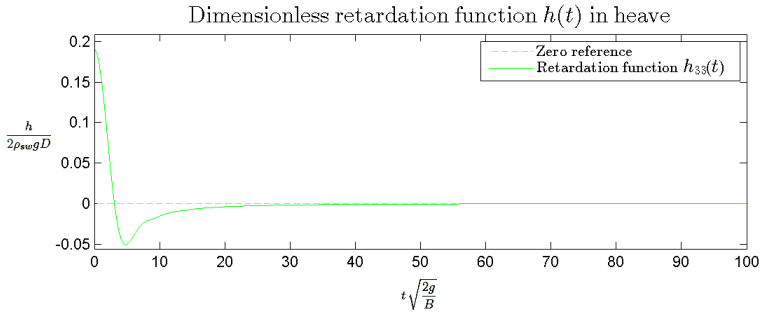

Figure 4.3: Retardation function calculated using damping (equation (4.3b))

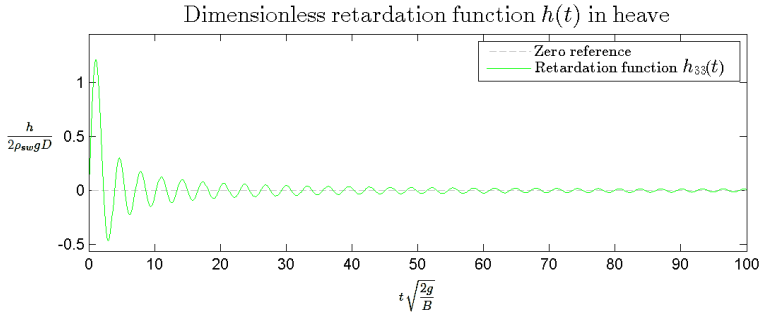

Figure 4.4: Retardation function calculated using added mass (equation (4.3a))

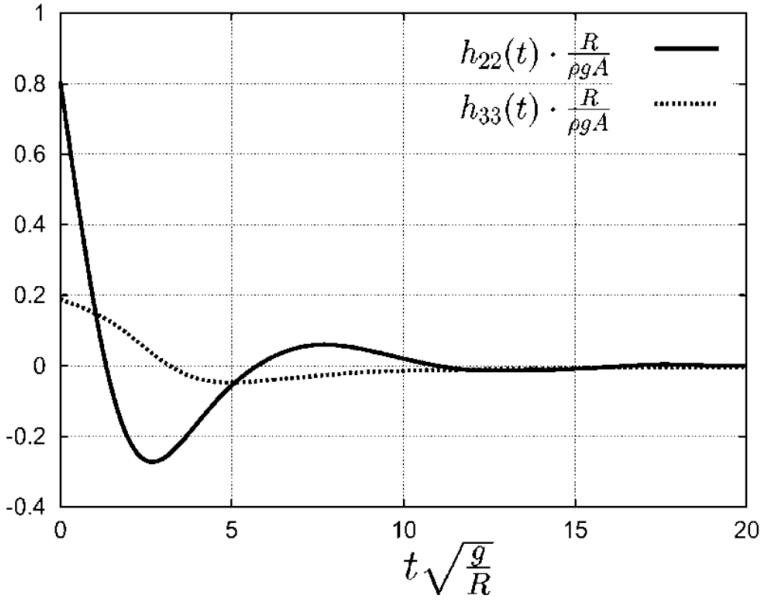

Figure 4.5: Retardation function in heave (dotted line) for a rectangular cross-section of $B/D = 2$ [3]

Mangulis (1962) [18] have proven that $A_{33} \to \infty$ as $\omega \to 0$ for all two-dimensional surface-piercing structures. An insufficient accuracy of the value the added mass coefficient seemingly stabilizes at might further add to the error.

4.3 Verification of accuracy by inverse calculation

The retardation function (4.3) for uncoupled heave motion is given by equation (4.4).

$$h_{33}(t) = -\frac{2}{\pi} \int_0^\infty \omega(A_{33}(\omega) - A_{33}(\infty)) \sin(\omega t)\, d\omega \tag{4.4a}$$

$$= \frac{2}{\pi} \int_0^\infty (B_{33}(\omega) - B_{33}(\infty)) \cos(\omega t)\, d\omega \tag{4.4b}$$

The accuracy of the retardation function found might be checked by performing an inverse calculation as explained by Rognebakke (2002) [17]. Equations (4.5) and (4.6) show how the obtained retardation function might be used to calculate

respectively the added mass or damping coefficient in heave for all frequencies [16].

$$A_{33}(\omega) = A_{33}(\infty) - \frac{1}{\omega}\int_0^\infty h_{33}(t)\sin(\omega t)\,dt \tag{4.5}$$

$$B_{33}(\omega) = B_{33}(\infty) + \int_0^\infty h_{33}(t)\cos(\omega t)\,dt \tag{4.6}$$

This enables for comparison of the initial $A_{33}(\omega)$ or $B_{33}(\omega)$ with the result from respectively (4.5) or (4.6). A good correspondence between the initial coefficients and the coefficients found by the inverse calculation indicates a good accuracy in the retardation function $h_{33}(t)$ [17].

As the added mass related approach was seen from Figure 4.4 not to work, the procedure of inverse calculation was tested on the retardation function found by utilizing the damping related approach. The result is shown in Figure 4.6. The red line, found from the inverse calculation, is seen not to coincide with the initial damping coefficients shown with the black line. This indicates a bad accuracy of the retardation function, and is most likely connected to the method used for finding the function describing the initial damping coefficients. This procedure is explained in chapter 5.2.2. However, the two lines do show similar trends, they both start at zero for zero frequency and seemingly converges to zero for large frequencies. Thus, the retardation function found using the damping related approach is used in the calculation of the convolution integral in this thesis, despite the deficient accuracy.

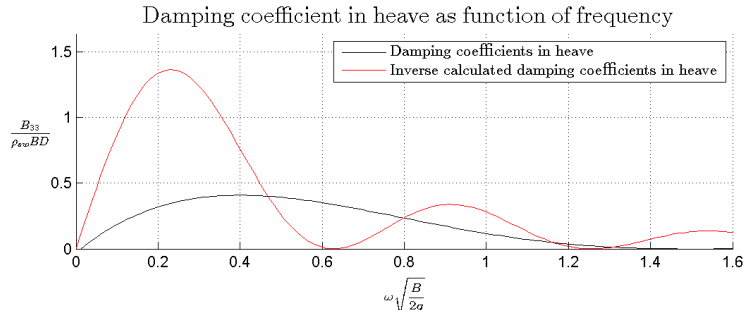

Figure 4.6: Initial damping coefficients compared to the damping coefficients found by inverse calculation

4.4 The convolution integral

A general mathematical description of what a convolution integral is, and which mathematical operations that are valid when dealing with convolution integrals, are given in appendix A.

Rognebakke (2002) states that: "The convolution integral takes care of the memory effect of past motions" [17]. As mentioned in the beginning of this chapter, this is because the free surface of the water represents a 'memory' of the system. "Each occurrence is, in fact, dependent on all preceding occurrences." [9]. Considering equation (4.7), the integral term from equation (4.1), it can be seen that the integral limits are 0 and t.

$$\int_0^t h_{33}(\tau)\dot{\eta}_3(t-\tau)\,d\tau \tag{4.7}$$

This means that the integral is supposed to be evaluated from the beginning of time and up to this moment. This would require enormous computer resources and is hence neither practically possible nor necessary. By investigating the $h(t)$-term plotted in Figure 4.5, it is observed that the retardation function appears to be non-zero only for a limited time range. This implies that a "cut-off" value might be introduced, limiting the time history for the integral term. Equation (4.8) shows the convolution integral over a limited time span.

$$\int_{t-t_1}^t h_{33}(\tau)\dot{\eta}_3(t-\tau)\,d\tau \tag{4.8}$$

$t - t_1$ defines the "cutoff" value, with t_1 the amount of time in the time span included in the calculation. In the following, equation (4.7) will be referred to as the full convolution integral, while equation (4.8) will be referred to as the limited or reduced integral. A further investigation of the time span that needs to be included in the calculation of the memory effects are conducted in chapter 5.

Chapter 5

Analyses in MATLAB

To illustrate the time history of the memory functions, a MATLAB program is written. A simplified system is tested, to help estimate the amount of memory that needs to be included in the convolution integral (4.8) to obtain a certain accuracy compared to the complete integral. What is considered a satisfactory accuracy will be discussed in chapter 6.3 after the results of the analyses are presented. One major simplification in the analysis is the geometry of the structure tested. The 'platform' is considered to be a two-dimensional rectangular cross-section oscillating at the free surface. Another important simplification is that the system analysed is assumed to oscillate only in heave, decoupled from other motions. The system hence becomes a single degree of freedom mass-spring system.

5.1 Single-degree mass-spring system

Wave-induced motions on a floating body might to a large extent be described by a mass-spring system with damping [3]. Faltinsen [3] states that coupling between different modes of motions are of importance, but a single degree of freedom system is sufficient to exemplify essential features of the motions. Equation (5.1) represents the steady-state motion in heave for a floating structure.

$$(m + a)\ddot{y}(t) + b\dot{y}(t) + cy(t) = f(t) \tag{5.1}$$

Here m is the structures mass, a is the added mass in heave, b is the damping coefficient (according to [3] caused by, among others, wave radiation due to heave oscillations), c is the restoring coefficient due to changes in the buoyancy etc, and $f(t)$ is the excitation force.

The oscillating body is assumed to be positioned at $x = 0$, which means that the surface elevation from equation (2.8) might be expressed as in equation (5.2) [3].

Similarly the vertical velocity component of the wave might be expressed as in equation (5.3)

$$\zeta(t) = \sum_{n=1}^{N} A_n \sin(\omega_n t + \varepsilon_n) \tag{5.2}$$

$$w(t) = \sum_{n=1}^{N} \omega_n A_n \cos(\omega_n t + \varepsilon_n) \tag{5.3}$$

5.2 The MATLAB program

A MATLAB code is written to find two vectors, $\dot{\eta}(t-\tau)$ and $h_{33}(\tau)$ for time from 0 to a specified end time t, with time step dt, and numerically integrate the convolution of the two vectors from 0 to t. The program then calculates the integral from $t-t_1$ to t, where t_1 is a given time interval, and compare the deviation between this integral and the first one. The MATLAB program with all associated functions are found in the appendix.

5.2.1 Simulating waves

Three quite similar, but still slightly different functions are written to calculate the $\dot{\eta}(t)$ vector. The program asks the user to choose a wave scenario to be used in the analysis, and asks for the relevant input information for the chosen wave type. The chosen function will return a vector of vertical velocities for different t-values. The velocities are calculated at $x = 0$, since this is considered to be the floating structure's location in this simplified analysis.

The transfer function $|\eta_3|/\zeta_a$ is assumed to be equal to 1 for all frequencies. This means that the structure analysed behaves like a cork floating on the water, i.e. the structures vertical velocity equals the waves vertical velocity component [3].

regWave

The first wave-function is called *regWave*, and is used for creating regular waves. It takes the end time $maxT$, time increment dt, wave amplitude ζ_a and wave period T as input from the user. The function then calculates the wave profile and vertical velocity vectors at $x = 0$, using equation (2.2) and (2.5) respectively. Finally the function makes an animated plot of the wave and the corresponding vertical velocity vector. A snapshot of the animation is given in Figure 5.1 for a wave with amplitude $\zeta_a = 5m$ and period $T = 5s$. As seen, the wave profile at the given time instant is $T/4$ out of phase with the vertical velocity component. The vertical velocity is zero under the wave crests and troughs, and reaches its maximum absolute values

at the wave profile's points of intersection with the still water level. This is as expected when comparing to the theory of regular waves presented by for instance Faltinsen(1990) [1]. The vector containing vertical velocities for all elements of t at $x = 0$ is returned to the main program.

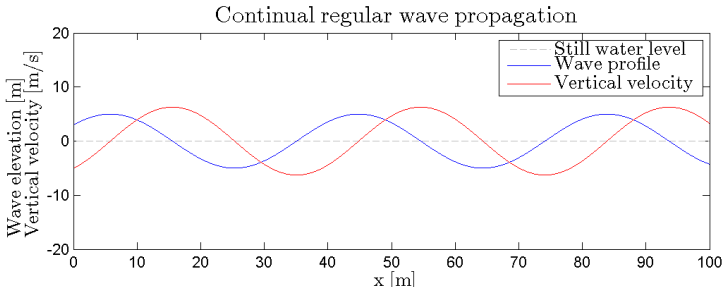

Figure 5.1: Snapshot of regular wave propagation with vertical velocity component

singleRegWave

The second wave function is *singleRegWave*. The function works precisely like the *regWave* function, except that it only creates one single wave excursion. The vectors of wave elevation and vertical velocities at $x = 0$ hence contains non-zero values only for one wave period. This function does not represent a plausible wave scenario, but it is a practical tool for understanding the convolution integral.

