Master's thesis

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

Jon Arnt Kårstad

Real-Time Simulation of Ship Waves

Master's thesis in Marine Technology
Supervisor: Associate Professor David Kristiansen
June 2022

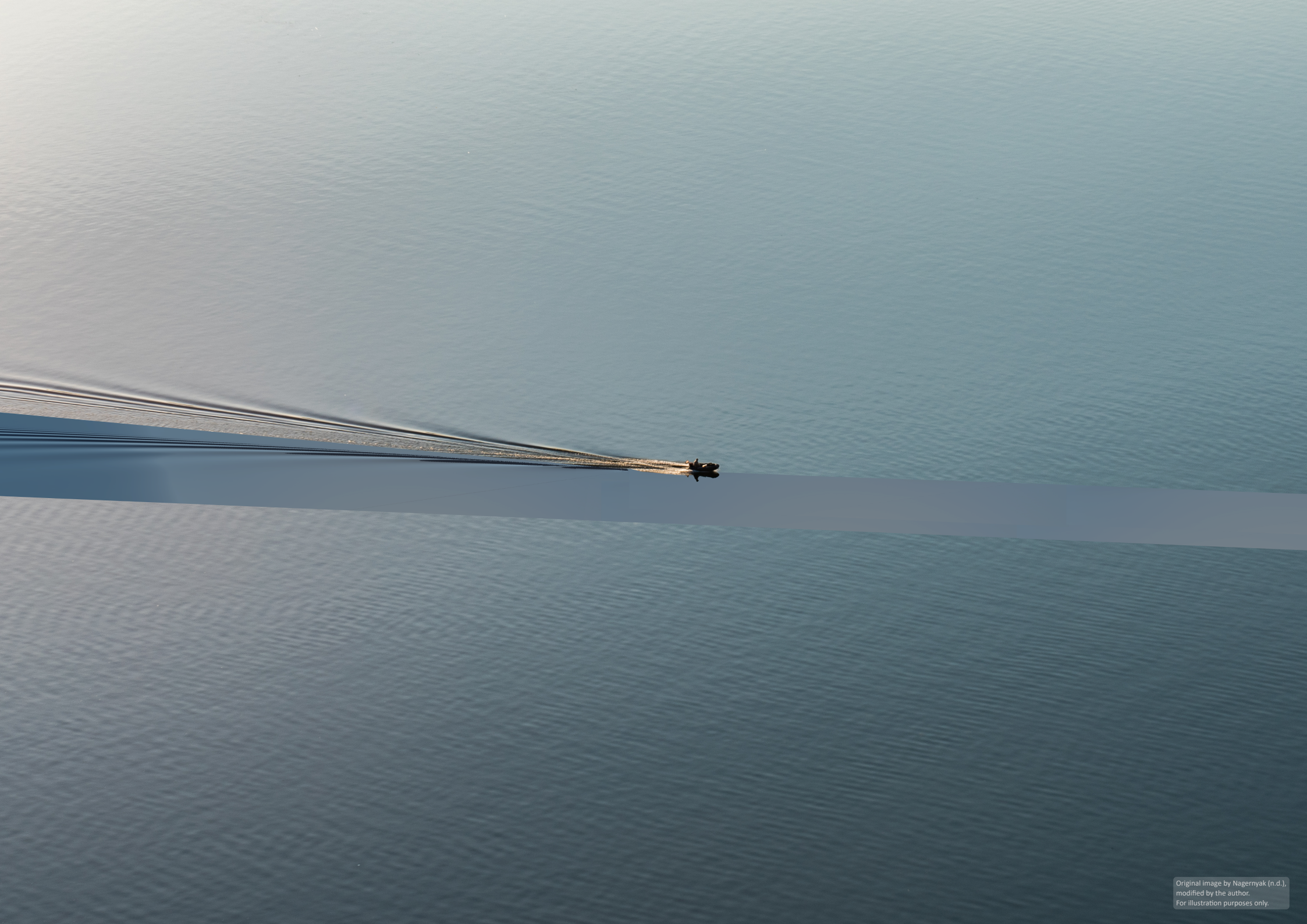
NTNU
Norwegian University of
Science and Technology

Jon Arnt Kårstad

Real-Time Simulation of Ship Waves

Master's thesis in Marine Technology
Supervisor: Associate Professor David Kristiansen
June 2022

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



Original image by Nagernyak (n.d.),
modified by the author.
For illustration purposes only.

Preface

The following presents the work performed for a Master's thesis in Marine Technology at the Norwegian University of Science and Technology (NTNU), Trondheim, specialising in the discipline of Marine Hydrodynamics. The work in its entirety corresponds to 30 ECTS.

Ever since I first undertook the task *Real-Time Simulation of Ship Waves*, I, as so many before me, became a victim of the beautiful nature that is the wave pattern following a vessel moving in calm water. It fascinates me how more than a century old contributions to the topic are to this day relevant, and it motivates me to take part in a subject for which many of the greatest minds of hydrodynamics have previously lended their thoughts. I am forever grateful for the journey that has been my thesis.

I would like to express my very great appreciation to my supervisor; Associate Professor David Kristiansen, for his continuous support and valuable discussions. I wish to acknowledge the help provided by Zeabuz, in particular by PhD Candidate Kjetil Vasstein, and the assistance of PhD Candidate Benjamin Lagemann in facilitating my use of the Ocean Systems Lab. Lastly, I would like to thank my family and my colleagues in the office. Without you, this would not have been possible.

Jon Arnt Kårstad

09.06.2022, Trondheim

Date & location

Abstract

The thesis aims to develop a routine for realistic and efficient visual modelling of ship waves in real-time. The purpose is to improve the purely virtual environments where algorithms to be used in autonomous vessels can more safely, more cost-efficiently and more rapidly be tested and trained on interpreting and reacting to real-life scenarios.

The accurate implementation of ship-generated wave systems is important to ensure the training environment of the algorithm is as realistic as possible. If not, the behaviour of autonomous vessel may become unreliable when applied in a real-world environment.

Previously, detailed results with high precision have been computed using for instance Navier-Stokes solvers. However, due to the computational effort required, such methods are not suitable for use in real-time simulations.

To address this, an approach based on the thin-ship theory of J. H. Michell (1898) has been implemented, enabling estimation of ship wakes in deep and finite water depth using linear potential flow theory. The procedure considers the true hull shape of the vessel but is limited to the far-field wave system. It was found that by utilising the disturbance pattern of Kelvin's method of stationary phase for a single point source moving on the surface, the steady state solution may be adapted to also account for turning and acceleration of the vessel. In addition, wall reflections and wake interactions are implemented.

The final procedure is capable of rapidly computing the wave pattern for several vessels at a time in a large variety of environments, showing good qualitative agreement with the theory. Calculated wave resistance coefficients are verified according to relevant studies on linearised theory of ship waves.

Future research should optimise the procedure to properly investigate its ability to run in real-time, as well as validate the wave pattern for turning and accelerating vessels using more quantitative means of comparison. The presence of spray and foam in the wake region, the near-field wave pattern and the effect of current should also be examined.

Keywords: ship waves, wake, wave pattern, real-time simulation, Michell's thin ship theory

Samandrag

Denne oppgåva har som mål å utvikle ei rutine for realistisk og effektiv visuell modellering av skipsbølgjer i sanntid. Formålet er å forbetre dei virtuelle miljøa der algoritmar som skal brukast i autonome fartøy enklare, raskare og tryggare kan trenast og testast på å kjenna att og reagera på verkelegsnære hendingar.

Nøyaktig implementering av skipsgenererte bølgjesystem er viktig for å forsikra at miljøet algoritmane trenar i er så realistisk som mogleg. Dersom det ikkje er tilstrekkeleg kan oppførsla til dei autonome fartøya bli upåliteleg når dei blir tatt i bruk i verkelegheita.

Tidligare har detaljerte resultat med høg presisjon blitt berekna ved bruk av blant anna Navier-Stokes løysarar. Grunna den store mengda beregningsinnsats som krevst ved bruk av slike metodar blir dei derimot rekna som uegna til bruk i sanntid.

For å ta tak i dette har ein framgangsmåte basert på Michells tynne skip teori blitt implementert, der ein bereknar skipsbølgjene i uendeleg og endeleg vanndjup ved hjelp av lineær potensialteori. Prosedyren tar utgangspunkt i den sanne skrogforma til fartøyet, men er avgrensa til bølgjemønsteret i fjernfeltet. Det blir vist at ein ved å ta utgangpunkt i forstyrringsmønsteret til Kelvins stasjonær fase modell for eit punkt som beveg seg på overflata, kan tilpasse den stabile løysinga til å ta omsyn til svinging og akselerasjon. I tillegg blir veggrefleksjonar og interaksjon mellom ulike bølgjemønster studert.

Den endelege prosedyren er i stand til å hurtig berekna bølgjemønsteret for fleire fartøy samtidig i ei rekkje ulike miljø, og viser godt kvalitativt samsvar med teorien. Berekna bølgjemotstandskoeffisientar blir verifisert ved bruk av relevante studiar som omhandlar linearisert teori for skipsbølgjer.

Vidare arbeid bør optimalisere prosedyren for å kunne bekrefta om den lar seg køyre i sanntid, samt validere bølgjemønsteret for svingande og akselererande fartøy ved å ta i bruk kvantitative samanlikningar. Nærværet av bølgjeskum i kjølvatnet, bølgjemønster i nærfeltet, og effekt av straumningar i vatnet bør òg implementerast.

Nøkkelord: skipsbølgjer, kjølvatn, bølgjemønster, simulering i sanntid, Michells tynne skip teori

Table of Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

ASV	Autonomous surface vessel
CPU	Central processing unit
FPS	Frames per second
GPU	Graphics processing unit
HLSL	High-level shader language
NaN	Not a number
ODE	Ordinary differential equation

Greek symbols

α	Angular polar coordinate of (x, y, z)
α_c	Kelvin angle
β	Wave propagation angle
λ	Wave length
μ	Dynamic viscosity, $\mu = \nu\rho$
ν	Kinematic viscosity, $\nu = \frac{\mu}{\rho}$
ω	Angular frequency
Φ	Velocity potential
ϕ	Velocity potential caused by the ship
ρ	Density of fluid
θ	Angle used in classification of ship waves, ref. Fig. 3.1
ζ	Wave elevation
ζ_a	Amplitude of wave elevation

List of symbols

Fn_l	Length Froude number
Fn_h	Depth Froude number

$\Re$	Real part of a complex number
cp	Centerplane of the hull
B	Breadth
C_w	Wave resistance coefficient
D	Draught
g	Gravitational acceleration
h	Water depth
k	Wave number
k_0	g/U^2
L	Length
Lwl	Length of waterline
r	Radial polar coordinate of (x, y, z)
R_w	Wave resistance
S	Wetted surface
U	Velocity of vessel
V_g	Group velocity
V_p	Phase velocity
X, Y, Z	Earth-fixed coordinate system
x, y, z	Ship-fixed coordinate system

Chapter 1

Introduction

1.1 Background

The training and testing of algorithms to be used in autonomous surface vessels (ASV) require a vast amount of data. Considering the various senors needed and the uncontrollable nature of the environment that is the water surface, it is natural to desire a method for generating customised data-sets. This is the motivation behind Gemini (Vasstein and Skarshaug 2021), a visual simulator whose purpose is to provide a foundation for development and testing of autonomous applications.