Like the *regWave* function, *singleRegWave* produces an animated plot of the wave propagation, and a snapshot of this plot is given in Figure 5.2. The function returns the vector of vertical velocity components at $x = 0$ to the main program.

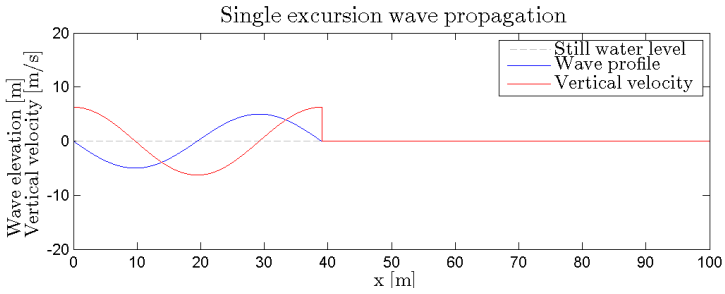

Figure 5.2: Snapshot (at $t = 0$) of regular wave propagation for a single wave excursion with its vertical velocity component

irrSea

The third wave function represents the most realistic representation of a sea state, and is called *irrSea*. Instead of getting a single wave amplitude and period from the user, the function requests a significant wave height H_S and a zero up-crossing period T_Z. A Bretschneider wave spectrum is then used to find a set of wave amplitudes and frequencies to be summed up to represent an irregular sea state, as explained in chapter 2. Finally, an animated plot showing the sea state is made, and a vector of vertical velocities at $x = 0$ is returned to the main program. A snapshot of an irregular sea state is given in Figure 5.3.

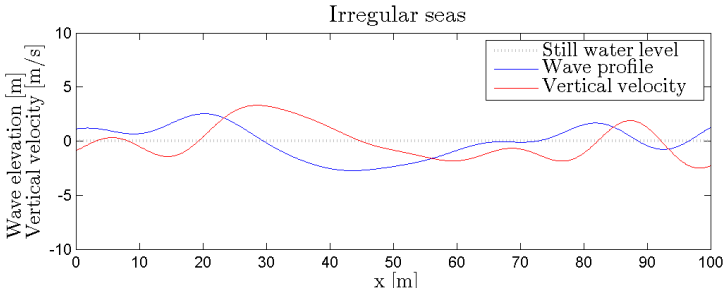

Figure 5.3: Snapshot of the irregular sea elevation and its vertical velocity component

5.2.2 Finding the retardation function

After the $\dot{\eta}(t)$ term is found, the next step is to find the retardation function. As explained in chapter 4.2, the retardation or impulse response function might be found using one of two approaches, equation (4.3a) or equation (4.3b). Both approaches where tested in this thesis, and only one gave a reasonable accuracy compared to the expected result. As shown in figure 4.6, the verification of accuracy for the best resulting function revealed inaccurate results also for this approach. This is most likely due to inaccurate information of the frequency-dependent added mass and damping coefficients introduced by the procedure used. The procedure for finding the retardation function will be presented here only for the damping related approach, but the procedure used for the added mass approach is identical.

Figure 4.2 constituted the basis of the calculation. The figure was digitized using a program called *GetData Graph Digitizer*. Figure 5.4 shows the process of digitizing; the program traces the specified line and returns a set of x- and y-values. For the frequency-dependent damping coefficients, the x- and y-values were then copied over to a spreadsheet. A scatter diagram was drawn based on the values, and finally, a polynomial equation was found as an approximated function of the coefficient. Figure 5.5 shows the scatter diagram and corresponding function found for the frequency dependent damping coefficients in heave. This function is used by the

MATLAB function *retardationD* to calculate equation (4.3b).

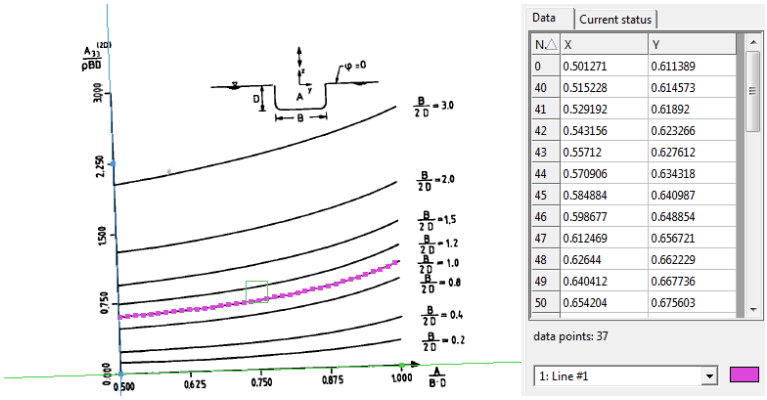

Figure 5.4: The process of digitizing a graph. Here shown for finding $A_{33}(\infty)$ for a rectangular cross-section

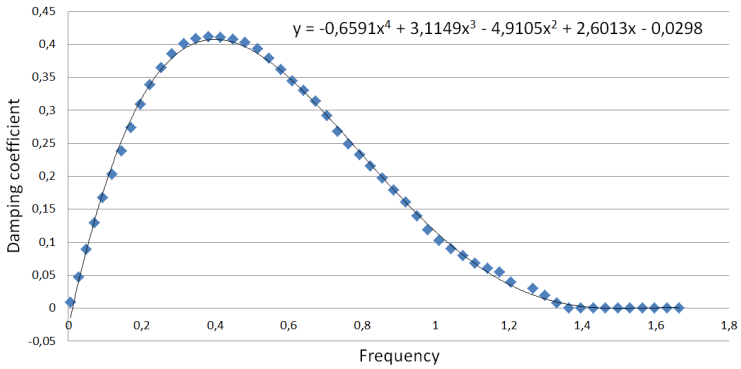

Figure 5.5: Damping coefficient B_{33} as function of frequency

retardationD

The MATLAB function *retardationD* is used to calculate the $h_{33}(t)$ vector. It takes the time increment dt and end time $maxT$ as input. The function from Figure 5.5 is then used as a basis for calculating equation (4.3b). Finally, a plot showing the retardation function is made, as shown in Figure 4.3. A vector containing the retardation values for all elements of t are returned to the main program.

5.2.3 Performing the convolution

Finally, the actual convolution of the two vectors is performed. This is done in a function called *contConvolution*. There exists a predefined MATLAB function *conv* that returns the convolution of two vectors given as input. This function is not used in this thesis since the supereminent objective is to prepare for including a corresponding function in USFOS, which is written in Fortran.

contConvolution

The last function, *contConvolution*, takes the vertical wave velocity and retardation vectors as input, in addition to the time increment dt and end time $maxT$. The function defines a new 'time' variable τ, ranging from $-maxT$ to $maxT$. The two input vectors are expressed in terms of τ, creating $\dot{\eta}(\tau)$ and $h_{33}(\tau)$. The vertical velocity vector is then mirrored around the y-axis, creating $\dot{\eta}(-\tau)$ for plotting of the convolution process. The time span to be included in the limited integral is defined, and the corresponding number of steps in the time vector is calculated. The lower integral limit is calculated based on the chosen time span. A loop iterates over each element in the time vector t. For each iteration, the full integral from 0 to t is calculated, as well as the reduced integral only including the the number of seconds defined as the time span in the integral. The deviation between the two integrals for each value of t is stored in a vector. For $t < t_1$, where t_1 is the time span, the full time interval is included in the reduced integral, causing the deviation to be zero.

Finally, an animated plot showing the process of convolution is made. Figure 5.6 shows a snapshot of this animation. For better visualization of the process, the two vectors $\dot{\eta}(t)$ and $h_{33}(t)$ are scaled (in this figure) to be of the same order of magnitude. This is possible since the main interest here is the ratio of the error between the two integrals, and not the actual numerical values. The upper part of the figure shows the shifting of $\dot{\eta}(t-\tau)$ through a coordinate system fixed in τ. In the middle of the figure, the product of $\dot{\eta}(t-\tau)$ and $h(\tau)$ is shown for all values of time t. The area under the graph is shaded, and corresponds to the value of the convolution in time t. The vertical lines show the integral limits, the full integral is calculated from time $t = 0$ (left line) to time t. The limited integral only include the area between the middle and right vertical lines, in Figure 5.6 corresponding to 8 seconds. Lastly, in the two calculated integrals are plotted in the lower part of the figure.

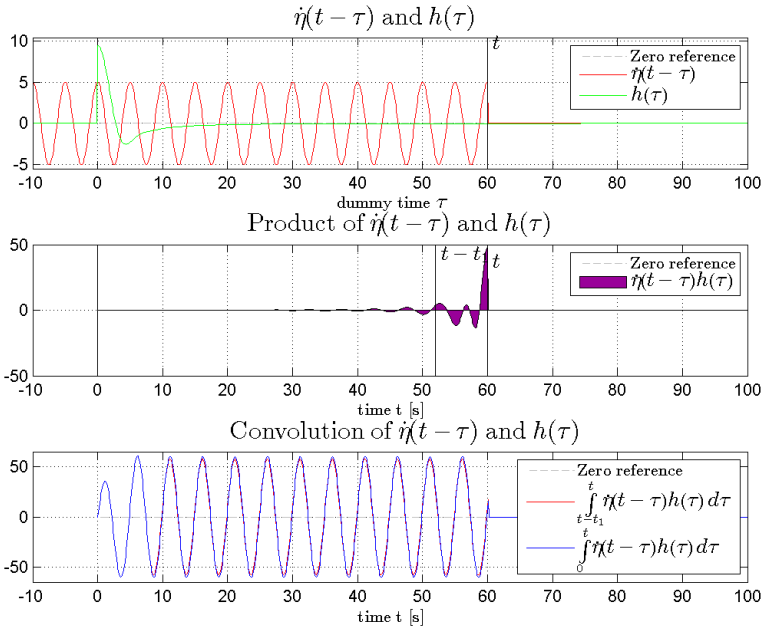

Figure 5.6: Snapshot of the animation of the convolution process for a rectangular cross-section oscillating in heave in regular waves. $t = 60$s, time span $t_1 = 8$s

Chapter 6

Required memory

The main results of the simulations conducted in MATLAB are presented in this chapter. The different wave scenarios are tested, and the effect of different wave parameters on the memory required to achieve a specific accuracy are investigated.

6.1 Regular waves

Table 6.1 shows the maximum deviation between the two integrals calculated with different time spans included in the 'memory' for the reduced integral. In this analysis, a regular wave with amplitude $\zeta_a = 4$ meters and period $T = 5$ seconds are used. A total of 100 seconds are included in the full integral, and a time increment $dt = 0.01$ seconds is used. The first column shows how many seconds are included in the 'memory'. The second column shows the maximum absolute deviation of the second integral, and the third column shows the maximum absolute deviation as a ratio of the maximum value from the full integral.

Only including a few seconds in the 'memory' is seen to give bad estimations of the full convolution integral. For one second, the maximum deviation amounts to 65% of the maximum value in the full convolution integral. However, a few seconds extension of the time interval significantly reduces the deviation. Including a time span of four seconds gives a maximum deviation of 20%, while 10% deviation is reached with less than six seconds included in the 'memory'. As the time span included in the reduced integral increases, it becomes more and more expensive in seconds to achieve a certain percentual improvement of the limited integral. Figure 6.1 illustrates this effect. The upper part of the figure shows the obtained deviation, while the two lower plots shows the percentual deviations.

As seen from table 6.1, a time span of 6 seconds included in the limited convolution integral results in an accuracy of 10%. Figure 6.2 shows the full integral and the

Time span included	**Maximum deviation**	**Relative deviation**
1	0.82370	0.67807
2	0.37273	0.30683
3	0.26563	0.21867
4	0.25403	0.20912
5	0.17585	0.14476
6	0.12061	0.09929
7	0.10027	0.08254
8	0.08233	0.06778
9	0.06309	0.05194
10	0.05134	0.04226
11	0.04496	0.03702
12	0.03800	0.03128
13	0.03136	0.02582
14	0.02755	0.02268
15	0.02483	0.02044
16	0.02147	0.01767
17	0.01860	0.01531
18	0.01704	0.01403
19	0.01557	0.01281
20	0.01369	0.01127
25	0.00878	0.00722
30	0.00626	0.00515
35	0.00450	0.00371
40	0.00344	0.00283
45	0.00280	0.00231
50	0.00224	0.00184
55	0.00179	0.00148
60	0.00157	0.00129
65	0.00134	0.00110
70	0.00111	0.00091
75	0.00099	0.00082
80	0.00091	0.00075
85	0.00078	0.00064
90	0.00066	0.00054
95	0.00059	0.00049
100	0.00000	0.00000

Table 6.1: Maximum deviation between the two integrals for different time spans included in the 'memory'. End time $t = 100$s and time increment $dt = 0.01$s. Tested with a regular wave with amplitude $\zeta_a = 4$m and period $T = 5$s.

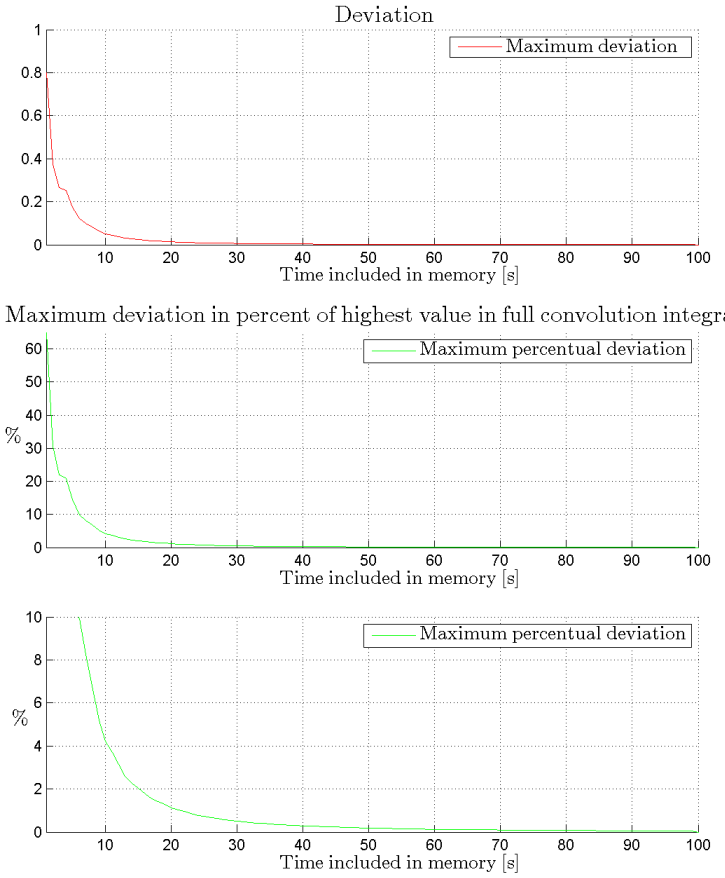

Figure 6.1: The deviations obtained while calculating the convolution integrals for various time spans in regular waves of amplitude $\zeta_a = 4$m and period $T = 5$s

reduced integral for this time span. The blue line for the full integral completely covers the reduced integral line for the first 6 seconds, as should be expected. After 6 seconds, the reduced integral deviates from the full integral, and an oscillating error or deviation occurs.

Investigating the upper part of Figure 6.2, the maximum deviation appears to be less than 10% because the lines seem to coincide well. The maximum error can however be observed by enlarging the figure. This is done in Figure 6.3, where it is seen that the maximum deviations do not occur at the largest absolute values of the full integral. The largest deviations occur due to the shift, or 'phase difference', between the two integrals. At the points of maximum absolute values in the full convolution integral, the reduced integral is seen to give a better approximation to the full integral than the value from table 6.1 indicates. In fact, the reduced integral overestimates the maximum value of the 'memory function' by 1.3%.

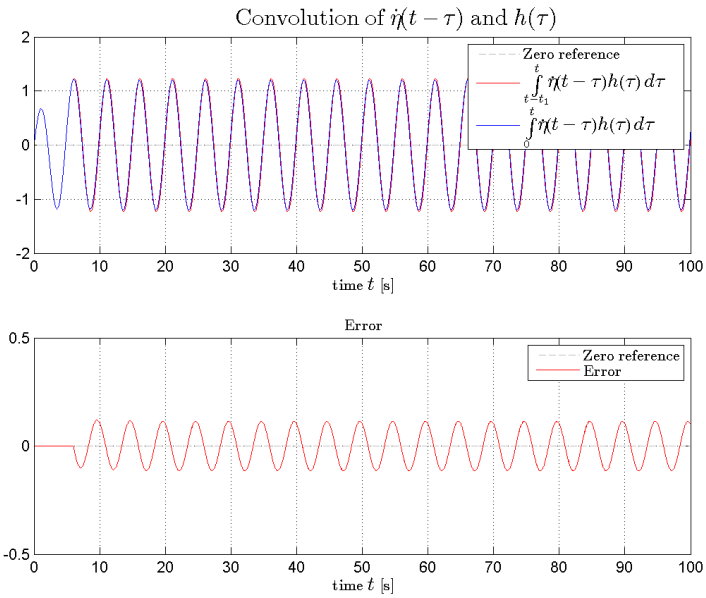

Figure 6.2: Full and reduced integral for a time span of 6 seconds

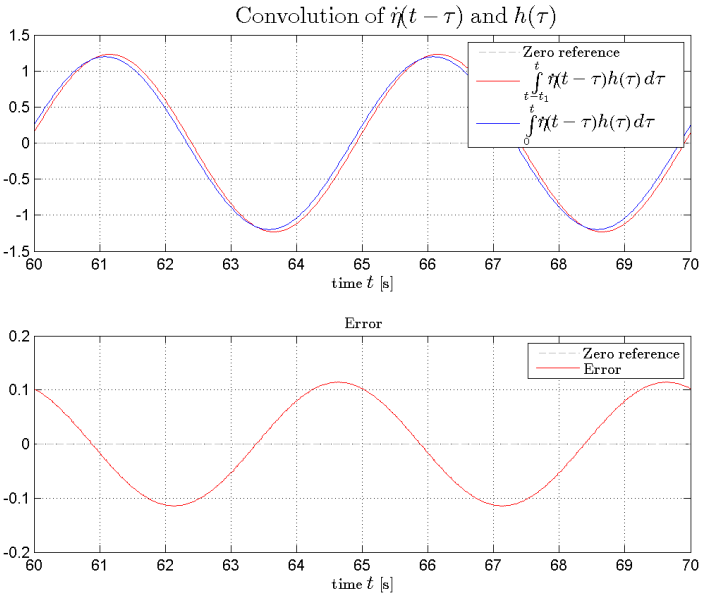

Figure 6.3: Enlarged part of Figure 6.2 showing the full and reduced integral for a time span of 6 seconds

6.1.1 A single wave excursion

The 'memory function' obtained for an analysis in regular waves, such as the analysis conducted in Figure 6.2 is seen to be oscillating with constant amplitude and period after a short initial phase. This is despite of the fact that more and more of $\dot{\eta}(t-\tau)$ is shifted into the integrated area. The reason for this becomes obvious when considering Figure 6.4. The contribution from a single wave excursion constitutes a negligible amount after being shifted only 20s, due to the low values of the retardation function after the initial fluctuations. Even though the value of the retardation function is visually non-zero where the vertical velocity vector is non-zero, the plotted product of the two vectors are almost invisible.

Since the convolution integral corresponds to the area under the middle graph with the area below the zero reference cancelling out the area above zero, the contribution to the 'memory function' is practically non-existent. Hence the 'constant' oscillation is understood to arise from a continuous wave of vertical velocities being shifted through the initial part of the retardation function, independent of the vertical velocities that are shifted past the initial part. The initial phase of the 'memory function', observed in Figure 6.2 to last only a couple of seconds, arises in the time interval until t reaches the part of the retardation function where the values are so low that the product of the two vectors are neglectable.

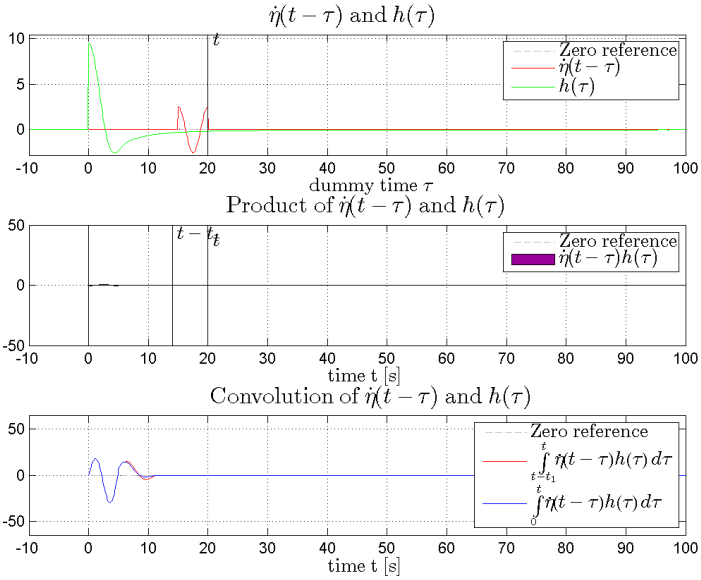

Figure 6.4: The convolution process for a single wave excursion

6.1.2 The effect of higher waves

To investigate how the height of the waves affect the time interval that needs to be included in the limited integral to obtain a certain accuracy, a sample of time spans are tested for different wave amplitudes ζ_a with a constant period T. The maximum relative deviation are written to table 6.2. The wave amplitude of a regular wave is observed not to affect the relative deviation for regular waves. The numerical values of the deviation will however depend on the wave amplitude chosen.

Time span	**Relative deviation, ζ_a =**		
	2	**4**	**6**
1	0.67807	0.67807	0.67807
2	0.30683	0.30683	0.30683
3	0.21867	0.21867	0.21867
4	0.20912	0.20912	0.20912
5	0.14476	0.14476	0.14476
10	0.04226	0.04226	0.04226
20	0.01127	0.01127	0.01127
30	0.00515	0.00515	0.00515
40	0.00283	0.00283	0.00283
50	0.00184	0.00184	0.00184
60	0.00129	0.00129	0.00129
70	0.00091	0.00091	0.00091
80	0.00075	0.00075	0.00075
90	0.00054	0.00054	0.00054
100	0.00000	0.00000	0.00000

Table 6.2: Effect of wave amplitude on relative deviation

6.1.3 The effect of longer wave periods

The effect of increasing the wave period is investigated by analysing a sample of time intervals for different wave periods of regular waves, while keeping the amplitude of the waves constant. The results are given in Figure 6.5. It is easily seen that longer wave periods require greater time intervals included in the 'memory' of the reduced integral to achieve a certain accuracy. A wave with period $T = 5$ seconds will for instance have a maximum deviation of approximately 4% of the correct value for a time interval of 10 seconds. For a wave with period $T = 25$ seconds to achieve the same level of accuracy, almost 20 seconds must be included in the reduced integral. Similarly, it is seen that an included time interval of 10 seconds yields an error of only 4% for a wave with period $T = 5$ seconds, but an error of 12% for a wave with period $T = 25$ seconds.

The period of a regular wave is hence observed to affect the time interval that must be included in the limited convolution integral. Long wave periods require larger

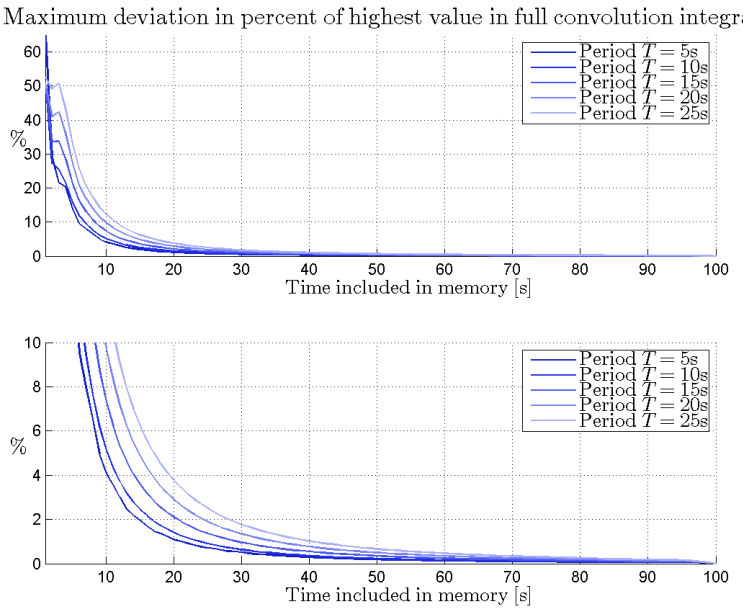

Figure 6.5: Effect of increasing the period for regular waves

time intervals included in the 'memory' than shorter wave periods.