A key criteria for simulators aiming to recreate real-life scenarios and physics is to obtain sufficient accuracy. Otherwise, any training or testing conducted in the simulator becomes at best useless, and at worst unsafe if applied in real-life. However, most often an increase in accuracy comes along with an increase in computational effort required. As a consequence, the most accurate solutions are not necessarily the best suited.

Ship wakes have from experience shown to disrupt the electromagnetic radiation sensors of the ASV, resulting in false error detection (Zeabuz 2021). Which aspects of the ship wake that causes the errors are not yet verified in detail, but the far-field wave pattern is assumed to be one. Other possibilities include the near-field wave pattern and the occurrence of spray and foam in the wake region. It is for this reason desired to implement hydrodynamically accurate modelling of the ship wake for Gemini, which is the purpose of this master thesis.

1.2 Scope

The scope of the thesis is to investigate and propose a method for rapid computation and visualisation of ship wakes using the linear thin-ship theory of Michell. A requirement is that the procedure is capable of computing the ship wake of several vessels in real-time, meaning that more complex methods for estimating the wake, such as CFD, are considered too inefficient. The wave pattern should be dynamic, i.e., change depending on the path taken by the ship, the velocity, the water depth, the hull geometry etc.

The procedure should be implemented in Unity using the High-Level Shader Language (HLSL), i.e., it must be parallelizable. The thesis does however not require full optimization of the code, nor need the appearance (color, transparency, reflection etc.) of the water surface be visualised as authentic. Indications on runtime performance is nevertheless desired as a proof of concept.

1.3 Thesis outline

First, in Chapter 2, a brief review will be given on the historic development of ship waves as a research topic, addressing in particular the methods relevant to the thesis. Chapter 3 will proceed by describing the theory of ship waves in deep and finite water, the linear thin-ship theory of Michell and the numerical procedures needed to evaluate the derived expressions. In Chapter 4, the procedure is implemented numerically for the case of a ship on a straight course and a circular course, an accelerating vessel and for wall reflections. A brief introduction to the game engine Unity is also provided, touching on some topics relevant to the thesis. The results are then presented and discussed in Chapter 5, followed by a listing of further work in Chapter 6 and the conclusion in Chapter 7. Lastly, additional visualisations of ship waves for various courses, the code structure and the code implementation are attached in the appendix.

Chapter 2

Literature review

> I must now call your attention to the most beautiful, the most difficult, and in some respects the most interesting part of my subject, that is, the pattern of waves formed in the rear of a ship at sea, not confined by the two banks of a canal.
>
> *Lord Kelvin (Sir William Thomson), 1887*

Ship waves, or wake, is a phenomena equally meaningful for a duckling swimming in a pond as for a destroyer sailing on the ocean. It has been a relevant hydrodynamic research area for almost two centuries, and still is today. This chapter will guide you through parts of the historic evolution of ship waves as a research topic, although admittedly, the list of eminent researchers contributing to the field is far greater than what will be presented here.

The very first remarks on the topic are by Lord Kelvin (1887a) credited W. Froude during his many experiments on ship resistance. Froude (1875) describes the hills and valleys formed by a body near the surface as, in a sense, waves; representing energy wasted from the system attending the ship. Commonly known as wave resistance, the actions involving its cause were at the time described by Froude as *"so complicated that no extensive theoretical treatment of the subject can be usefully attempted."* However, and as will be seen, it is in the light of wave resistance much of the early work on ship waves will have its roots.

It is during the time of Froude's experiments the streamline theory based on Stokes principles is evolving within the theory of ship resistance. Contrary to earlier belief, it describes how the resistance of a ship, including the wave-resistance, is not solely dependent on the maximum cross-section of the body, but on the entire hull shape. Regarding the streamline theory in ship design, Froude (1876) says: *"It is, I repeat, a simple fact that the whole framework of thought by which the search for improved forms is commonly directed, consists of ideas which, if the doctrine of stream-lines is true, are absolutely delusive and misleading."*

Following the work of Lord Rayleigh (1876), Lord Kelvin (1887a) became the first to give an explanation of the v-shaped pattern characteristic to the ship wake. He showed that for a ship following a straight path in infinite water depth, the part of the wave pattern that is not exceedingly small is confined between two straight lines drawn from the bow of the ship, making an angle of $19°28'$ between the outer boundary and the ship's course, known as the Kelvin angle. In addition, and as accurately illustrated by hand-drawings in Lord Kelvin (1887a), he described how the structure of the ship wake consists of transverse and divergent waves. The method of stationary phase for approximating highly oscillatory integrals, first developed by Lord Kelvin (1887b) and applied to the problem of ship waves, is to this day a viable option for obtaining the asymptotic solution of general Fourier integrals.

In 1898, J. H. Michell published *The Wave-Resistance of a Ship*—a purely analytical approach for predicting the wave resistance of a ship with constant velocity in infinite water depth. By Tuck (1989) described as Michell's perhaps finest achievement, the triple integral that forms the expression for wave resistance provided an output quantity of design interest from the specified unrestricted hull geometry. However, and for reasons that can only be speculated in, the work of Michell should remain apparently unnoticed to the targeted audience for almost thirty years, before amongst others W. C. S. Wigley and Sir T. H. Havelock began to recognize its fundamental importance (A. G. M. Michell 1941). The thin-ship theory of Michell has since been extended and applied to a broad range of hydrodynamic problems, and was as late as 1979 concluded by Bai and McCarthy (1979) as being *"... rather consistent in comparison with experimental data and not worse than the envelope of predictions of seemingly more sophisticated methods ..."* for the numerical computation of wave resistance for several vessel shapes. In hindsight of Michell's discovery, one may easily wonder why he never followed up with more papers on a subject he had put an enormous amount of effort into not only evolving, but also checking, rechecking and providing a specific example including a numerical evaluation of the integral (A. G. M. Michell 1941). Although a definitive answer can not be asserted, Tuck (1989) indicates the lack of recognition for his work as a probable reason, supported by the words of Michell's brother in the obituary; *"Michell never showed concern for recognition of his work, but the absence of any response to such a paper as the 'Ship Waves' inevitably discouraged the making, or at least the publication, of investigations involving, for him, so much anxious labour."*(A. G. M. Michell 1941).

Since then, numerous papers have been published on the topic of ship waves and ship resistance, many of which in the collected works of Havelock (1965), but it may also be referred to the work of Lamb (1932), Stoker (1957), Wehausen and Laitone (1960), Newman (1977), Noblesse (1983, 2000) and many others. For this thesis in particular, the Neumann-Michell theory of ship waves given by Noblesse et al. (2013) will be used for comparison.

With the rapid development of computers in the later years, the degree of accuracy and detail in the animation of ship waves (and fluids in general) have seen a major improvement. An example of this is seen in the recent photo-realistic simulations of fluids and ship wakes by L. Huang et al. (2021), applying a hybrid approach combining the BEM (Boundary Element Method) and FLIP (Fluid Implicit Particle) method for the incompressible Euler equations. However, the use of CFD (Computational Fluid Dynamics) currently is, and most likely will remain too computationally expensive to be used in a real-time simulation.

Khan (1994) presents in his thesis a simpler and more efficient approach suitable for animating the wake created by a ship moving along an arbitrary path. He applies the mathematical treatment of Kelvin's method of stationary phase for a single point source moving on the surface, approximating the wake by superimposing circular waves emanating from points along the ship's track. The resulting pattern manages to capture the characteristics of the transverse and divergent wave systems as well as the Kelvin angle for both a turning and accelerating vessel, but with limited accuracy. A similar approach is described by Glassner (2002), but for a ship with constant speed on a straight path.

Yuksel (2010) and Yuksel et al. (2007) presents the concept of wave particles for the real-time simulation of fluid waves and their interactions with floating objects. The method is simple, fast and unconditionally stable, providing results of plausible realism for fluid and floating object interactions in real time. However, when considering the results obtained for ship waves in particular, several key characteristics are not captured, thereamong the Kelvin angle (Swierkowski et al. 2009).

Considering the drawbacks of the methods previously presented, it has for the present project been decided to develop a new method based on the thin ship theory of Michell. Therefore, as the primary motivation, the SWPE (Sea Wave Pattern Evaluation) project of Tuck, Scullen et al. (1999a,b, 2000a,b, 2001a, 2002) will be used, considering its similar use of Michell's theory. The six reports (not including appendages and user documentation) describes several features which are highly relevant to the present project, such as the far-field and near-field wave patterns, the effect of finite water depth, the effect of viscosity etc.

Chapter 3

Theory

3.1 Ship waves

For a ship in calm water moving along a straight course with constant speed and constant water depth, the wave pattern as observed from a ship-fixed coordinate system will be steady, i.e., independent of time. If observed from an Earth-fixed coordinate system however, the wave pattern is unsteady. It is therefore necessary to define two coordinate systems as displayed in Fig. 3.1; one ship-fixed (x, y, z) and one Earth-fixed (X, Y, Z). For the ship-fixed coordinate system, the x- and y-axes are defined in the mean free surface with the z-axis pointing upwards according to the right-hand rule. The Earth-fixed coordinate system is related to the ship-fixed as follows:

$$x = X + Ut,\ y = Y,\ z = Z, \tag{3.1}$$

where U is the forward speed of the ship and t denotes the time.

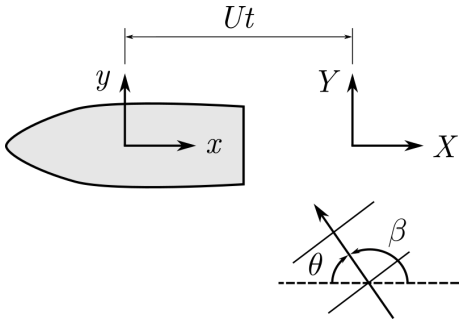


FIGURE 3.1: Ship-fixed (x, y, z) and Earth-fixed (X, Y, Z) coordinate systems for a ship on a straight course. β is the wave propagation angle while θ is the angle used in classification of ship waves, $-0.5\pi \leq \theta \leq 0.5\pi$. The θ displayed in the figure is negative.

The wave profile for a regular wave with amplitude ζ_a, wave number k and angular frequency ω propagating with an angle β relative to the X-axis can using linear wave theory be expressed as

$$\zeta = \zeta_a \cos\left(kX\cos\beta + kY\sin\beta - \omega t - \epsilon\right), \tag{3.2}$$

where ϵ is an arbitrary constant phase angle. Introducing the angle $\theta = \pi - \beta$ and transforming Eq. (3.2) to the ship-fixed coordinate system gives

$$\zeta = \zeta_a \cos(kx\cos\theta + ky\sin\theta + (-kU\cos\theta + \omega)t + \epsilon). \tag{3.3}$$

For the wave pattern to be independent of time in the ship-fixed coordinate system, Eq. (3.3) requires $\omega = kU\cos\theta$.