6.2 Irregular waves

Any floating structure at sea will be subjected to irregular sea states which makes analyses in irregular seas important. The results obtained when testing in irregular seas are however dependent of the exact sea state generated from the *irrSea* function. Due to the random phase angles of the wave components, re-running the code results in a new sea state, and hence slightly different deviations. The main characteristics of the analysis do however stay valid unless input parameters are changed.

Table 6.3 presents the maximum deviation obtained for different time spans included in the 'memory'. An irregular sea state with significant wave height $H_S = 5m$ and zero up-crossing period $T_Z = 10s$ is used in this analysis. As was the case for the regular wave analysis, only including a few seconds in the 'memory' results in bad estimations of the full integral. The deviance is seen to be significantly reduced by including a few seconds more in the 'memory'. By including 7 seconds in the 'memory', the largest obtained deviance is 9.2%, i.e. less than 10% of the largest value in the full convolution from 0 to 100 seconds. By comparing the results in table 6.3 to the results found for regular seas given in table 6.1, it is seen that the

Time span included	Maximum deviation	Relative deviation
1	0.42615	0.50774
2	0.21282	0.25357
3	0.19576	0.23324
4	0.16987	0.20239
5	0.12360	0.14726
6	0.09496	0.11314
7	0.07724	0.09203
8	0.06301	0.07507
9	0.05172	0.06163
10	0.04373	0.05210
11	0.03753	0.04471
12	0.03224	0.03841
13	0.02806	0.03343
14	0.02480	0.02955
15	0.02198	0.02619
16	0.01953	0.02327
17	0.01757	0.02093
18	0.01591	0.01896
19	0.01440	0.01716
20	0.01310	0.01560
25	0.00874	0.01041
30	0.00622	0.00741
35	0.00464	0.00553
40	0.00362	0.00431
45	0.00289	0.00345
50	0.00235	0.00280
55	0.00196	0.00234
60	0.00167	0.00199
65	0.00142	0.00169
70	0.00123	0.00146
75	0.00108	0.00129
80	0.00095	0.00114
85	0.00084	0.00100
90	0.00075	0.00090
95	0.00068	0.00081
100	0.00000	0.00000

Table 6.3: Maximum deviation between the two integrals for different time spans included in the 'memory'. End time $t = 100s$ and time increment $dt = 0.01s$. Tested with an irregular sea with significant wave height $H_S = 5m$ and zero up-crossing period $T_Z = 10s$.

results are very similar. This is not unexpected, as the significant wave height and zero up-crossing period used in this analysis are the same as the wave amplitude and period used for the regular wave case.

The deviation and relative deviation for different time intervals included in the 'memory' are shown in Figure 6.6 for the irregular sea analysis. For small time spans, both the maximum deviation and the relative deviation are seen to be less in the irregular sea case than for the regular wave. The relative deviation does however decrease faster for the regular wave, causing the relative deviation of a time span of more than 5 seconds to be smallest in the regular wave case.

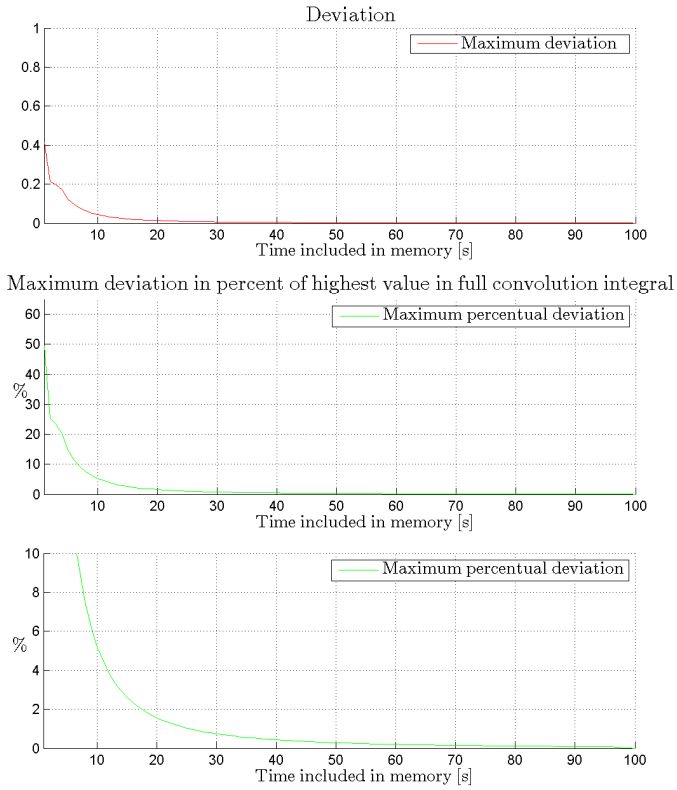

Figure 6.6: Illustration of deviations obtained while calculating the convolution integrals for various time spans in regular waves

Figure 6.7 illustrates the convolution process for a time span of 7 seconds included in the limited integral. This corresponds to an accuracy of 9.2%. The values in the figure are scaled to be of the same order of magnitude for a better visualization. The convolved function is observed not to have an initial and a stable oscillating phase, as was the case for the regular wave analysis conducted in Figure 6.2.

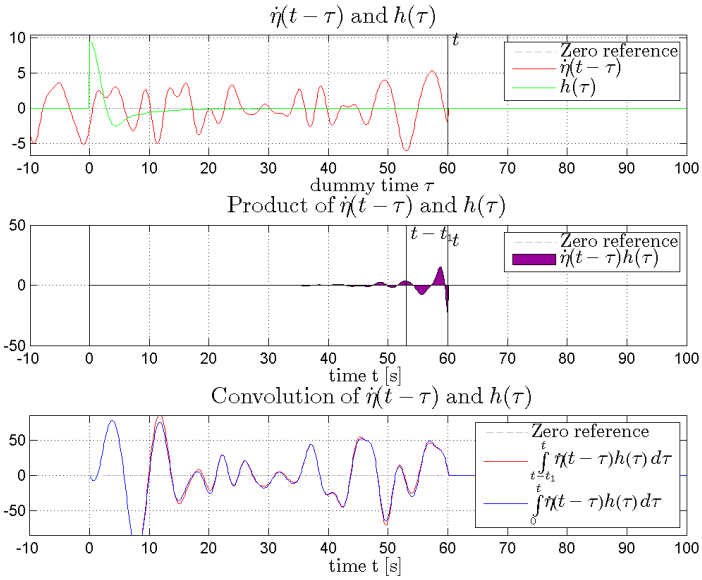

Figure 6.7: The process of convolution shown for an irregular sea state with a time span of 7 seconds included in the limited integral

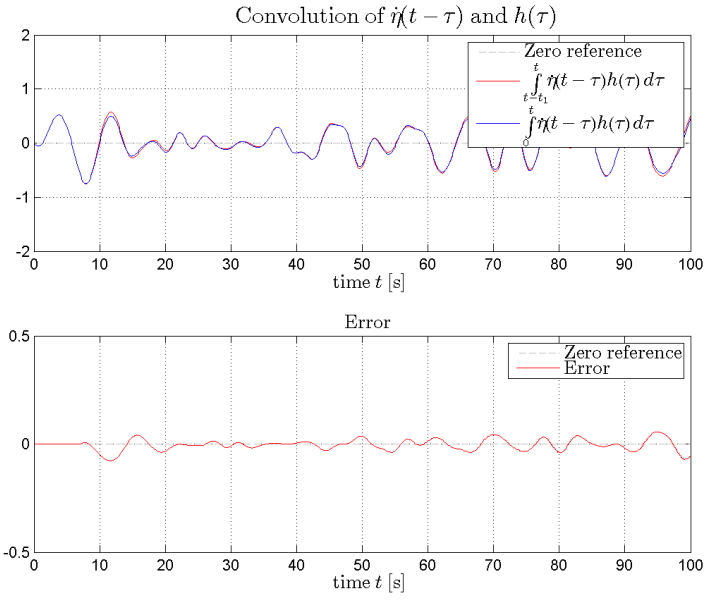

Figure 6.8: The error of the convolved function from Figure 6.7

The irregular sea state cause a vector of irregular vertical velocities, which again cause the convolved function to be irregular. It should be noted that the error, or the deviation, of the limited integral does not oscillate with constant amplitude and period like it did in the regular wave scenario. This is shown in Figure 6.8. From $t = 22$ to $t = 42$ the error is seen to be small, due to a period of low values in the vertical velocity vector causing the area in the part of the product plot not included in the limited integral to be small. Vice versa, from $t = 65$ to $t = 85$, the error is seen to be larger and rapidly changing sign. This is caused by a set of large vertical velocity values causing the limited integral to cut off some of the area contributing to the value in the full integral.

6.2.1 The effect of more severe sea states

Using the set of values for H_{m0} and T_p for different sea states given by Myrhaug (2007), the significant wave heights and zero up-crossing periods listed in table 6.4 might be calculated [19]. The significant wave heights and zero up-crossing periods given in the table is understood to be the most severe sea state with a duration of 3 hours that are likely to occur during the given return period.

Return period	H_S [m]	T_Z [s]
1 year	7.3	10.4
10 years	8.4	10.9
100 years	11.1	12.4

Table 6.4: Significant wave height and zero up-crossing period for sea states of different return periods

The effect of more severe sea states are analysed by comparing the deviations obtained for the three different sea states given in Table 6.4. Figure 6.9 illustrates the relative maximum deviations found. The three cases are seen to give very equal results. This is not unexpected, considering the experience gained from the regular wave analyses. The main difference between the sea state with 1 year return period, and the sea state with 100 years return period is in the significant wave height. The wave height alone does not affect the relative deviation, as concluded in section 6.1.2. The difference in zero up-crossing period is seen from Table 6.4 to be only 2 seconds between the 1 year and 100 year return periods. This difference can hence not be expected to give any large effects to the outcome, and the small differences between the different sea states seem reasonable.

6.3 Evaluation of the results

To decide what is a satisfactory accuracy is not straightforward, as a given accuracy might be considered sufficient in some cases and insufficient in other. For a situation where the memory effects constitutes a small part of the total forces acting on a

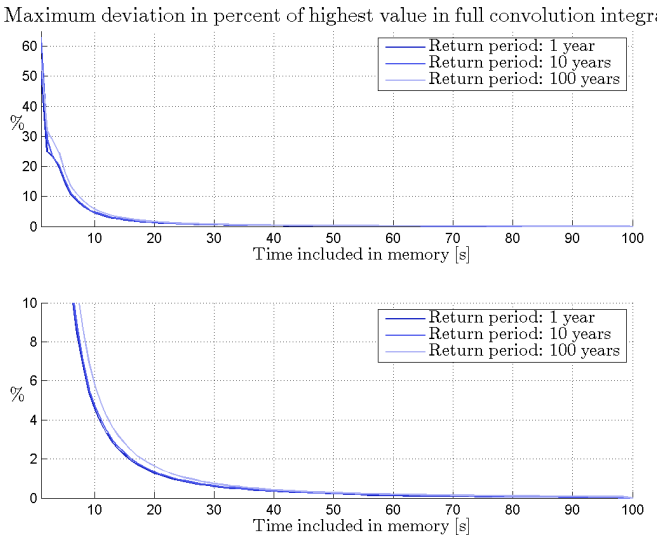

Figure 6.9: Effect of more severe sea states

structure, a rough estimate might be sufficient. However, if the memory effects constitutes a large part of the total forces on the structure, the accuracy might be crucial for the simulated vessel response. In general, a good accuracy is always best, but the additional accuracy comes with a cost in the form of slower calculations and higher computer memory requirements.

Considering the results obtained from the analyses conducted in MATLAB, a time interval of approximately 25 seconds should be sufficient to yield an accuracy of 99%, i.e. less than 1% in maximum deviation. Saving the time history of 25 seconds should not induce too high costs in terms of computational memory requirements. Considering Table 6.3, reducing the error with 0.5%, from 1% to 0.5% costs approximately an additional 12-13 seconds. This is an increase of 50% in storage requirements, only to achieve an additional 0.5% in accuracy. Thus, including 25 seconds in the time history seems to be an appropriate choice.