3.1.1 Ship waves in deep water

If assuming deep water (infinite water depth), the dispersion relation

$$\omega^2 = gk \tag{3.4}$$

is valid and the phase velocity V_p and group velocity V_g are given by

$$V_p = \frac{\omega}{k} = \frac{g}{\omega}, \tag{3.5}$$

$$V_g = \frac{d\omega}{dk} = \frac{1}{2}V_p. \tag{3.6}$$

The condition for a steady wave system then becomes

$$\omega = \frac{g}{U\cos\theta}, \tag{3.7}$$

where the trivial solution $\omega = 0$ has been neglected. Since $\omega > 0$, $g > 0$ and $U > 0$, Eq. (3.7) is limited by $\cos\theta > 0$, that is, $-\pi/2 \leq \theta \leq \pi/2$. The group velocity of the wave system can now be expressed as:

$$V_g = \frac{1}{2}U\cos\theta, \tag{3.8}$$

and the wave number k and wave length λ as:

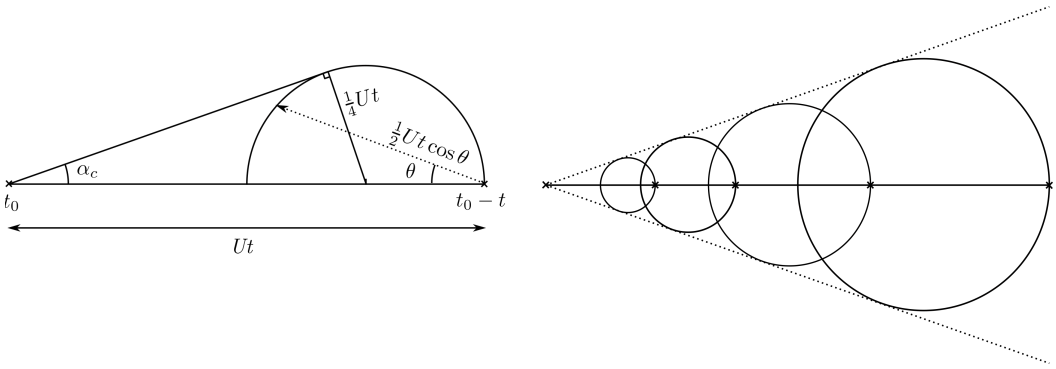
$$k = \frac{g}{U^2\cos^2\theta}, \tag{3.9}$$

$$\lambda = \frac{2\pi}{g}U^2\cos^2\theta. \tag{3.10}$$

The group velocity in Eq. (3.8) describes the propagation of the front of a harmonically oscillating wavetrain, and may be used to give a simplified evaluation of the Kelvin angle. Consider the ship as a pressure point moving with constant velocity U along a straight path in infinite water depth. At time t_0, the wave energy created at time $t_0 - t$ will be described by Eq. (3.8) as shown in Fig. 3.2a. By idealizing the ship as a point, a line may be drawn from it's location at time t_0 which is tangent to the curve of wave energy. The resulting angle α_c—the Kelvin angle—is using trigonometry expressed as

$$\alpha_c = \arcsin\frac{1}{3} = 19°28'. \tag{3.11}$$

The envelope which defines the region of disturbance for a ship on a straight course is displayed in Fig. 3.2b. A more detailed derivation of the Kelvin angle will later be made using the method of stationary phase, but it may also be found in the works of e.g. Lord Kelvin (1887b), Lamb (1932) and Stoker (1957).

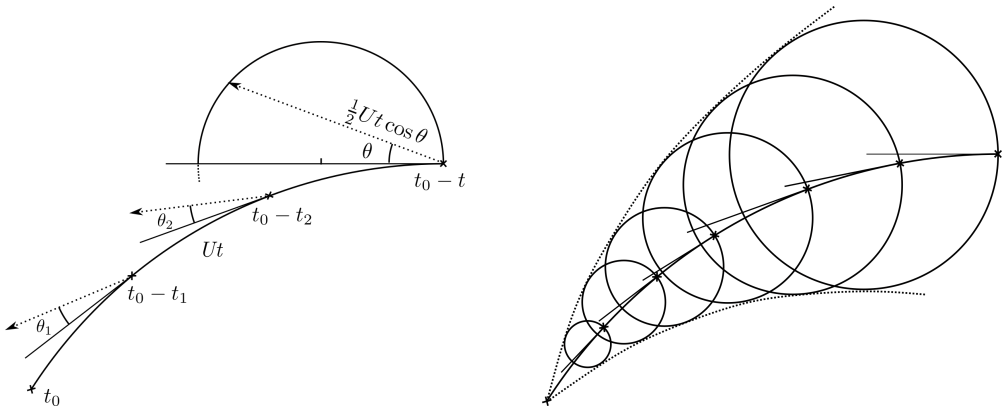


(A) At time t_0, displays the curve of wave energy (propagating with the group velocity V_g) generated at time $t_0 - t$.

(B) Region of disturbance for a straight course.

FIGURE 3.2: Simplified evaluation of the Kelvin angle $\alpha_c = 19°28'$ for a ship on a straight course in deep water. The visualisation idealises the ship as a point moving from right to left.

A similar evaluation can be made for a pressure point on a circular course, as shown in Fig. 3.3. The key difference here lies in the continuous change of definition for θ as observed from the Earth-fixed coordinate system as the ship moves. For a more thorough derivation of the disturbance pattern for a pressure point moving along a circular course (and hence a straight course as $R \to \infty$), reference is made to STOKER (1957).



(A) At time t_0, displays the curve of wave energy (propagating with the group velocity V_g) generated at time $t_0 - t$.

(B) Region of disturbance for a circular course.

FIGURE 3.3: Simplified evaluation of the wave pattern for a ship on a circular course in deep water. The visualisation idealises the ship as a point moving from right to left.

Far-field wave pattern

To describe the complete wave system generated by the ship, all regular waves satisfying Eq. (3.7) to Eq. (3.10) are summed up giving:

$$\zeta(x,y) = \int_{-\pi/2}^{\pi/2} |A(\theta)| \cos\left(\frac{g}{U^2\cos^2\theta}(x\cos\theta + y\sin\theta) + \epsilon(\theta)\right) d\theta, \tag{3.12}$$

where $|A(\theta)|$ and $\epsilon(\theta)$ are functions of the submerged ship shape. By introducing polar coordinates (r, α) as $x = r\cos\alpha$, $y = r\sin\alpha$ and complex form, Eq. (3.12) may be rewritten as

$$\zeta(r,\alpha) = \Re \int_{-\pi/2}^{\pi/2} A(\theta) \exp\left[i \frac{g}{U^2 \cos^2\theta} r\cos(\theta - \alpha)\right] d\theta. \tag{3.13}$$

As r increases, the integrand of Eq. (3.13) becomes highly oscillatory, and extra care is needed for its evaluation. Following the procedure presented by Faltinsen (2005), the integral will be evaluated using the method of stationary phase which is further explained in Sec. 3.4.1.

Let the exponential function in Eq. (3.13) read $\exp[irG(\theta)]$ where

$$G(\theta) = \frac{g\cos(\theta - \alpha)}{U^2\cos^2\theta}. \tag{3.14}$$

In accordance with the method of stationary phase, the main contribution to the integral is assumed to come from the point(s) where $G'(\theta) = 0$. It can be shown that

$$\frac{U^2}{g}\frac{dG(\theta)}{d\theta} = \frac{3\sin\alpha + \sin(2\theta - \alpha)}{2\cos^3\theta}, \tag{3.15}$$

which implies the points of stationary phase equal to the solutions of

$$3\sin\alpha + \sin(2\theta - \alpha) = 0. \tag{3.16}$$

Since $|\sin(2\theta - \alpha)| \leq 1$, any solution to Eq. (3.16) requires $|3\sin\alpha| \leq 1$, that is, $|\alpha|$ less than the Kelvin angle $\alpha_c = 19°28'$. For $0 \leq \alpha \leq \alpha_c$, there exist two solutions within the evaluated interval $\theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$:

$$\theta_1 = \frac{\alpha}{2} - \frac{1}{2}\arcsin(3\sin\alpha), \tag{3.17}$$

$$\theta_2 = \frac{\alpha}{2} - \frac{\pi}{2} + \frac{1}{2}\arcsin(3\sin\alpha). \tag{3.18}$$

Further, by the method of stationary phase described in Sec. 3.4.1, Eq. (3.13) may for $0 \leq \alpha \leq \alpha_c$ be approximated as

$$\zeta(r,\alpha) = \sum_{i=1}^{2} \Re\left[A(\theta_i)\left(\frac{2\pi}{r|G''(\theta_i)|}\right)^{\frac{1}{2}} \exp\left[i\left(rG(\theta_i) \pm \frac{\pi}{4}\right)\right]\right], \tag{3.19}$$

where the $\pm$ sign corresponds to the sign of $G''(\theta_i)$. Evaluating the second order derivative $G''(\theta)$ at $\theta = \theta_i$ after differentiating Eq. (3.15) gives

$$\left.\frac{d^2G(\theta)}{d\theta^2}\right|_{\theta=\theta_i} = \pm\frac{\sqrt{1 - 9\sin^2\alpha}}{|\cos^3\theta_i|}\frac{g}{U^2}, \tag{3.20}$$

where $+$ corresponds to θ_1 and $-$ to θ_2. The final expression for the wave elevation given $0 \leq \alpha \leq \alpha_c$ then reads

$$\begin{aligned}\zeta(r,\alpha) = \sqrt{\frac{2\pi}{k_0 r}}\left|1 - 9\sin^2\alpha\right|^{-1/4} \Re\Bigg\{ & A(\theta_1)|\cos\theta_1|^{3/2} \exp\left[i\left(k_0 r\frac{\cos(\theta_1 - \alpha)}{\cos^2\theta_1} + \frac{\pi}{4}\right)\right] \\ & + A(\theta_2)|\cos\theta_2|^{3/2} \exp\left[i\left(k_0 r\frac{\cos(\theta_2 - \alpha)}{\cos^2\theta_2} - \frac{\pi}{4}\right)\right]\Bigg\},\end{aligned} \tag{3.21}$$

where $k_0 = g/U^2$. The solution for $-\alpha_c \leq \alpha \leq 0$ is similar to Eq. (3.21), only reflected about the xz-plane. It should be noted that the solution presented in Eq. (3.21) following the method of stationary phase assumes $r \to \infty$, and that it gives an infinite value for the amplitude when $|\alpha| = \alpha_c$.

Divergent and transverse waves

The ship wave pattern consists of two wave systems; transverse and divergent waves. For the straight course scenario described above, the propagation angles are $-\theta_1$ and $-\theta_2$ for the transverse and divergent waves, respectively (FALTINSEN 2005). It is from the graph in Fig. 3.4 apparent that θ_i is always negative and that $|\theta_1| < |\theta_2|$.

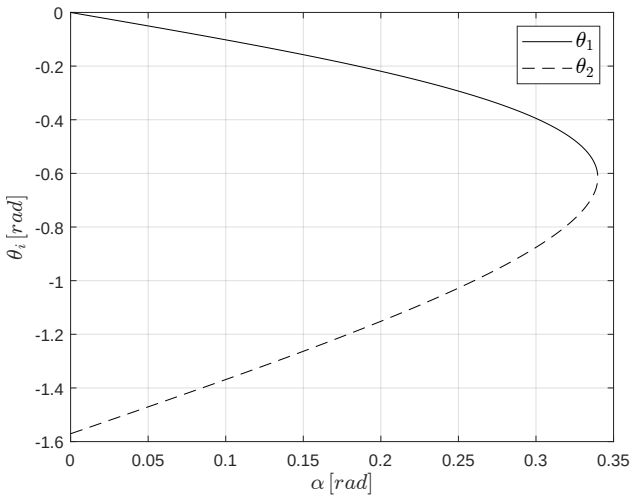


FIGURE 3.4: Angles of stationary phase θ_i versus the polar coordinate angle α for the wave pattern from a ship on a straight course in deep water according to Eq. (3.17) and Eq. (3.18).

To illustrate the formation of the transverse waves, take as a starting point the terms of Eq. (3.21) containing θ_1. If assuming $A(\theta_1)$ is real, the position of the transverse wave crests are given by

$$k_0 r \frac{\cos(\theta_1 - \alpha)}{\cos^2 \theta_1} + \frac{\pi}{4} = 2\pi n, \quad \text{for} \quad n = 0, 1, 2, ..., \tag{3.22}$$

where θ_1 and α are related through Eq. (3.17). Similarly, if assuming $A(\theta_2)$ is real and θ_2 given by Eq. (3.18), the position of the divergent wave crests may be expressed as

$$k_0 r \frac{\cos(\theta_2 - \alpha)}{\cos^2 \theta_2} - \frac{\pi}{4} = 2\pi n, \quad \text{for} \quad n = 0, 1, 2, ..., \tag{3.23}$$

with the resulting wave pattern following from Eq. (3.22) and Eq. (3.23) displayed in Fig. 3.5a.

STOKER (1957) (following the works of SRETENSKI (1946)[1]) shows that the curves of constant phase for a circular course, that is, the transverse and divergent waves, are given by

$$\begin{aligned} x/R &= \sin(\varkappa \cos\theta) - \frac{\varkappa}{2} \cos^2\theta \cos(\theta + \varkappa \cos\theta), \\ y/R &= 1 - \cos(\varkappa \cos\theta) - \frac{\varkappa}{2} \cos^2\theta \sin(\theta + \varkappa \cos\theta), \end{aligned} \tag{3.24}$$

where $\varkappa$ is a dimensionless parameter for which $\alpha = \varkappa \cos\theta$. For a fixed turning radius R, each

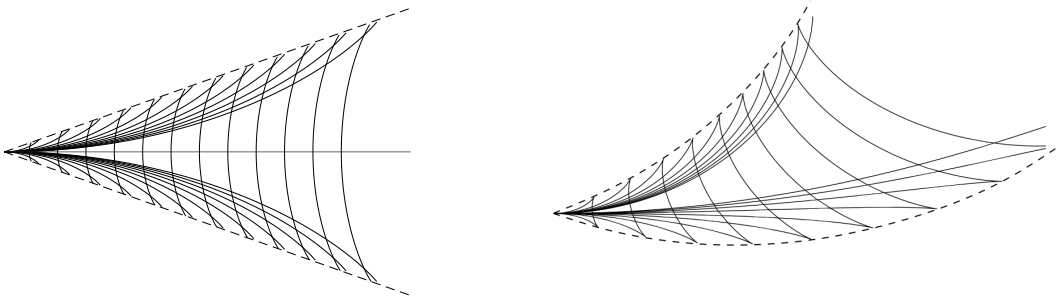
[1]This paper has not been obtained, but is cited by STOKER (1957, p. 234).

fixed value of $\varkappa$ is equivalent to fixing the phase and hence furnishing a curve. Fig. 3.5b shows the wave pattern from ten such curves, which when compared with photographs of actual cases are stated as *"given correctly by the theory, at least qualitatively"* by STOKER (1957).

The solution for a straight course is obtained by letting $R \to \infty$, giving the curves of constant phase as

$$\begin{aligned} x &= \frac{a}{2}(2\cos\theta - \cos^3\theta), \\ y &= -\frac{a}{2}\cos^2\theta\sin\theta, \end{aligned} \tag{3.25}$$

where fixing the quantity a is equivalent to fixing the phase. The pattern obtained from Eq. (3.25) is similar to that of Eq. (3.22) and Eq. (3.23).



(A) Straight course (B) Circular course

FIGURE 3.5: Divergent and transverse wave crests for a straight and circular course in deep water, as well as the outlined region of disturbance.

The amplitude of the transverse and divergent waves is dependent on the length Froude number $Fn_l = U/\sqrt{gL}$, where for $Fn_l >\approx 0.6$, the divergent waves will be dominant (FALTINSEN 2005). This is also indicated by amongst others LUNDE (1951) in his numerical computation of wave resistance contributions from divergent and transverse waves for a Wigley model at increasing velocity. For $Fn_l <\approx 0.5$, the phasing between the bow and stern waves will largely impact the amplitude of the transverse waves.

3.1.2 Ship waves in finite water depth

In the case of finite water depth, it is convenient to first introduce the non-dimensional depth Froude number as

$$Fn_h = \frac{U}{\sqrt{gh}}, \tag{3.26}$$

where h is the water depth. The dispersion relationship for regular waves in finite water depth reads

$$\omega^2 = gk\tanh kh, \tag{3.27}$$

and may be rewritten as

$$Fn_h^2 kh\cos^2\theta = \tanh kh \tag{3.28}$$

using Eq. (3.26) and the previously established condition for a steady wave pattern, $\omega = kU\cos\theta$.

An approximation to the ship waves appearing from a point impulse moving on the surface of water of finite water depth was first carried out by HAVELOCK (1908) using the method of stationary phase. He showed that for $Fn_h < 1$, the general wave pattern is much the same as for infinite water depth, with both transverse and divergent waves. However, the Kelvin angle was found to be highly dependent on the depth Froude number, increasing from $\alpha_c \approx 19°28'$ for

$Fn_h < 0.5 - 0.6$ to 90° at $Fn_h = 1$. For $Fn_h > 1$, the transverse wave system ceases to exist and the Kelvin angle decreases as shown in Fig. 3.6.

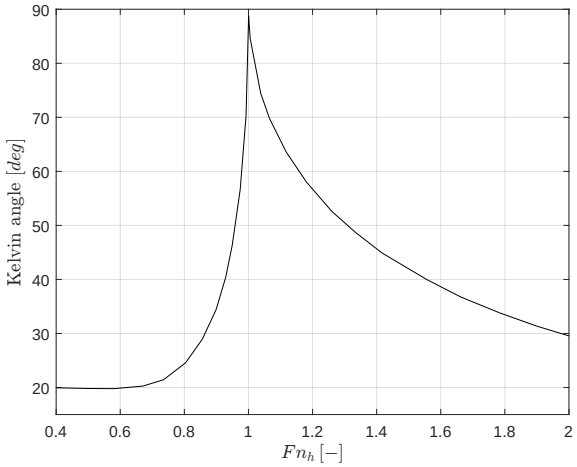


FIGURE 3.6: Kelvin angle as a function of depth Froude number for ship waves in finite water depth. Calculated based on linear theory with an infinite free-surface extent by YANG et al. (2001) (data extracted from graph).

Using the group velocity in finite water depth,

$$V_p = \frac{\omega}{k} = \left(\frac{g}{k}\tanh kh\right)^{1/2}, \tag{3.29}$$

$$V_g = \left(\frac{1}{2} + \frac{kh}{\sinh 2kh}\right) V_p, \tag{3.30}$$

an outline of the disturbance region may be obtained by adopting a similar approach as for deep water. Curves of wave energy are in Fig. 3.7 illustrated for increasing depth Froude numbers. As opposed to the circular regions described for deep water, the finite water regions are more ellipse-shaped. Due to the absence of the transverse wave system for $Fn_h > 1.0$, the minimum angle θ_0 for which the wave energy propagates is given by $\theta_0 = \arccos \frac{1}{Fn_h}$. Consequently, sharp discontinuities are present in the regions of disturbance, as shown in Fig. 3.7b.

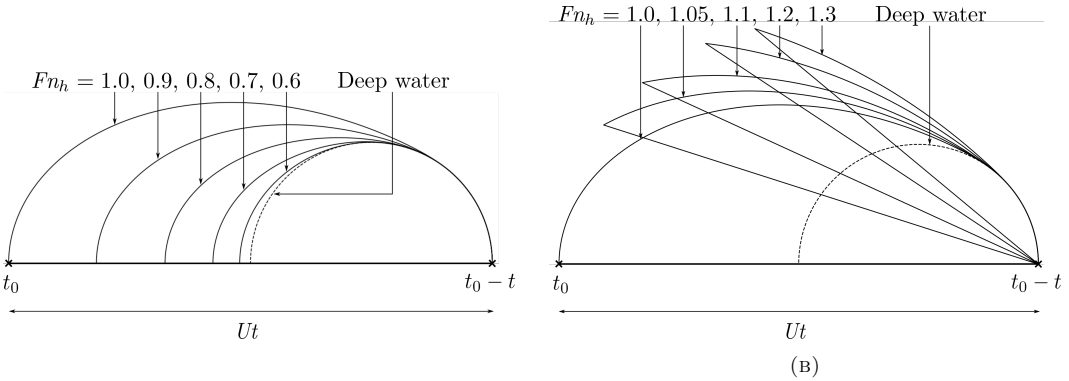


FIGURE 3.7: At time t_0, displays the curve of wave energy (propagating with the group velocity V_g) generated at time $t_0 - t$ for increasing depth Froude numbers.

Far-field wave pattern

As for deep water waves in Eq. (3.13), we start by representing the finite water wave system generated by the ship as

$$\zeta(r,\alpha) = \Re \int_{-\pi/2}^{\pi/2} A(\theta) \exp[irG(\theta)]\, d\theta, \tag{3.31}$$

where

$$G(\theta) = k(\theta)\cos(\theta - \alpha). \tag{3.32}$$

To obtain $k(\theta)$, and hence $G(\theta)$, a numerical solution of Eq. (3.28) is required, e.g. using Newtons method. By the method of stationary phase, a solution to Eq. (3.31) is given by

$$\zeta(r,\alpha) = \sum_{i=1}^{2} \Re \left[A(\theta_i) \left(\frac{2\pi}{r|G''(\theta_i)|} \right)^{\frac{1}{2}} \exp\left[i \left(rG(\theta_i) \pm \frac{\pi}{4} \right) \right] \right], \tag{3.33}$$

where the $\pm$ sign corresponds to the sign of $G''(\theta_i)$ and $i = 1, 2$ refers to the transverse and divergent waves, respectively. It can be shown that for $Fn_h > 1.0$ there will be no transverse waves, and hence Eq. (3.33) should be evaluated for $i = 2$ only.

3.1.3 Viscous correction factor

A correction factor for the viscous damping of far-field waves is following the suggestion of Tuck, Scullen et al. (2002) introduced as

$$V = \exp\left(-4\nu U k^3 \cos\theta (x + y\tan\theta)\right), \tag{3.34}$$

where $\nu = 0.0002\,\mathrm{m^2\,s^{-1}}$ represents the eddy viscosity effects produced by the vessel and the wave number k is obtained from Eq. (3.9) and Eq. (3.28) for infinite and finite water depth, respectively. If using the usual molecular kinematic viscosity for water, in the order of $10^{-6}\mathrm{m^2\,s^{-1}}$, the damping resulting from Eq. (3.34) is negligible.

The presented correction factor is obtained by the addition of a dissipative term to the linearised free-surface condition in Eq. (3.43), with the full derivation given in Appendix A. If disregarding the dependency on y, a factor of two separates it from the correction factor proposed by Lamb (1932, p. 625).

The damping effects from surface tension is considered negligible compared to the eddy viscosity, and is thus not accounted for. However, in the presence of surface tension a similar dissipative term may be added to the free-surface condition, with reference being made to amongst others Lamb (1932, p. 626), Tuck (1974) and Plesset and Whipple (1974).