However, the analyses conducted in this thesis only considers uncoupled heave motion for a rectangular cross-section oscillating in heave at the free surface. As discussed in section 3.1.2, the added mass and frequency coefficients depends on the motion mode and cross-sectional geometry, as well as the submergence. Different values for the $A(\omega)$ term might result in a different retardation function, which again might change the time interval necessary to obtain a sufficient accuracy in the approximated convolution integral. More analyses should thus be conducted in order to state a specific minimum time interval to be included in the 'memory'.

Chapter 7

Implementation in USFOS

On the basis of the experience obtained with the single degree of freedom system dealt with in chapter 5 and 6, a brief description is given of how frequency dependent added mass and damping can be implemented in USFOS.

7.1 Program flow

Figure 7.1 shows a possible program flow of the subroutine that must be written in Fortran code and implemented in USFOS. The first phase consists of collecting data. The cross-sectional geometry is taken as input from the existing USFOS code. The subroutine then checks whether or not information of either $A_{ij}(\omega)$ or $B_{ij}(\omega)$ is known. If not, the program flow is returned to the gathering of cross-sectional data. If the required information is known, the program proceed to the second phase. $A_{ij}(\infty)$ or $B_{ij}(\infty)$ is taken as input from the existing USFOS code, and h_{ij} is calculated. The accuracy is then verified by inverse calculation as described in section 4.3. If a predefined requirement is met, the program flow proceeds to the third phase and if not, the entire process must start over.

In the third phase of the subroutine, the time interval defining the amount of time history stored is taken as input. The convolution integral is the calculated, and the last phase of the subroutine is entered. Here, the structures displacement, velocity and acceleration are taken as input from the existing program, and the equation of motion in the time domain is calculated and returned as output.

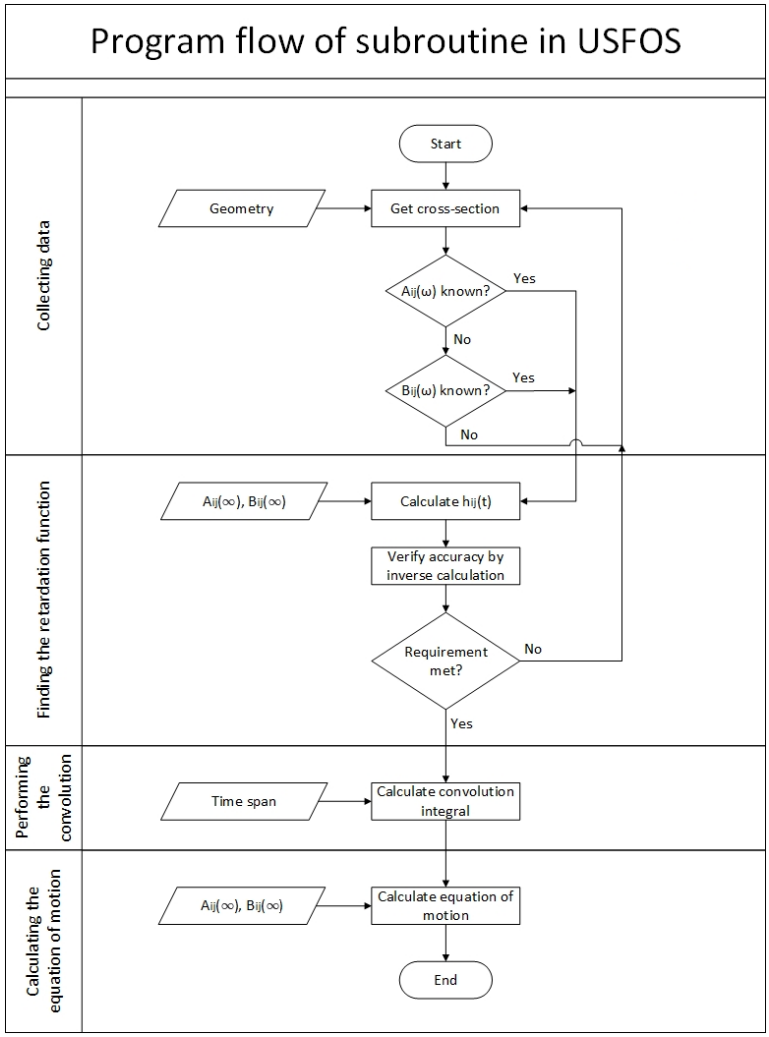

Figure 7.1: Flow chart showing a possible program flow of the subroutine in USFOS

Chapter 8

Concluding Remarks

8.1 Conclusion

Frequency dependence of the diffraction and radiation loads on a floating offshore structure might be implemented in USFOS using memory functions expressed by convolution integrals. One of the terms in each convolution integral is a retardation function, and the procedure of establishing the retardation function requires information of either the added mass coefficients or the damping coefficients to be known for all frequencies. A detailed information over the entire frequency range is necessary for a satisfactory accuracy of the retardation function to be achieved. Digital reading of the data from a scanned plot did not produce an accurate retardation function. This was demonstrated by verifying the accuracy of the retardation function by inverse calculation.

It is possible to obtain a good accuracy of the memory function by including only a short time span in the convolution integral. The analyses conducted in MATLAB for a single-degree of freedom system simulating a platform oscillating in heave, showed that a time span of 25 seconds was sufficient to achieve an error of 1% compared to the full convolution integral. The analyses showed that the relative deviation of the limited integral was independent of the amplitude of the waves the structure was exposed to. The amount of memory needed to obtain a certain accuracy does however depend on the wave period. Waves with long periods require longer time spans included in the memory than waves with shorter periods.

The main findings might be summarized as follows:

- Extreme caution must be taken in establishing the retardation function, as this function is very sensitive to the frequency dependent added mass or damping coefficients.
- The time span that needs to be included in the convolution integral to achieve

a certain accuracy is independent of the amplitude of the waves the structure is subjected to.

- Long wave periods require larger time spans included in the memory function than the smaller wave periods do.
- A time span of 25 seconds was found to be sufficient to achieve a maximum deviation of 1% compared to the maximum value of the full convolution integral for a rectangular cross-section oscillating at the free surface in uncoupled heave motion.

8.2 Recommendations for further work

Many aspects of the calculations performed in this thesis can be further investigated. The importance of the memory terms might be investigated by calculating the total motion response of a platform oscillating in incident waves. The results from an analysis conducted using the time-domain equation of motion might then be compared to the results from an analysis conducted assuming constant added mass and damping terms. This was initially intended to be within the scope of this thesis, but was omitted due to time restrictions.

The amount of time that needs to be included in the memory functions might be further investigated by calculating the retardation functions for other motion modes. Finding the retardation function in heave for other cross-sectional geometries and using that as a basis for the convolution integral could also be interesting. Different retardation functions might require that a different time span is included in the convolution integral to obtain an accuracy of 1%.

Finally, and most interestingly, a subroutine must be written in Fortran to calculate the motion response in the time domain. This subroutine might be tested by calculating the motion response of a platform pontoon, and compare the result to a similar analysis conducted in an extended version of the MATLAB program. The result might then be verified by comparison to results obtained from other programs known to be accurate.

Bibliography

[1] O.M. Faltinsen. *Sea loads on ships and offshore structures.* University of Cambridge, 1990.

[2] M. Huss. Notes on the modeling of irregular seas in time simulations. `http://www.mhuss.se/documents/Downloads/101129mh_Notes_IrrSea_R2.pdf`, 2010. [visited 24.05.2013].

[3] O.M. Faltinsen. *Hydrodynamics of High-Speed Marine Vehicles.* Cambridge University Press, 2005.

[4] M. Greco. Week 37. Lecture notes in Sea Loads, 2010.

[5] B. Pettersen. *Marin teknikk 3 - Hydrodynamikk.* Department of Marine Technology, NTNU, 2007.

[6] J.H. Vugts. *The hydrodynamic coefficients for swaying, heaving and rolling in a free surface.* Laboratorium Voor Scheepsbouwkunde: Technische Hogeschool Delft, 1968.

[7] J. Amdahl. *Marin teknikk 2 - Knekking.* Department of Marine Technology, NTNU, 2009.

[8] Various. Convolution. `http://en.wikipedia.org/wiki/Convolution`, 2012. [visited 22.11.2012].

[9] J.H. Vugts. *The hydrodynamic forces and ship motions in waves.* Delft, 1970.

[10] J.N. Newman. *Marine Hydrodynamics.* Cambridge: The MIT Press, 1977.

[11] R.H. Stewart. *Introduction to Physical Oceanography.* Department of Oceanography, Texas A & M University, 2008.

[12] Unknown. Usfos hydrodynamics: Theory, description of use, verification. `http://www.usfos.no/manuals/usfos/theory/documents/Usfos_Hydrodynamics.pdf`, 2010. [visited 22.05.2013].

[13] K. Hasselmann et al. Measurements of wind-wave growth and swell decay during the joint North Sea wave project (JONSWAP). *Ergänzungsheft zur deutchen hydrographischen Zeitschrift. Reihe A*, 1973.

[14] M.J. Tucker. *Waves in ocean engineering: measurement, analysis, interpretation.* New York: Ellis Horwood, 1991.

[15] W.E. Cummins. The impulse response function and ship motions. Technical report, Department of the Navy, David Taylor Model Basin, 1962.

[16] F.T. Ogilvie. Recent progress toward the understanding and prediction of ship motions. In *Proc. 5th Symp. on Naval Hydrodynamics*, pages 3–128. Office of Naval Research, Department of the Navy, 1964.

[17] O.F. Rognebakke. *Sloshing in Rectangular Tanks and Interaction with Ship Motions.* PhD thesis, Norwegian University of Science and Technology, 2002.

[18] J. Kotic and V. Mangulis. On the Kramers-Kronig relations for ship motions. *International Shipbuilding Progr.*, 1962.

[19] D. Myrhaug. *Marin dynamikk - Uregelmessig sjø.* Department of Marine Technology, NTNU, 2007.

[20] S. Khamis. The convolution integral explained. `http://engineersphere.com/math/the-convolution-integral-explained.html`, 2011. [visited 22.11.2012].

Appendix A

The convolution integral explained

Convolution is a mathematical operation on two functions f_1 and f_2, over the same variable, e.g. $f_1(t)$ and $f_2(t)$. The convolution produces a third function that describes how the first function modifies the second one, and conversely; the resulting function can be seen as how the second function modifies the first function [20]. The convolution is a mathematical operation "giving the area overlap between the two functions as a function of the amount that one of the original functions is translated" [8].

The convolution of two functions is denoted $*$ and is written as in equation (A.1). τ is used as a "dummy variable"; a variable used to shift f_2 through time t [20].

$$f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(\tau) \cdot f_2(t-\tau)\, d\tau \tag{A.1}$$

This process can be explained through the following steps, illustrated by Figure A.1 to A.4:

1. Start out with two functions $f_1(t)$ and $f_2(t)$. Express the two functions in terms of a dummy variable τ: $f_1(\tau)$ and $f_2(\tau)$ as shown in Figure A.1.
2. Reflect one of the functions: $f_2(\tau) \rightarrow f_2(-\tau)$. f_2 is now time-inverted. Leave the other function, $f_1(\tau)$ fixed in τ-space. This is shown in Figure A.2.
3. By adding a time-offset, t, $f_2(t-\tau)$ is allowed to slide along the τ-axis [8]. "Start t at $-\infty$ and slide it all the way to $+\infty$. Wherever the two functions intersect, find the integral of their product" [8]. This process is shown in Figure A.3 to A.4.

The result is a sliding, weight-average of function $f_1(\tau)$, where the weighting function is $f_2(\tau)$.

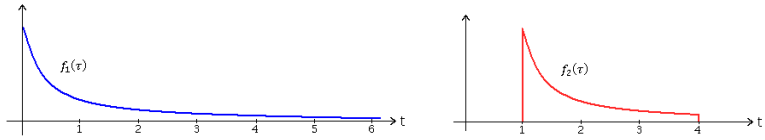

Figure A.1: $f_1$and f_2 as functions of τ [8]

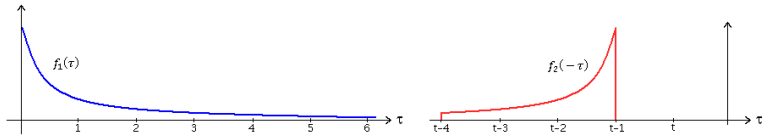

Figure A.2: f_2 time-inverted while f_1 is fixed [8]

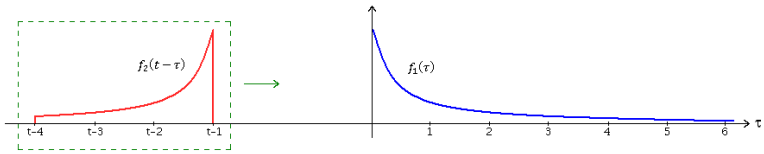

Figure A.3: Time-offset added so f_2 can be shifted through τ [8]

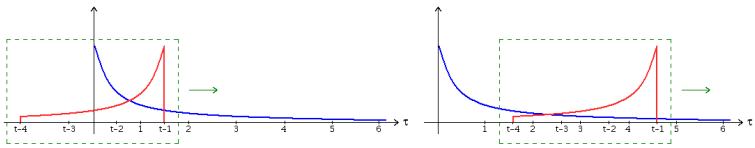

Figure A.4: Shifting of $f_2(t-\tau)$ through $f_2(\tau)$ [8]

Following is a list of properties useful when handling convolution integrals and a set of equations, equations (A.2) to (A.6), showing how the different mathematical operations can be performed on the convolution integrals [20]:

Communicative property: Changing the order of the operands does not change the result.

$$f_1(t) * f_2(t) = f_2(t) * f_1(t) \tag{A.2}$$

Distributive property: The order of the mathematical operations can be over-

ruled by parentheses.

$$f_1(t) * [f_2(t) + f_3(t)] = f_1(t) * f_2(t) + f_1(t) * f_3(t) \tag{A.3}$$

Associative property: The order in which the operations are performed within an expression containing two or more of the same associative operator in a row does not matter as long as the sequence of the operands is not changed.

$$f_1(t) * [f_2(t) * f_3(t)] = [f_1(t) * f_2(t)] * f_3(t) \tag{A.4}$$

Shift property: If equation (A.5) is true, then equation (A.6) is also true.

$$f_1(t) * f_2(t) = c(t) \tag{A.5}$$

$$f_1(t - T_1) * f_2(t - T_2) = c(t - T_1 - T_2) \tag{A.6}$$

Appendix B

MATLAB: Main program

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Marianne Mellbye Larsen:   Master thesis 2013                          %
%                          - Main program -                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear         % Clears all variables from the workspace
clc           % Clears the command window
close all     % Close all plots

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

% Properties for plots:
set(0, 'DefaultAxesFontSize', 11);
set(0, 'DefaultTextInterpreter', 'Latex');

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

disp('------------------------ PROGRAM STARTED ------------------------')

% Get input from user:
maxT = input('Choose end time (e.g. 100): ');        % End time
dt = input('Choose time increment (e.g. 0.1): ');    % Time increment

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

% Let user choose wave type:
type = menu('Select wave type', 'Regular wave', 'Single regular ...
    wave excursion', 'Irregular sea');

if (type == 1)
    disp('Regular wave chosen')
    zeta_a = input('Choose wave amplitude in meters (e.g. 5): ');
    T = input('Choose wave period in seconds (e.g. 5): ');
    % Finding a vector of vertical velocities at different times t ...
        for a wave at position x = 0:
```

```
34      if (T < maxT || T == maxT)      % Wave period must be shorter ...
            than maximum time
35          continualWaveVel = regWave(maxT, dt, zeta_a, T);
36      else
37          disp('Invalid combination of period T and end time maxT ...
                chosen, at least one wave period must be included in maxT')
38      end
39      disp('continualWaveVel:')
40      disp(continualWaveVel)
41
42  elseif (type == 2)
43      disp('Single regular wave excursion chosen')
44      zeta_a = input('Choose wave amplitude in meters (e.g. 5): ');
45      T = input('Choose wave period in seconds (e.g. 5): ');
46      % Finding a vector of vertical velocities at different times t ...
            for a wave at position x = 0:
47      if (T<maxT || T == maxT)
48          singleWaveVel = singleRegWave(maxT, dt, zeta_a, T);
49      else
50          disp('Invalid combination of period T and end time maxT ...
                chosen, at least one wave period must be included in ...
                maxT')
51      end
52      disp('singleWaveVel')
53      disp(singleWaveVel)
54
55  elseif (type == 3)
56      disp('Irregular sea chosen')
57      N = input('Choose number of wave components (e.g. 15): ');
58      H_S = input('Choose significant wave height in meters (e.g. 5): ');
59      T_Z = input('Choose mean zero up-crossing period (e.g. 10): ');
60      % Finding a vector of vertical velocities for an irregular sea ...
            state at position x = 0:
61      if (T_Z < maxT || T_Z == maxT)
62          irrSeaVel = irrSea(maxT, dt, N, H_S, T_Z);
63      else
64          disp('Invalid combination of "period" T_Z and end time maxT ...
                chosen, increase maxT or decrease T_Z')
65      end
66  %     disp('irrSeaVel:')
67  %     disp(irrSeaVel)
68  end
69
70  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
71
72  % Finding the retardation function using a damping related approach:
73  h = retardationD(dt, maxT);
74  disp('h(t)')
75  disp(h')
76
77  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
78
79  % Performing the convolution:
80  if (type == 1)
81      % Calculating the convolution integral for continual wave:
82      contConvTerm = contConvolution(continualWaveVel, h, dt, maxT);
83      disp('contConvTerm:')
```

```
84      disp(contConvTerm)
85  elseif (type == 2)
86      % Calculating the convolution integral for single wave excursion:
87      singleConvTerm = contConvolution(singleWaveVel, h, dt, maxT);
88      disp('singleConvTerm:')
89      disp(singleConvTerm)
90  elseif (type == 3)
91      % Calculating the convolution integral for irregular seas:
92      irrConvTerm = contConvolution(irrSeaVel, h, dt, maxT);
93      disp('irrConvTerm:')
94      disp(irrConvTerm)
95  end
96
97  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
98
99  disp('———————————————————— PROGRAM ENDED ————————————————————————')
```

Appendix C

regWave.m

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %      Marianne Mellbye Larsen:   Master thesis 2013       %
    %       - Function to find the vertical velocity of        %
    %                      a regular wave -                    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function vel = regWave(maxT, dt, zeta_a, T)

    disp('------ Continual regular wave function started ------')

    % Time and x-vector with increment dt:
    x = 0:dt:maxT;       % x vector [m]
    t = 0:dt:maxT;       % Time vector [s]

    % Constants:
    g = 9.81;            % Acceleration of gravity [m/s^2]

    % Wave parameters:
    omega = 2*pi/T;      % Circular frequency [s^-1]
    k = omega^2/g;       % Wave number [m^-1]

    % Stillwater-line:
    stillWater = zeros(length(x),1);

    % Vectors of the wave profile and vertical velocities at x=0:
    prof = zeros(length(t),1);
    vel = zeros(length(t),1);
    for i = 1:length(t)
        prof(i) = zeta_a*sin(omega*t(i));
        vel(i) = omega*zeta_a*cos(omega*t(i));
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%% PLOT %%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Speeds up the animation of the wave:
    increment = 1;
```

```
37      if length(t) > 501
38          increment = floor(length(t)/500);
39      end
40
41      % Plotting the wave animation:
42      for i = 1:increment:length(t)
43          j = t(i);
44          zeta = zeta_a*sin(omega*j - k*x);
45          w = omega*zeta_a*cos(omega*j - k*x);
46
47          figure(1)
48          subplot(2,1,1)
49          plot(x, stillWater, 'Linestyle', '--', 'LineWidth', 2, ...
                'Color', 0.75*[1 1 1]);
50          hold on
51          plot(x, zeta)
52          plot(x, w, 'r')
53          axis([0 maxT -20 20])
54          xlabel('x [m]');
55          str = {'Wave elevation [m]', 'Vertical velocity [m/s]'};
56          ylabel(str);
57          title('Continual regular wave propagation', 'FontSize', 14);
58          hleg1 = legend('Still water level', 'Wave profile', ...
                'Vertical velocity');
59          set(hleg1,'Location', 'NorthEast', 'Interpreter', 'Latex')
60          regWavePlot = gcf;
61          saveas(regWavePlot, 'Plots\regWavePlot', 'png');
62          hold off
63
64          if (i > 200)
65              break;      % Stop the animation (exit the for loop)
66          end
67
68          % Create movie frame
69          M(i) = getframe;
70      end
71
72      disp('------ Continual regular wave function closed ------')
73  end
```

Appendix D

singleRegWave.m

```
1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2   %       Marianne Mellbye Larsen:   Master thesis 2013                    %
3   %         - Function to find the vertical velocity -                     %
4   %                 - of a single wave excursion -                         %
5   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7   function singleVelShort = singleRegWave(maxT, dt, zeta_a, T)
8
9       disp('-------------- Single wave function started ----------------')
10
11      % Time and x-vector with increment dt:
12      x = 0:dt:maxT;      % x vector [m]
13      t = 0:dt:maxT;      % Time vector [s]
14
15      % Constants:
16      g = 9.81;           % Acceleration of gravity [m/s^2]
17
18      % Wave parameters:
19      omega = 2*pi/T;     % Circular frequency [s^-1]
20      k = omega^2/g;      % Wave number [m^-1]
21
22      % Calculating the wavelength:
23      lambda = 2*pi/k;    % [m]
24
25      % Number of x-steps in one wave excursion:
26      stepX = ceil(lambda/dt);
27
28      % Amplitude and vertical velocity for one single wave excursion ...
            at x=0:
29      singleAmp = zeros(length(t),1);
30      singleVel = zeros(length(t),1);
31      for i = 1:length(t)
32          time = t(i);
33          if time <= T
34              singleAmp(i) = zeta_a*sin(omega*time);
35              singleVel(i) = omega*zeta_a*cos(omega*time);
```

```
36          end
37      end
38
39      % Vertical velocity in vector of length one period at x=0:
40      stepT = length(0:dt:T);
41      singleVelShort = zeros(stepT,1);
42      for i = 1:length(singleVelShort)
43          time = t(i);
44          singleVelShort(i) = omega*zeta_a*cos(omega*time);
45      end
46
47      % Still-water level:
48      stillWater = zeros(length(x),1);
49
50      %%%%%%%%%%%%%%%%%%%%%%%%%%%% PLOT %%%%%%%%%%%%%%%%%%%%%%%%%%%%
51
52      % Wave profile and vertical velocity at t=0:
53      wave = zeros(stepX,1);
54      velo = zeros(stepX,1);
55      for i = 1:stepX
56          wave(i) = zeta_a*sin(-k*x(i));
57          velo(i) = omega*zeta_a*cos(-k*x(i));
58      end
59
60      % Plotting the wave:
61      offset = 0;
62      singleZeta = zeros(length(t),1);
63      singleW = zeros(length(t),1);
64      for i = 1:length(t)
65
66          if (offset == 0)
67              singleZeta(1:length(wave),1) = wave;
68              singleZeta(length(wave)+1:length(t),1) = 0;
69
70              singleW(1:length(velo),1) = velo;
71              singleW(length(velo)+1:length(t),1) = 0;
72
73          elseif (offset < length(t)-length(wave))
74              singleZeta(1:offset,1) = 0;
75              singleZeta(offset+1:offset+length(wave),1) = wave;
76              singleZeta(offset+length(wave)+1:length(t),1) = 0;
77
78              singleW(1:offset,1) = 0;
79              singleW(offset+1:offset+length(velo),1) = velo;
80              singleW(offset+length(velo)+1:length(t),1) = 0;
81
82          elseif (offset < length(t))
83              singleZeta(1:offset,1) = 0;
84              singleZeta(offset+1:length(t),1) = ...
                    wave(1:(length(t)-offset),1);
85
86              singleW(1:offset,1) = 0;
87              singleW(offset+1:length(t),1) = ...
                    velo(1:(length(t)-offset),1);
88
89          else
90              singleZeta = 0;
```

```
91
92              singleW = 0;
93          end
94
95          offset = offset + 3; % Speeds up the animation
96
97          if offset > length(t)
98              break;           % Stop the animation (exit the for loop)
99          end
100
101         figure(1)
102         subplot(2,1,1)
103         plot(x, stillWater, 'Linestyle', '--', 'Color', 0.75*[1 1 1]);
104         hold on
105         plot(x, singleZeta)
106         plot(x, singleW, 'r')
107         axis([0 maxT -20 20])
108         xlabel('x [m]')
109         str = {'Wave elevation [m]', 'Vertical velocity [m/s]'};
110         ylabel(str);
111         title('Single excursion wave propagation', 'FontSize',12)
112         hleg1 = legend('Still water level', 'Wave profile', ...
                'Vertical velocity');
113
114         set(hleg1,'Location', 'NorthEast', 'Interpreter', 'Latex')
115         singleRegWavePlot = gcf;
116         saveas(singleRegWavePlot, 'Plots\singleRegWavePlot', 'png');
117         hold off
118
119         if (i > 200)
120             break;      % Stop the animation (exit the for loop)
121         end
122
123         % Create movie frame
124         M(i) = getframe;
125     end
126
127     disp('---------- Single wave function closed ----------')
128 end
```