3.2 Michell's theory

The linear thin-ship theory of J. H. Michell (1898) represents the ship geometry as a source distribution along the longitudinal centerplane of the vessel with the local source strength proportional to the longitudinal change of thickness. It proceeds with two assumptions; *(i)* an inviscid liquid and *(ii)* a thin ship, that is, the inclination of the tangent plane at any point of the ship's surface to the vertical median plane is small.

The neglect of friction is assumed to be of little consequence, since the inclination to the virtual tangent plane that the wave-making is dependent on will not be be largely altered by eddying water close to the side. Neglecting viscosity will to some extent affect the destroying of smaller waves,

but because the larger waves give the principal contribution to the resistance, the effect must be small (J. H. Michell 1898). Tuck, Scullen et al. (2002) describes the extreme diverging waves travelling nearly perpendicular to the ship's track as the most affected by the non-present viscosity, seeing as they are the shortest-wavelength, highest-frequency and slowest-moving waves.

The thin ship assumption will for most ships not be satisfied near the middle body. Neither the waves nor the resistance is assumed to be greatly affected by this, considering that the waves are assumed to arise from the bow and stern sections of the ship where the tangent to the surface is inclined to the direction of the ship's motion (J. H. Michell 1898).

Following is a brief description of the mathematical background of Michell's theory. The coordinate system is defined as in Fig. 3.8, where the hull is expressed as

$$y = \pm\eta(x, z). \tag{3.35}$$

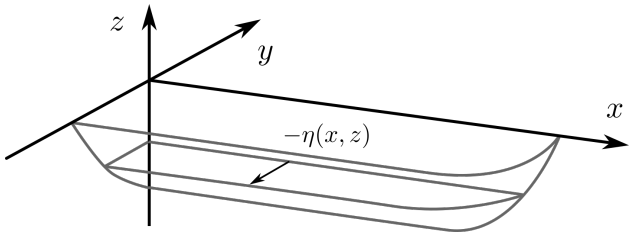


Figure 3.8: Coordinate system applied for Michell's theory. η is here describing the hull shape in negative y-direction, and thus negative.

Consider a ship with constant speed U on a straight course with the steady velocity potential

$$\Phi = Ux + \phi, \tag{3.36}$$

where ϕ is the velocity potential caused by the ship. The problem of finding a velocity potential of irrotational, incompressible fluid motion consist of finding a solution to Laplace's equation

$$\Phi_{xx} + \Phi_{yy} + \Phi_{zz} = 0 \tag{3.37}$$

with the relevant boundary conditions (Faltinsen 1990). Here, subscripts denote partial derivatives, i.e., $\Phi_x = \partial\Phi/\partial x$. For simplicity, it is assumed that the problem is symmetric about the plane $y = 0$. On the ship's hull $y = \eta(x, z)$, we require no flow through the hull surface using the Neumann boundary condition

$$\Phi_y = \Phi_x\eta_x + \Phi_z\eta_z. \tag{3.38}$$

Two conditions need to be satisfied on the free surface $z = \zeta(x, y)$. The kinematic free-surface condition—if assuming a steady system—is expressed as

$$\Phi_z = \Phi_x\zeta_x + \Phi_y\zeta_y \quad \text{on} \quad z = \zeta(x, y) \tag{3.39}$$

and requires that a fluid particle on the free surface stays on the free-surface. The dynamic free-surface condition gives that the water pressure is equal to the constant atmospheric pressure on the free-surface, which if using the Bernoulli equation and assuming steady state can be expressed as

$$2g\zeta + \Phi_x^2 + \Phi_y^2 + \Phi_z^2 = 0 \quad \text{on} \quad z = \zeta(x, y), \tag{3.40}$$

where g is the constant acceleration of gravity (Faltinsen 1990). Suitable boundary conditions are then assumed for the limits where $x, y \to \infty$ (Tuck 1989) and at the water bottom where

$$\Phi_z = 0 \quad \text{on} \quad z = -h. \tag{3.41}$$

The Neumann-Stokes problem formulated above is nonlinear due to the quadratic nature of the Bernoulli condition. Michell therefore proceeds with a full linearisation of both the body and free-surface conditions, justifying it on the basis of the thin-ship assumption previously described. It then follows that the new body boundary condition—which may be called a Michell condition—becomes

$$\phi_y = U\eta_x, \tag{3.42}$$

with a linearised combined free-surface condition expressed as

$$g\phi_z = U^2\phi_{xx}. \tag{3.43}$$

The latter condition may be called a Kelvin condition, meaning that the problem itself can be described as a "Michell-Kelvin" boundary value problem (Tuck 1989). In the derivation of J. H. Michell (1898), the exact solution to the problem is then solved using Fourier-transform methods and the assumption of infinite water depth. Newman (1977) showed that the deep water amplitude function $A(\theta)$ described in Eq. (3.12) can according to Michell's theory be expressed as

$$A(\theta) = \frac{2}{\pi} k_0 \sec^3\theta \iint_{cp} \eta_x(x,z) e^{k_0 \sec^2\theta (z + ix\cos\theta)}\, dz\, dx, \tag{3.44}$$

where $k_0 = g/U^2$ and cp denotes the centerplane of the hull. Eq. (3.44) is here dependent on η_x, the longitudinal derivative of the offset. This derivative is easily obtained for algebraically defined geometries such as the Wigley hull, but posing more of an issue for an arbitrary specified hull geometry. Using integration by parts while assuming that the offsets vanish at both ends, Tuck, Scullen et al. (1999a) rewrites Eq. (3.44) as

$$A(\theta) = -\frac{2i}{\pi} k_0^2 \sec^4\theta \iint_{cp} \eta(x,z) e^{k_0 \sec^2\theta (z + ix\cos\theta)}\, dz\, dx, \tag{3.45}$$

which is now solely dependent on the actual offsets η.

In finite water depth, the Michell formula for the far-field free-wave spectrum $A(\theta)$ in terms of the hull offsets is by Tuck, Scullen et al. (2000b) derived as

$$A(\theta) = \frac{2i}{\pi} \frac{k^2}{1 - k_0 h \sec^2\theta \operatorname{sech}^2 kh} \left[\iint_{cp} \eta(x,z) \frac{\cosh k(z+h)}{\cosh kh} e^{ikx\cos\theta}\, dxdz \right.$$
$$\left. - \frac{1}{ik\cos\theta} e^{ikx_s\cos\theta} \int_{z_s}^{0} \eta(x_s,z) \frac{\cosh k(z+h)}{\cosh kh} dz \right] \tag{3.46}$$

where k is given by Eq. (3.28) and $x = x_s$ at the stern. If there is no transom stern, the second term inside the bracket vanishes.

3.3 Wave resistance

The resistance caused by the ship-generated waves while on a straight course in calm water and constant velocity is called wave resistance, denoted R_w. The generated waves include both the far- and near-field wave patterns.

The complex amplitude function $A(\theta)$ can be related to the wave resistance using energy arguments, as is shown by Havelock (1934) and Newman (1977):

$$R_w = \pi\rho U^2 \int_0^{\pi/2} |A(\theta)|^2 \cos^3\theta \, d\theta. \tag{3.47}$$

Havelock (1934) proceeded to derive the relation for finite water depth as

$$R_w = \pi\rho U^2 \int_{\theta_0}^{\pi/2} |A(\theta)|^2 (\coth kh - kh \operatorname{csch}^2 kh) \cos^3\theta \, d\theta \tag{3.48}$$

where

$$\theta_0 = \begin{cases} \arccos \frac{1}{Fn_h} & \text{for } Fn_h > 1.0 \\ \theta_0 = 0 & \text{for } Fn_h < 1.0 \end{cases} \tag{3.49}$$

and k is obtained from Eq. (3.28). If using the amplitude functions derived from the linear thin ship theory of Michell, the deep water assumption gives that Eq. (3.45) should be applied with Eq. (3.47) while the finite water assumption requires Eq. (3.46) with Eq. (3.48).

The wave resistance coefficient C_w is defined as

$$C_w = \frac{R_w}{0.5\rho U^2 S}. \tag{3.50}$$

3.4 Numerical methods

3.4.1 Method of stationary phase

The method of stationary phase is a numerical method for computing highly oscillatory integrals, and is applicable to general Fourier integrals given that they have a point of stationary phase within the evaluated integral. It does only give asymptotic results, and for a fixed frequency the accuracy of the approximation is limited (Olver 2008). Although originally enunciated by Lord Kelvin (1887b), a brief review of the method will be given following the derivation of Newman (1977).

A general Fourier integral is assumed on the form

$$\int_{-\infty}^{\infty} f(x) \, e^{i\omega g(x)} \, dx, \tag{3.51}$$

where $f(x)$ and $g(x)$ are assumed to be arbitrary regular functions. The phase $\omega g(x)$ will for large values of ω change rapidly, and as a consequence, the integral will become highly oscillatory. Let there now exist a point where the first derivative of $g(x)$ is zero, that is, $g'(x_0) = 0$. Except for the vicinity of such a point x_0, the oscillations will tend to cancel each other out. The motivation behind the method of stationary phase is to evaluate the contribution from the vicinity of this point, assuming it is the principal contribution to the integral.

The function $g(x)$ is near $x = x_0$ expanded as a Taylor series on the form

$$g(x) = g(x_0) + (x - x_0)g'(x_0) + (x - x_0)^2 \frac{g''(x_0)}{2} + ... \tag{3.52}$$

and inserted into the original Fourier integral as

$$\int_{x_0+\epsilon}^{x_0-\epsilon} f(x)\, e^{i\omega[g(x_0)+(x-x_0)^2 g''(x_0)/2]}\, dx. \tag{3.53}$$

The term involving the first derivative is in Eq. (3.53) cancelled, while ϵ is a small value which does not depend on ω, but on g and f. The integral is simplified by substituting

$$\xi = (x - x_0)\left[\frac{|\omega g''(x_0)|}{2}\right]^{\frac{1}{2}}, \tag{3.54}$$

which with accordingly substituted boundaries and variable of integration becomes

$$\frac{f(x_0)}{[0.5|\omega g''(x_0)|]^{\frac{1}{2}}} \int_{-\epsilon[0.5|\omega g''(x_0)|]^{\frac{1}{2}}}^{\epsilon\left[0.5|\omega g''(x_0)|\right]^{\frac{1}{2}}} e^{i[\omega g(x_0) \pm \xi^2]}\, d\xi, \tag{3.55}$$

where $\pm$ has the same sign as $g''(x_0)$ and f has been approximated by its value at x_0. In the limits $\omega \to \infty$, the integral can be evaluated using the definite Fresnel integral with boundaries approaching infinity:

$$\int_0^\infty \sin(\xi^2)\, d\xi = \int_0^\infty \cos(\xi^2)\, d\xi = \frac{1}{2}\sqrt{\frac{\pi}{2}}. \tag{3.56}$$

Following, it can be shown that the final solution to Eq. (3.51) using the method of stationary phase is

$$f(x_0)\left[\frac{2\pi}{|\omega g''(x_0)|}\right]^{\frac{1}{2}} e^{i\left[\omega g(x_0) \pm \frac{\pi}{4}\right]}, \tag{3.57}$$

which is valid for sufficiently large values of ω if $g''(x_0) \neq 0$. In the case of multiple points of stationary phase within the evaluated interval, the contribution from each point may be treated separately, according to Eq. (3.57), and added together to give the final contribution.