Appendix E

irrSea.m

```
1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2   %      Marianne Mellbye Larsen:   Master thesis 2013      %
3   %       - Function to find the vertical velocity of       %
4   %                  an irregular sea state -                %
5   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7   function vel_sum = irrSea(maxT, dt, N, H_S, T_Z)
8
9       disp('-------------- Irregular sea function started ----------')
10
11      % Time and x-vector with increment dt:
12      x = 0:dt:maxT;
13      t = 0:dt:maxT;
14
15      % Constants:
16      g = 9.81;       % Acceleration of gravity
17
18      % Finding random phase angles (epsilon):
19      eps = rand(N,1)*2*pi;
20      disp('eps:')
21      disp(eps)
22
23      % Delta omega:
24      domega = (2.18-0.2)/(N-1);
25
26      % Creating a vector of amplitudes:
27      zeta_a = zeros(N,1);
28
29      % Calculating the amplitudes from the wave spectrum:
30      S = @(omega) (H_S^2*T_Z/(8*pi^2)) * (2*pi./(omega.*T_Z)).^5 .* ...
            exp(-(1/pi)*(2*pi./(omega.*T_Z)).^4);
31      for i = 1:N
32          lower = (i-1)*domega;
33          upper = i*domega;
34          int = integral(S,lower,upper);
35          zeta_a(i) = sqrt(2*int);
```

```
36      end
37      disp('zeta_a:')
38      disp(zeta_a)
39
40      % Plotting the wave spectrum:
41      omega_plot = 0:0.01:2;
42      S_plot = zeros(1,length(omega_plot));
43      disp('omega_plot:')
44      disp(omega_plot)
45      disp('S_plot:')
46      disp(S_plot)
47      for i = 1:length(omega_plot)
48          S_plot(i) = (H_S^2*T_Z/(8*pi^2)) * ...
                (2*pi/(omega_plot(i)*T_Z))^5 * ...
                exp(-(1/pi)*(2*pi/(omega_plot(i)*T_Z))^4);
49      end
50      figure(80)
51      hold on
52      plot(omega_plot, S_plot)
53      grid on
54      hold off
55
56      % Finding the frequency values:
57      omega_mid = zeros(N,1);
58      for i = 1:N
59          omega_mid(i,1) = 0.2 + (i-1)*domega;
60      end
61      disp('omega_mid:')
62      disp(omega_mid)
63
64      % Calculating periods:
65      T = zeros(N,1);
66      for i = 1:N
67          T(i,1) = (2*pi)/omega_mid(i,1);
68      end
69      disp('T:')
70      disp(T)
71
72      % Wave number:
73      k = zeros(N,1);
74      for i=1:N
75          k(i,1) = omega_mid(i,1)^2/g;
76      end
77      disp('k:')
78      disp(k)
79
80      % Still-water level:
81      stillWater = zeros(1,length(x));
82
83      % Vector of the wave profile and vertical velocities at x=0;
84      prof = zeros(length(t), N);
85      prof_sum = zeros(length(t), 1);
86      vel = zeros(length(t), N);
87      vel_sum = zeros(length(t), 1);
88      for i = 1:length(t)
89          for n = 1:N
```

```
90              prof(i,n) = zeta_a(n,1)*sin(omega_mid(n,1)*t(i) + ...
                    eps(n,1));
91              vel(i,n) = ...
                    omega_mid(n,1)*zeta_a(n,1)*cos(omega_mid(n,1)*t(i) ...
                    + eps(n,1));
92          end
93          prof_sum(i,1) = sum(prof(i,:));
94          vel_sum(i,1) = sum(vel(i,:));
95      end
96
97
98      %%%%%%%%%%%%%%%%%%%%%%%%%%%% PLOT %%%%%%%%%%%%%%%%%%%%%%%%%%%%
99
100      % Plotting the wave:
101      zeta = zeros(N,length(t));
102      zeta_tot = zeros(1,length(t));
103      w = zeros(N,length(t));
104      w_tot = zeros(1,length(t));
105      for i = 1: length(t)
106          for n = 1:N
107              zeta(n,:) = zeta_a(n,1)*sin(omega_mid(n,1)*t(i) - ...
                    k(n,1)*x + eps(n,1));
108              w(n,:) = ...
                    omega_mid(n,1)*zeta_a(n,1)*cos(omega_mid(n,1)*t(i) ...
                    - k(n,1)*x + eps(n,1));
109          end
110
111          for j = 1:length(t)
112              zeta_tot = sum(zeta);
113              w_tot = sum(w);
114          end
115
116          % Plotting:
117          figure(1)
118          subplot(2,1,1)
119          plot(x, stillWater, 'Linestyle', ':', 'LineWidth', 2, ...
                 'Color', 0.75*[1 1 1])
120          hold on
121          plot(x,zeta_tot)
122          plot(x,w_tot, 'r')
123          axis([0 maxT -4 4])
124          xlabel('x [m]')
125          str = {'Wave elevation [m]', 'Vertical velocity [m/s]'};
126          ylabel(str)
127          title('Irregular seas', 'FontSize', 12)
128          hleg1 = legend('Still water level', 'Wave profile', ...
                 'Vertical velocity');
129          set(hleg1,'Location', 'NorthEast', 'Interpreter', 'Latex')
130          irrSeaPlot = gcf;
131          saveas(irrSeaPlot, 'Plots\irrSeaPlot', 'png');
132          hold off
133
134          if i > 200
135              break;      % Stop the animation (exit the for loop)
136          end
137
138          % Create movie frame:
```

```
139             M(i) = getframe;
140         end
141
142         disp('————————— Irregular sea function ended ——————————')
143     end
```

Appendix F

retardationD.m

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %      Marianne Mellbye Larsen:   Master thesis 2013      %
    %      - Function to find the retardation function -      %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function h_nondim = retardationD(dt, maxT)

    disp('------------ Retardation function started --------------')

    % Platform dimensions (the beam to draft ratio must be two):
    B = 16;                 % Beam [m]
    D = B/2;                % Draft [m]

    % Constants:
    rho_sw = 1025;          % Density seawater [kg/m^3]
    g = 9.81;               % Acceleration of gravity [m/s^2]

    % Damping in heave for 2D platform as omega -> infinity:
    B_inf = 0;              % [kg/m]

    % Vector of damping values at spesific omega values:
    stepsize = 0.001;
    valueRange = 1.6;               % B is defined for omega < 1.6
    nSteps = valueRange/stepsize +1;
    B_coef = zeros(nSteps,1);
    omega_nondim = zeros(nSteps,1);
    for i = 1:nSteps
        omega_nondim(i,1) = (i-1)*stepsize;
        B_coef(i,1) = -0.6591*omega_nondim(i,1)^4 + ...
            3.1149*omega_nondim(i,1)^3 - ...
            4.9105*omega_nondim(i,1)^2 + 2.6013*omega_nondim(i,1) ...
            - 0.0298;
    end

    % Evaluate h33(t):
    t_range = maxT;
```

```
34      n_t_steps = t_range/dt + 1;
35      time = zeros(n_t_steps,1);
36      for i = 1:n_t_steps
37          time(i,1) = (i-1)*dt;
38      end
39
40      zero_ref = zeros(n_t_steps,1);  % Plotting a zero reference vector
41
42      h_eq = @(w)(2/pi) * ((-0.6591*(w*sqrt(B/(2*g)))^4 + ...
            3.1149*(w*sqrt(B/(2*g)))^3 - 4.9105*(w*sqrt(B/(2*g)))^2 + ...
            2.6013*(w*sqrt(B/(2*g))) - ...
            0.0298)*rho_sw*B*D/sqrt(B/(2*g)) - B_inf)*cos(w*(time));
43      h = integral(h_eq,0,1.6,'ArrayValued',true);
44
45      % Non-dimentionals for plotting h33:
46      t_nondim = time*sqrt(g/(B/2));
47      h_nondim = h/(2*rho_sw*g*D);
48
49      %%%%%%%%%%%%%%%%%%%%%%%%%%%% PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%
50
51      % Properties for plots:
52      set(0, 'DefaultAxesFontSize', 10);
53      set(0, 'DefaultTextInterpreter', 'Latex');
54
55      % Plotting non-dimentional damping in heave:
56      figure(2)
57      subplot(2,1,1)
58      hold on
59      plot(omega_nondim, B_coef, 'k')
60      y1_max = max(B_coef)*1.2;
61      axis([0.0 valueRange 0.0 y1_max])
62      xlabel('$\omega \sqrt{\frac{B}{2g}}$')
63      ylabel('$\frac{B_{33}}{\rho_{sw} B D}$', 'Rotation', 0.0)
64      title('Damping coefficient in heave as function of ...
            frequency','FontSize',12)
65      hleg2 = legend('Damping coefficient in heave');
66
67      set(hleg2,'Location', 'NorthEast', 'Interpreter', 'Latex')
68      heaveOmegaPlot = gcf;
69      saveas(heaveOmegaPlot, 'Plots\heaveOmegaPlot', 'png');
70      hold off
71
72      % Plotting dimentionless retardation function h:
73      figure(3)
74      subplot(2,1,1)
75      plot(t_nondim, zero_ref, 'Linestyle', '--', 'Color', 0.75*[1 1 1])
76      hold on
77      plot(t_nondim, h_nondim, 'g')
78      y4_min = min(h_nondim)*1.1;
79      y4_max = max(h_nondim)*1.1;
80      axis([0 t_range y4_min y4_max])
81      xlabel('t $\sqrt{\frac{2g}{B}}$')
82      ylabel('$\frac{h}{2 \rho_{sw}gD}$', 'Rotation', 0.0)
83      title('Dimensionless retardation function $h(t)$ in heave', ...
            'FontSize', 12)
84      hleg3 = legend('Zero reference', 'Retardation function ...
            $h_{33}(t)$');
```

```
85
86      set(hleg3,'Location', 'NorthEast', 'Interpreter', 'Latex')
87      retDPlot = gcf;
88      saveas(retDPlot, 'Plots\retDPlot', 'png');
89      hold off
90
91      disp('————————— Retardation function closed ————————————')
92  end
```

Appendix G

contConvolution.m

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %      Marianne Mellbye Larsen:   Master thesis 2012      %
    % - Function to find the convolution of two functions -   %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function answer = contConvolution(etaDot,h,dt,endt)

    disp('------------ Convolution function started --------------')

    % Time vector t:
    t = 0:dt:endt;

    % Creating vectors for the results of the convolution integrals:
    result = zeros(length(t)+1,1);
    result_limits = zeros(length(t)+1,1);

    % Creating a zero reference for plotting:
    zero_ref = zeros(length(t), 1);

    % Making etaDot a vector of length(t) for plotting:
    etaDotPlot = zeros(length(t),1);
    for i = 1:length(etaDot)
        etaDotPlot(i) = etaDot(i);
    end

    % Plotting the two functions that will be convolved vs time t:
    figure(5)
    plot(t, zero_ref, 'Linestyle', '--', 'Color', 0.75*[1 1 1])
    hold on
    plot(t, etaDotPlot, 'r')
    plot(t, h, 'g')
    ymin = min(min(etaDot), min(h))*1.1;
    ymax = max(max(etaDot), max(h))*1.1;
    axis([0 endt ymin ymax])
    xlabel('time $t$ [s]')
    title('$\dot{\eta}(t)$ and $h(t)$', 'FontSize',12)
```

```
37      hleg5 = legend('Zero reference', 'Retardation function: ...
            $h(t)$', 'Vertical velocity: $\dot{\eta}(t)$');
38      set(hleg5,'Location', 'NorthEast', 'Interpreter', 'Latex')
39      fig5Plot = gcf;
40      saveas(fig5Plot, 'Plots\fig5Plot', 'png');
41      hold off
42
43      % Defining dummy time variable T:
44      T = -endt:dt:endt;
45      zero_t_index = ceil(length(T)/2);  % Finding the index of t=0 ...
            in vector T:
46
47      % Expressing the functions in terms of dummy variable T:
48      etaDot_T_mirror = zeros(length(T),1);
49      h_T = zeros(length(T),1);
50      for i = 1:length(h)
51          h_T(zero_t_index - 1 + i) = h(i);
52      end
53      for i = 1:length(etaDot)
54          etaDot_T_mirror(zero_t_index + 1 - i) = etaDot(i);
55      end
56
57      % Creating a zero reference in T for plotting:
58      zero_ref_T = zeros(length(T), 1);
59
60      etaDot_mirror = zeros(length(etaDot),1);
61      for i = 1:length(etaDot)
62          etaDot_mirror(i) = etaDot(length(etaDot) + 1 - i);
63      end
64
65      % Adding zeros after the vectors because of length issues when ...
            shifting ETA:
66      ETA = [etaDotPlot, zeros(length(etaDotPlot),1)];
67      H = [h, zeros(length(h),1)];
68
69      % Vector of t values used in plotting of the convolution:
70      t_result = zeros(length(result),1);
71      zero_ref_result = zeros(length(result),1);
72      temp = 0:dt:2*endt;
73      for i = 1:length(result)
74          t_result(i) = temp(i);
75      end
76
77      % Defining global paramter for printing of matrix to latex:
78      global maxDiff;
79      maxDiff = zeros(endt,3);
80
81      % Choose time span to be included in the limited integral:
82      timespan = 10;      % [s]
83      % or run a loop over many timespans:
84      % for timespan = 1:endt
85      %     disp(timespan)
86
87          % Time span in steps:
88          timeSpan_steps = timespan/dt;
89
90          product = zeros(length(result));
```

```
91          result = zeros(length(t)+1,1);
92          result_limits = zeros(length(t)+1,1);
93
94          % Lower limit:
95          low_limit = zeros(length(t),1);
96          for i = 1:length(t)
97              if i ≤ timeSpan_steps
98                  low_limit(i,1) = 0;
99              else
100                 low_limit(i,1) = t(i-timeSpan_steps);
101             end
102         end
103
104         % Compare:
105         comp = zeros(length(t),2);
106         comp(:,1) = low_limit(:,1);
107
108         % Figure used for plotting of animation (slow):
109         figure(6)
110
111         offset = -length(etaDot_mirror);  % Position relative to ...
                t=0 of first non-zero element in etaDot_mirror
112         for i = 1:length(result)
113
114             comp(i,2) = t(i);
115
116             etaDot_shift = zeros(length(T), 1);     % Empty ...
                    etaDot_shift
117
118             if offset < (length(t) - length(etaDot_mirror))
119                 for j = 1:length(etaDot_mirror)
120                     etaDot_shift(zero_t_index + offset + j) = ...
                            etaDot_mirror(j);
121                 end
122             else
123                 etaDot_shift(zero_t_index + offset + ...
                        1:length(etaDot_shift)) = ...
                        etaDot_mirror(1:(length(T)-zero_t_index-offset));
124             end
125
126             % Calculating the convolution integral from 0 to t:
127             for j = 1:length(h)
128                 if (i-j+1 > 0)
129                     product(j) = ETA(j) * H(i - j + 1);
130                     result(i+1) = result(i+1) + (ETA(j) * H(i - j + ...
                            1))*dt;
131                 end
132             end
133
134             % Calculating the convolution integral from t-timeSpan ...
                    to t:
135             for j = 1:length(h)
136                 if (i-j+1 > 0)
137                     if (j > low_limit(i)/dt)
138                         result_limits(i+1) = result_limits(i+1) + ...
                                (ETA(j) * H(i - j + 1))*dt;
139                     end
```

```
140                 end
141             end
142
143             if (timespan == 1 || timespan == 10)
144                 if (i == 0.6*endt/dt +1)
145                     % Animated plot (slow):
146                     subplot(3,1,1)
147                     plot(T, zero_ref_T, 'Linestyle', '--', 'Color', ...
                            0.75*[1 1 1])
148                     hold on
149                     plot(T, etaDot_shift, 'r')
150                     plot(T, h_T, 'g')
151                     x7min = -endt/10;
152                     x7max = endt;
153                     axis([x7min x7max ymin ymax])
154                     grid on
155                     if i < length(t)
156                         t_0 = t(i);
157                         opts.vpos = 'top';
158                         opts.halign = 'left';
159                         opts.staircase = true;
160                         vline2(t_0, {'k'}, {'  $t$'}, opts)     % ...
                                Uses a function found on MatWorks File ...
                                Exchange to plot vertical lines (not ...
                                my code)
161                     end
162                     xlabel('dummy time $\tau$')
163                     title('$\dot{\eta}(t-\tau)$ and $h(\tau)$', ...
                            'FontSize',12)
164                     hleg11 = legend('Zero reference', ...
                            '$\dot{\eta}(t-\tau)$', '$h(\tau)$');
165
166                     set(hleg11,'Location', 'NorthEast', ...
                            'Interpreter', 'Latex')
167                     hold off
168
169                     subplot(3, 1, 2)
170                     plot(t_result, zero_ref_result, 'Linestyle', ...
                            '--', 'Color', 0.75*[1 1 1])
171                     hold on
172             %         plot(t_result, product); % Faster than area
173                     area(t_result, product, 'FaceColor', [.6 .0 .6]);
174                     axis([x7min x7max -40 40])
175                     grid on
176                     if i < length(t)
177                         opts.vpos = 'top';
178                         opts.halign = 'left';
179                         opts.staircase = true;
180                         vline2([0 low_limit(i) t(i)], ...
                                {'k','k','k'}, {'','  $t-t_1$','  ...
                                $t$'}, opts)     % Uses a function ...
                                found on MatWorks File Exchange to ...
                                plot vertical lines (not my code)
181                     end
182                     xlabel('time t [s]')
183                     title('Product of $\dot{\eta}(t-\tau)$ and ...
                            $h(\tau)$', 'FontSize',12)
```

```
184                     hleg12 = legend('Zero reference', ...
                            '$\dot{\eta}(t-\tau) h(\tau)$');
185
186                     set(hleg12,'Location', 'NorthEast', ...
                            'Interpreter', 'Latex')
187                     hold off
188
189                     subplot(3, 1, 3)
190                     plot(t_result, zero_ref_result, 'Linestyle', ...
                            '--', 'Color', 0.75*[1 1 1])
191                     hold on
192                     plot(t_result, result_limits, 'r')
193                     plot(t_result, result, 'b')
194                     axis([x7min x7max -40 40])
195                     grid on
196                     xlabel('time t [s]')
197                     title('Convolution of $\dot{\eta}(t-\tau)$ and ...
                            $h(\tau)$', 'FontSize',12)
198                     hleg13 = legend('Zero reference', ...
                            '$\int\limits_{t-t_1}^t \dot{\eta}(t-\tau) ...
                            h(\tau) \, d\tau$ \quad', ...
                            '$\int\limits_{0}^t \dot{\eta}(t-\tau) ...
                            h(t) \, d\tau$');
199
200                     set(hleg13,'Location', 'NorthEast', ...
                            'Interpreter', 'Latex')
201                     if (timespan == 1)
202                         convAnPlot_1 = gcf;
203                         saveas(convAnPlot_1, 'Plots\convAnPlot_1', ...
                                'png');
204                     elseif (timespan == 10)
205                         convAnPlot_10 = gcf;
206                         saveas(convAnPlot_10, ...
                                'Plots\convAnPlot_10', 'png');
207                     end
208                     hold off
209
210                     % Create movie frame
211                     M(i) = getframe;
212                 end
213             end
214
215             offset = offset + 1;
216
217             if i >=length(t)
218                 break
219             end
220         end
221
222         disp('comp:')
223         disp(comp)
224
225         res = zeros(length(result),5);
226         res(:,1) = result(:,1);
227         res(:,2) = result_limits(:,1);
228         for i = 1:length(res)
229             res(i,3) = (res(i,1)-res(i,2));
```

```
230             res(i,4) = (res(i,3)/res(i,1));
231             res(i,5) = res(i,4)*100;
232         end
233         disp('res:')
234         disp(res)
235 
236         maxIntegralValue = max(abs(res(:,1)));
237 
238         maxDiff(timespan,1) = timespan;
239         maxDiff(timespan,2) = max(abs(res(:,3)));
240         maxDiff(timespan,3) = maxDiff(timespan,2)/maxIntegralValue;
241 
242 
243         maxError = max(res(:,4));
244         disp('maxError:')
245         disp(maxError)
246 
247         maxPercent = max(res(:,5));
248         disp('maxPercent:')
249         disp(maxPercent)
250 
251     % end
252     answer = maxDiff(:, 3);
253     disp('answer:')
254     disp(answer)
255 
256     % Figure:
257     figure(7)
258     subplot(2,1,1)
259     plot(t_result, zero_ref_result, 'Linestyle', '--', 'Color', ...
            0.75*[1 1 1])
260     hold on
261     plot(t_result, result_limits, 'r')
262     plot(t_result, result, 'b')
263     axis([0 endt -40 40 ])
264     grid on
265     xlabel('time $t$ [s]')
266     title('Convolution of $\dot{\eta}(t-\tau)$ and $h(\tau)$', ...
            'FontSize',12)
267     hleg71 = legend('Zero reference', '$\int\limits_{t-t_1}^t ...
            \dot{\eta}(t-\tau) h(t) \, d\tau$', '$\int\limits_{0}^t ...
            \dot{\eta}(t-\tau) h(t) \, d\tau$');
268     set(hleg71,'Location', 'NorthEast', 'Interpreter', 'Latex')
269     hold off
270 
271     subplot(2,1,2)
272     plot(t_result, zero_ref_result, 'Linestyle', '--', 'Color', ...
            0.75*[1 1 1])
273     hold on
274     plot(t_result, res(:,3), 'r')
275     axis([0 endt -10 10 ])
276     grid on
277     xlabel('time $t$ [s]')
278     title('Error')
279     hleg72 = legend('Zero reference', 'Error');
280     set(hleg72,'Location', 'NorthEast', 'Interpreter', 'Latex')
281     fig7Plot = gcf;
```

```
282     saveas(fig7Plot, 'Plots\fig7Plot', 'png');
283     hold off
284
285     % Used only when the loop iterating over many time spans is ...
            activated:
286     figure(8)
287     subplot(2,1,1)
288     hold on
289     plot(maxDiff(:,1), maxDiff(:,2), 'r')
290     axis([1 length(maxDiff) 0 7 ])
291     grid on
292     xlabel('Time included in memory [s]')
293     title('Deviation', 'FontSize',12)
294     hleg81 = legend('Maximum deviation');
295     set(hleg81,'Location', 'NorthEast', 'Interpreter', 'Latex')
296
297     subplot(2,1,2)
298     hold on
299     plot(maxDiff(:,1), maxDiff(:,3)*100, 'g')
300     axis([1 length(maxDiff) 0 65 ])
301     grid on
302     xlabel('Time included in memory [s]')
303     title('Maximum deviation in percent of highest value in full ...
            convolution integral', 'FontSize',12)
304     hleg82 = legend('Maximum percentual deviation');
305     set(hleg82,'Location', 'NorthEast', 'Interpreter', 'Latex')
306     fig8Plot = gcf;
307     saveas(fig8Plot, 'Plots\fig8Plot', 'png');
308     hold off
309
310
311     figure(9)
312     subplot(2,1,2)
313     hold on
314     plot(maxDiff(:,1), maxDiff(:,3)*100, 'g')
315     axis([1 length(maxDiff) 0 10 ])
316     grid on
317     xlabel('Time included in memory [s]')
318     ylabel('\%', 'Rotation', 0.0)
319     title('Maximum deviation in percent of highest value in full ...
            convolution integral', 'FontSize',12)
320     hleg9 = legend('Maximum percentual deviation \quad');
321     set(hleg9,'Location', 'NorthEast', 'Interpreter', 'Latex')
322     fig9Plot = gcf;
323     saveas(fig9Plot, 'Plots\fig9Plot', 'png');
324     hold off
325
326 disp('————————— Convolution function ended ——————————————')
327 end
```