3.4.2 Filon's integration formula

The Filon quadrature formula introduced by Filon (1930) is used for numerical evaluation of trigonometric integrals on the form

$$C = \int_{x_0}^{x_{2n}} f(x)\cos(\omega x)\, dx, \quad S = \int_{x_0}^{x_{2n}} f(x)\sin(\omega x)\, dx. \tag{3.58}$$

For smooth $f(x)$, and especially for large values of ω, the Filon formula is advantageous over usual numerical integration since the number of points which need to be tabulated depends on the behaviour of $f(x)$ rather than on $f(x)\sin(\omega x)$ or $f(x)\cos(\omega x)$ (Fosdick 1968).

The Filon quadrature formula for the integrals defined in Eq. (3.58) is expressed as follows (Abramowitz and Stegun 1964):

$$C = h\left[\alpha(\omega h)(f_{2n}\sin(\omega x_{2n}) - f_0\sin(\omega x_0)) + \beta(\omega h)C_{2n} + \gamma(\omega h)C_{2n-1}\right] + E_c, \tag{3.59}$$

$$S = h\left[\alpha(\omega h)(f_0\cos(\omega x_0) - f_{2n}\cos(\omega x_{2n})) + \beta(\omega h)S_{2n} + \gamma(\omega h)S_{2n-1}\right] + E_s, \tag{3.60}$$

where $h = \delta x$ and

$$C_{2n} = \sum_{i=0}^{n} f_{2i} \cos(\omega x_{2i}) - \frac{1}{2} \left[f_{2n} \cos(\omega x_{2n}) + f_0 \cos(\omega x_0) \right], \tag{3.61}$$

$$C_{2n-1} = \sum_{i=1}^{n} f_{2i-1} \cos(\omega x_{2i-1}), \tag{3.62}$$

$$S_{2n} = \sum_{i=0}^{n} f_{2i} \sin(\omega x_{2i}) - \frac{1}{2} \left[f_{2n} \sin(\omega x_{2n}) + f_0 \sin(\omega x_0) \right], \tag{3.63}$$

$$S_{2n-1} = \sum_{i=1}^{n} f_{2i-1} \sin(\omega x_{2i-1}), \tag{3.64}$$

$$\alpha(\theta) = \frac{1}{\theta} + \frac{\sin 2\theta}{2\theta^2} - \frac{2\sin^2 \theta}{\theta^3}, \tag{3.65}$$

$$\beta(\theta) = 2 \left(\frac{1 + \cos^2 \theta}{\theta^2} - \frac{\sin 2\theta}{\theta^3} \right), \tag{3.66}$$

$$\gamma(\theta) = 4 \left(\frac{\sin \theta}{\theta^3} - \frac{\cos \theta}{\theta^2} \right). \tag{3.67}$$

α, β and γ should for small values of θ be replaced with the power series

$$\alpha(\theta) = \frac{2\theta^3}{45} - \frac{2\theta^5}{315} + \frac{2\theta^7}{4725} - ..., \tag{3.68}$$

$$\beta(\theta) = \frac{2}{3} + \frac{2\theta^2}{15} - \frac{4\theta^4}{105} + \frac{2\theta^6}{567} - ..., \tag{3.69}$$

$$\gamma(\theta) = \frac{4}{3} - \frac{2\theta^2}{15} + \frac{\theta^4}{210} - \frac{\theta^6}{11340} + ..., \tag{3.70}$$

to avoid the loss of significant figures due to cancellation in the original expressions. E_c and E_s in Eq. (3.59) and Eq. (3.60) are the error terms associated with Filon's formula, and may be found in Abramowitz and Stegun (1964).

Filon-Trapezoidal integration formula

The original Filon formula is derived on the assumption that $f(x)$, rather than the complete integrand, may be stepwise approximated by parabolas. The Filon-Trapezoidal method proposed by Tuck (1967) assumes that—rather than parabolas—$f(x)$ may be stepwise approximated as linear.

Following Tuck (1967), the Filon-Trapezoidal rule will be presented for the approximate evaluation of Fourier integrals such as

$$F = \int_{-T}^{T} f(x)\, e^{i\omega x}\, dx, \tag{3.71}$$

although the method may be easily adapted to fit the integrals in Eq. (3.58) using Euler's formula. The Filon-Trapezoidal rule gives for Eq. (3.71) the approximation

$$F = \sum_{n=-N}^{N} \alpha_n e^{i\omega x_n} f(x_n), \tag{3.72}$$

with weights

$$\alpha_{-N} = \frac{1 + i\omega h - e^{i\omega h}}{\omega^2 h^2}, \tag{3.73}$$

$$\alpha_n = \left(\frac{\sin \frac{1}{2}\omega h}{\frac{1}{2}\omega h} \right)^2, \quad n \neq \pm N, \tag{3.74}$$

$$\alpha_N = \frac{1 - i\omega h - e^{-i\omega h}}{\omega^2 h^2}. \tag{3.75}$$

As $\omega h \to 0$, the weights of the Filon-Trapezoidal method tend to the trapezoidal values

$$\alpha_n = 1, \quad n \neq \pm N, \tag{3.76}$$

$$\alpha_{\pm N} = \frac{1}{2}. \tag{3.77}$$

3.5 Reflection and diffraction of water waves

When operating in restricted waters such as harbors, piers, canals and rivers, a significant portion of the resulting ship wave pattern is due to reflection and diffraction from nearby obstacles. A brief overview will be given with regards to the case of a breakwater, although the thesis will neglect the effects of diffraction in the consequent implementation.

By making use of the mathematical analogous problems in the diffraction of light, and in particular the solution by Sommerfeld (1896), Penney et al. (1952) investigated the diffraction of sea waves round the end of a long straight breakwater. Their assumptions were; *(i)* nearly sinusoidal wave-profiles, *(ii)* the wave height is small relative to the wave length, *(iii)* uniform water depth and *(iv)* ideal breakwaters, that is, the waves are completely reflected.

Consider waves travelling at an angle θ_0 relative to the breakwater and divide the surrounding area into three regions as shown in Fig. 3.9. The region S, for which $0 \leq \theta \leq \theta_0$, defines the lee side of the breakwater. The region Q is given for $\theta_0 \leq \theta \leq 2\pi - \theta_0$ and is little affected by the breakwater while R defines the region containing waves reflected by the breakwater, that is, for $2\pi - \theta_0 \leq \theta \leq 2\pi$. The incident waves are present in region Q and R.

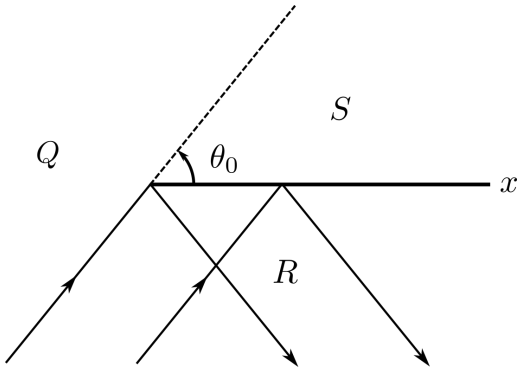


Figure 3.9: Notation for the regions Q, R and S for oblique incidence of waves on a breakwater as given by Penney et al. (1952).

The total effect of the breakwater can be considered a sum of the shadow and reflection effects in S and R originating from geometrical optics[2] and the diffraction effects present in all three regions. Penney et al. (1952) showed that the wave fronts near the geometric shadow at $\theta = \theta_0$

[2]Geometrical optics is a branch of optics where light is described by rays.

will experience increasing phase-lag with increasing x, and hence bend round towards the lee side of the breakwater. When drawn, as can be seen in PENNEY et al. (1952, pp. 242, 246) and DAEMRICH and KOHLHASE (1978, p. 657), they closely resemble arcs of circles with center at the open end. Upon moving out of S, the diffraction waves will experience a phase-shift of one-half wave-length. A similar phase shift will also occur upon entering R, as well as for $\theta = \pi$ if given a rigid breakwater. For large radius r, the diffraction waves are shown to be declining at a rate of $r^{-\frac{1}{2}}$, except for at the boundaries $\theta = \theta_0$ and $\theta = 2\pi - \theta_0$.

Reflection and transmission coefficients

The thesis will limit the obstacles considered to a set of rigid vertical concrete walls ranging from the seabed to above the surface, independent of the depth. Consequently, the reflection and transmission coefficients of all walls are defined as unity and zero, respectively (CHOWDHURY et al. 2017). To prevent the reflection coefficient of unity to result in an infinite number of reflections, only the initial reflection of the incident waves is considered. This limitation is assumed valid in open waters and near shore, but not in very restricted waters such as canals.

The interaction between the wake and other vessels are considered negligible for all vessels, hence giving a reflection coefficient of zero and a transmission coefficient of unity.

Chapter 4

Method

4.1 Ship waves for a straight course

4.1.1 Deep water

The wave pattern for a ship on a straight course is implemented using the amplitude function following Michell's thin ship theory in Eq. (3.45) with the expression for far-field waves in infinite water depth as given by Eq. (3.21).

For the numerical evaluation of Eq. (3.45), the double integral is first reduced to a pair of separate integrals with respect to length and depth following the procedure of Tuck, Scullen et al. (1999a).

For all sections of x along the ship, evaluate the integral

$$F(x,\theta) = \int_D \eta(x,z) \exp(k_0 z \sec^2\theta)\, dz, \tag{4.1}$$

with vertical limits of z from the lowest point of the hull section ($z < 0$) up to the waterline $z = 0$. For a submerged hull, artificial points may be specified between the top of the hull and the free surface where $\eta(x,z) = 0$.

Proceed by evaluating the pair of integrals

$$P(\theta) = \int_L F(x,\theta) \cos(k_0 x \sec\theta)\, dx, \tag{4.2}$$

$$Q(\theta) = \int_L F(x,\theta) \sin(k_0 x \sec\theta)\, dx, \tag{4.3}$$

along the body from stern to bow, where $F(x,\theta)$ is given by Eq. (4.1). It follows that Eq. (3.45) can be rewritten as

$$A(\theta) = -\frac{2i}{\pi} k_0^2 \sec^4\theta \left[P(\theta) + iQ(\theta)\right]. \tag{4.4}$$

The integrals in Eq. (4.2) and Eq. (4.3) could in principle be numerically solved using standard methods such as Simpson's rule with sufficiently small step size. However, and especially as $\theta \to \infty$, the integrand becomes highly oscillatory, suggesting that extra care should be taken. Further described in Sec. 3.4.2, Filon's quadrature is a procedure with similar accuracy as Simpsons's rule, but better suited for integrals with a highly oscillatory integrand.

Divide the length of the ship into N_x equidistant segments of length Δx such that $x = x_i$, $i = 0, 1, 2, ..., N_x - 1, N_x$. If assuming no contribution from the stern (no transom stern) and bow,

that is, $\eta = 0$ which implies $F = 0$ at the bow $x = x_0$ and stern $x = x_{N_x}$, Filon's quadrature then gives

$$P(\theta) = \sum_{i=1}^{N_x-1} \omega_i F(x_i, \theta) \cos(k_0 x \sec\theta)\Delta x, \tag{4.5}$$

$$Q(\theta) = \sum_{i=1}^{N_x-1} \omega_i F(x_i, \theta) \sin(k_0 x \sec\theta)\Delta x, \tag{4.6}$$

with even and odd weights as respectively

$$\omega_{2i} = \frac{3K + K\cos 2K - 2\sin 2K}{K^3}, \tag{4.7}$$

$$\omega_{2i+1} = \frac{4(\sin K - K\cos K)}{K^3}, \tag{4.8}$$

where $K = k_0 \Delta x \sec\theta$.

For Eq. (4.1), if we assume η itself, rather than $\eta \exp(k_0 z \sec^2\theta)$ is piecewise linear, then the Filon-Trapezoidal rule described in Sec. 3.4.2 is advantageous to Simpson's rule (TUCK, SCULLEN et al. 1999a). Each vertical section along the length of the ship is divided into N_z equidistant segments of height Δz such that $z = z_i$, $i = 0, 1, 2, ..., N_z - 1, N_z$, giving the quadrature

$$F(x, \theta) \approx \sum_{j=0}^{N_z} \omega_j \eta(x, z_j) \exp(k_0 z_j \sec^2\theta)\Delta z, \tag{4.9}$$

with weights

$$\omega_0 = \frac{e^K - 1 - K}{K^2} \quad \text{for} \quad j = 0, \tag{4.10}$$

$$\omega_{N_z} = \frac{e^{-K} - 1 + K}{K^2} \quad \text{for} \quad j = N_z, \tag{4.11}$$

$$\omega_j = \frac{e^K + e^{-K} - 2}{K^2} \quad \text{for} \quad j \neq 0, N_z, \tag{4.12}$$

where $K = k_0 \Delta z \sec^2\theta$ (not equal to the definition of K in the above Filon quadrature in x-direction). Although the depth, i.e., N_z, according to Eq. (4.9) may vary over the length of the ship, it is convenient to define a constant set of waterplanes $z = z_j$, where $\eta(x, z) = 0$ is defined below the hull for varying draught.

In reality, vessels will also start planing for $Fn_l >\approx 1.0$ (FALTINSEN 2005) and hence drastically changing the hull's wetted area. This is not considered in the present implementation, meaning that care should be taken when evaluating results for $Fn_l >\approx 1.0$.

4.1.2 Finite water depth

The implementation of the wave pattern for a ship on a straight course in finite water depth is in principle similar to that of deep water, but applying different expressions. Since the solution may no longer be expressed analytically, care should be taken in the numerical procedure applied, especially with regards to the phase function $G(\theta)$ found in Eq. 3.33.

To significantly improve the computational effort required, the roots of $G'(\theta)$ used in the method of stationary phase are preemptively computed, tabulated and stored rather than computed in real-time.

The finite water amplitude function in Eq. (3.46) is rewritten as

$$A(\theta) = -\frac{2i}{\pi}\frac{k^2}{1 - k_0 h \sec^2\theta \operatorname{sech}^2 kh}[P(\theta) + iQ(\theta)], \tag{4.13}$$

where $P(\theta)$ and $Q(\theta)$ are expressed as

$$P(\theta) = \sum_{i=1}^{N_x - 1} \omega_i F(x_i, \theta)\cos(kx\cos\theta)\Delta x, \tag{4.14}$$

$$Q(\theta) = \sum_{i=1}^{N_x - 1} \omega_i F(x_i, \theta)\sin(kx\cos\theta)\Delta x, \tag{4.15}$$

and are evaluated along the body from the bow to the stern using the same assumptions as for deep water. k is given by the finite water dispersion relation presented in Eq. (3.27), while the weights ω_i are given by Eq. (4.7) and Eq. (4.8), but with $K = k\cos\theta\,\Delta x$.

Similarly,

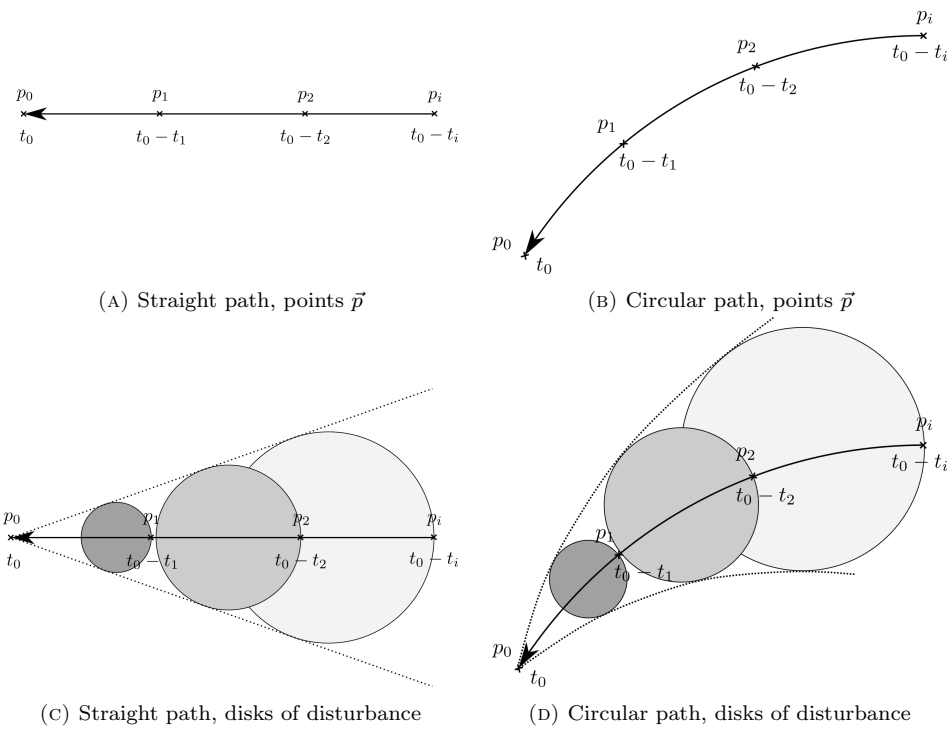
$$F(x, \theta) \approx \sum_{j=0}^{N_z} \omega_j \eta(x, z_j)\frac{\cosh k(z + h)}{\cosh kh}\Delta z \tag{4.16}$$

is evaluated at each vertical section along the length of the ship, also here using the same assumptions and limits as stated for deep water. The weights ω_j are given by Eq. (4.10), Eq. (4.11) and Eq. (4.12), but with $K = k\Delta z$.

4.2 Ship waves for a circular course

Contrary to most codes developed to compute ship waves, a requirement for the present implementation is to give the wave pattern following a vessel on an arbitrary course. With the additional limitation of real-time calculation for up to several vessels at a time, no analytical solution similar to that of a straight course has successfully been obtained. The following procedure will therefore adapt the steady solution for a straight course to an arbitrary course using the regions of disturbance previously described (see Fig. 3.2 and Fig. 3.3). If proven sufficient, a major benefit of this method is that the steady wave patterns may be calculated prior to the simulation, hence saving valuable computational effort during the real-time simulation.

Consider the ship's path as described by a series of points $\vec{p} = \{p_0, p_1, p_2, ..., p_N\}$ as shown in Fig. 4.1, where each point p_i denotes the position of the ship at time t_i. We let $t = t_0$ represent the current position and require $t_i > t_{i+1}$ for all i. In addition, we assume the instantaneous velocity and heading to be known for all points. If the effect of finite water depth is to be included, the water depth should be given as well.



(A) Straight path, points $\vec{p}$ (B) Circular path, points $\vec{p}$

(C) Straight path, disks of disturbance (D) Circular path, disks of disturbance

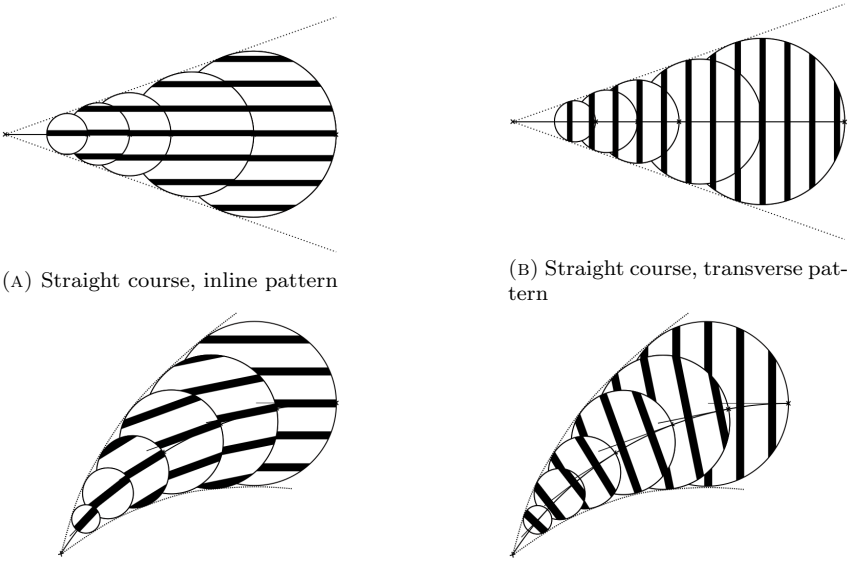
FIGURE 4.1: Outlined procedure for the wave pattern for a ship on a circular course in deep water. Example of points $\vec{p}$ and their corresponding proposed disks of disturbance. For finite water depth, the same principle applies but with non-circular disks of disturbance.

Similar to the simplified evaluation of the region of disturbance in Sec. 3.1.1, circular disks of disturbance can be computed for all points $\vec{p}$ using the expression for group velocity (Eq. (3.8) for infinite water depth and Eq. (3.30) for finite). The disks obtained from the steady solution with velocity U_i at a straight course (Sec. 4.1) are then rearranged and rotated to fit the new arbitrary path, exemplified by the circular course in Fig. 4.1. In the case of overlapping disks, the disk with origin at point p_i should always dominate those with origin at points p_j, $j > i$.

In finite water depth, due to the complexity involved in computing the ellipse-shaped disk of disturbance, the proposed approach is to estimate each disk as an ellipse. By doing so, analytical equations may be used to check if a point is inside the ellipse, rather than numerically computing a new value of k for each check. However, as will be discussed in Sec. 5.1.2, the simplification will introduce some unwanted features as well.

Using the procedure stated, the illustrations in Fig. 4.2 indicates how the inline and transverse patterns are affected by the transformation. We notice how the largest discrepancies are found for the transverse wave pattern, especially at the outer edge of the turn where the previously transverse stripes are separated into several noncontinuous pieces. This discontinuity will improve when decreasing the distance between each point p_i, although still be present. For the inline stripes, a decrease in length between the points will give seemingly continuous lines along the track of the ship.

It should be noted that procedures similar to the proposed has not been found in existing publications, with the closest being that of KHAN (1994). It is nevertheless assumed to give a sufficiently accurate representation of the wave pattern for an arbitrary path using the minimum amount of computational effort.



(A) Straight course, inline pattern

(B) Straight course, transverse pattern

(C) Circular course, inline pattern

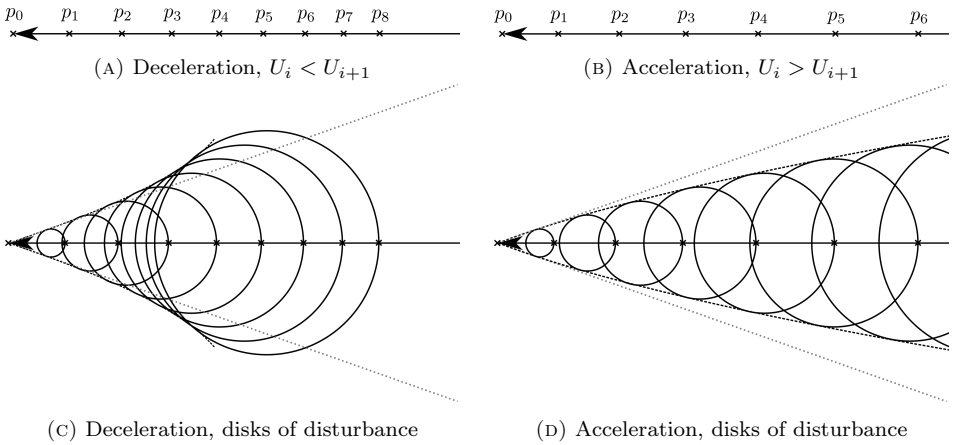
(D) Circular course, transverse pattern

FIGURE 4.2: The individual effect on transverse and inline stripes from the adaption of the wave pattern to a circular course.

4.3 Ship waves for an accelerating vessel

Given a direct application of the method proposed for ship waves on a circular course in Sec. 4.2, no additional adjustment is needed to account for the acceleration of the vessel. This is because each disk of disturbance, whether in deep or finite water, is computed based on the instantaneous velocity U_i stored for every point along the path.

Fig. 4.3 illustrates the effect on the region of disturbance for a decelerating and accelerating vessel, comparing it to the deep water Kelvin angle. The exact shape of the disturbance region will depend on the magnitude of acceleration, although the main principle is the same. The suggested wake outlines does also coincide with those presented by KHAN (1994, pp. 37, 38).



(A) Deceleration, $U_i < U_{i+1}$

(B) Acceleration, $U_i > U_{i+1}$

(C) Deceleration, disks of disturbance

(D) Acceleration, disks of disturbance

FIGURE 4.3: Outlined procedure for the wave pattern for an accelerating vessel in deep water. The dotted grey line represents the Kelvin angle in deep water. For finite water depth, the same principle applies but with non-circular disks of disturbance.

4.4 Code structure

A flow diagram describing the general code structure of the main program is given in Fig. 4.4. It describes how the main features such as accounting for multiple vessels and wall reflections are managed, but does not cover the details. Flow diagrams describing the approaches for computing wall reflections and contribution to wave elevation are attached in Appendix B.

The foundation of the main program follows from the choice of parallel computing; a single function call should be necessary to obtain the elevation at a point (x, y), without needing to reiterate over the same point during the same frame. The thesis will not investigate the effect on runtime performance for alternative code structures, suggesting that more optimal configurations may be possible.

Alongside the input point of interest, (x, y), the following information is assumed known and available:

- the total number of vessels, including a list describing each vessels path. The information consists of for each point along the path; the incident velocity, the heading relative to a global coordinate system, the water depth and a time stamp.
- the hull geometry for each vessel, formatted as described in Sec. 4.1.
- the current time.
- the total number of walls, as well as their locations formatted as pairs of points.

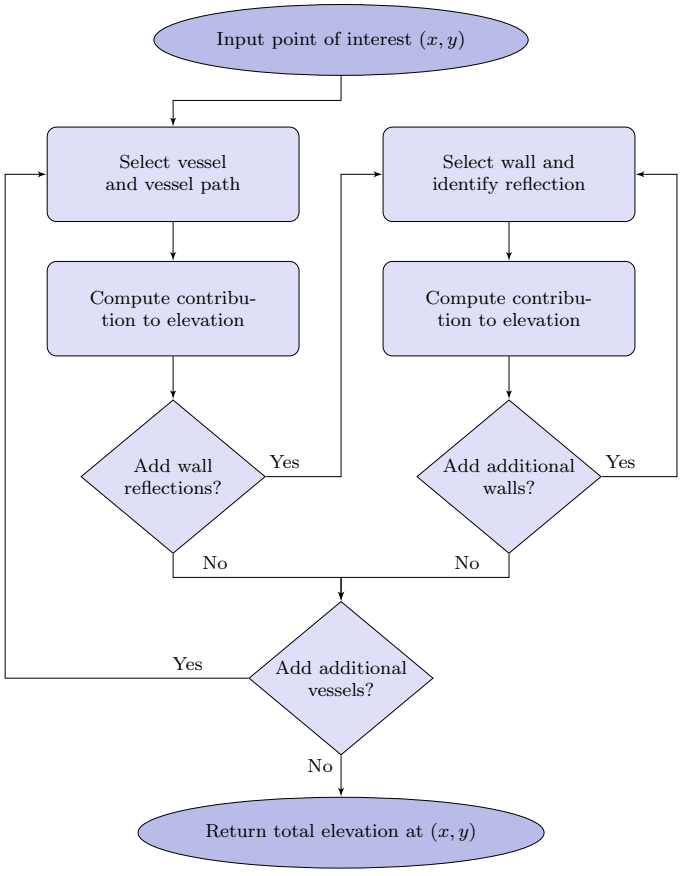


FIGURE 4.4: Flow diagram indicating general code structure of the main program.

4.5 Ship models

4.5.1 Wigley hull

The Wigley parabolic model is an algebraically defined hull shape commonly used in numerical and experimental investigations of ship wave pattern and ship resistance (see e.g. KAJITANI et al. (1983) and LUNDE (1951)) due to its uncomplicated shape. For $y > 0$, the hull is expressed as

$$\frac{y}{L} = \frac{B}{2L}\left(1 - \left(\frac{z}{D}\right)^2\right)\left(1 - \left(\frac{x}{0.5L}\right)^2\right), \tag{4.17}$$

where L is the length, B is the beam and D is the draft of the ship. The body plan and profile view of the Wigley hull is displayed in Fig. 4.5 with the corresponding hull characteristics given in Tab. 4.1. The parameters $B/L = 0.75/8$ and $D/L = 1/16$ are specified according to LUNDE (1951) to facilitate comparison of wave resistance coefficients, and used in the thesis unless stated otherwise.

TABLE 4.1: Hull characteristics of the Wigley model. Δx and Δy describes the mesh according to Sec. 4.1.

L	[m]	8.00
B	[m]	0.75
D	[m]	0.50
Δx	[m]	0.50
Δy	[m]	0.05

(A) (B)

FIGURE 4.5: Body plan (A) and visualisation (B) of the Wigley model.

Wave resistance

It follows from the mathematically defined geometry of the Wigley hull that the expressions for wave resistance introduced in Sec. 3.3 may be simplified significantly. These formulations will therefore serve as a verification of the more general procedures previously introduced.

According to LANDWEBER (1979), the Michell integral for the wavemaking resistance in deep water may be reduced to

$$R_w = \frac{16\rho B^2 U^2}{\pi}\int_0^{\pi/2} F(\xi)G(\eta)\cos\theta\, d\theta, \tag{4.18}$$

where

$$F(\xi) = \frac{(\sin\xi - \xi\cos\xi)^2}{\xi^4}, \tag{4.19}$$

$$G(\eta) = \left(1 - \frac{2}{\eta^2} + \frac{2}{\eta}\left(1 + \frac{1}{\eta}\right)e^{-\eta}\right)^2, \tag{4.20}$$

and

$$\begin{aligned} \xi &= \frac{1}{2}K_0 L\sec\theta, \\ \eta &= K_0 D\sec^2\theta, \\ K_0 &= g/U^2. \end{aligned} \tag{4.21}$$

Care should be taken when evaluating the above integral as $F(\xi)$ varies rapidly with θ for small length Froude numbers.

Yang (2002) goes on to show the wave resistance for a Wigley hull in finite water depth is given by

$$R_w = \frac{4\rho U^2}{\pi}\int_{\theta_0}^{\pi/2} \frac{(P^2+Q^2)K_0\cos\theta}{\cosh^2 K_0 Fn_h^2\cos^2\theta - 1}\,d\theta, \tag{4.22}$$

where K_0 now is the solution of $K_0 Fn_h^2\cos^2\theta - \tanh K_0 = 0$, $P = 0$ by symmetry and

$$\begin{aligned} Q = 2B\frac{\sin\xi - \xi\cos\xi}{\xi^2} \\ \times \frac{\sinh(K_0)(K_0^2D^2-2) + 2\cosh(-K_0D+K_0)K_0D + 2\sinh(-K_0D+K_0)}{K_0^3D^2}. \end{aligned} \tag{4.23}$$

The limit θ_0 is here obtained from Eq. (3.49), ξ is defined as

$$\xi = \frac{1}{2}K_0 L\cos\theta, \tag{4.24}$$

and all length dimensions are nondimensionalised by the water depth h.

4.5.2 Parabolic hull

The parabolic hull is another algebraically defined hull shape much used due to its simplicity and high satisfaction of the thin ship assumptions. It is amongst others applied in the experiments of Sharma (1969), which the Michell implementation will be compared against. The parabolic hull is defined by

$$y = \frac{B}{2}\left(1 - \left(\frac{x}{0.5L}\right)^2\right). \tag{4.25}$$

Unless stated otherwise, the hull characteristics in Tab. 4.2 are applied throughout the thesis.

Table 4.2: Hull characteristics of the parabolic model. Δx and Δy describes the mesh according to Sec. 4.1.

L	[m]	2.00
B	[m]	0.05
D	[m]	0.15
S	[m^2]	1.333
Δx	[m]	0.10
Δy	[m]	0.05

4.5.3 DTMB 5415

The DTMB 5415 is a high-speed destroyer-type hull including both a sonar dome and a transom stern. It was conceived as a preliminary design for a Navy surface combatant, but has never been built in full-scale. It is however much used as a benchmark ship for numerical codes, amongst others by Lindenmuth et al. (1991), Simman (2008, 2014) and Tuck, Scullen et al. (2001b).

The 5415 model is in light of the present implementation of ship waves too complicated to expect results comparable with experimental data, mainly due to the simplified procedure applied for the near-field wave pattern, as well as the transom stern which can not yet be accounted for in the computation. The model provides nevertheless a great hull shape for testing of variable draught over the length of the hull.

Table 4.3: Hull characteristics of the DTMB 5415 model. Δx and Δy describes the mesh according to Sec. 4.1.

Lwl	[m]	5.726
B	[m]	0.768
D	[m]	0.248
S	[m^2]	4.786
Δx	[m]	0.2
Δy	[m]	0.025

(a) (b)

Figure 4.6: Body plan (a) and visualisation (b) of the DTMB 5415 destroyer. The hull geometry was obtained from Simman (2008).

4.6 Unity

Unity is a cross-platform game engine with a built-in IDE (Integrated Development Environment) used to create two- and three-dimensional games, as well as interactive simulations and experiences. It is the software of choice for implementing the theory described in the thesis, mainly to ensure compatibility with the visual simulator Gemini (Vasstein and Skarshaug 2021).

An in-depth explanation of the full features used in Unity will not be provided, seeing as it is considered out-of-scope for the objective of the thesis. However, aspects relevant to the theoretical procedure and its implementation will be briefly mentioned.

4.6.1 Parallel computing

To ensure a sufficiently effective procedure, the adaption of parallel computing is a key element. It refers to the process of breaking down larger problems into small, independent problems that may be solved simultaneously. In the current context, the larger problem is to visualise the complete

far-field ship wave pattern, while the smaller problems may be considered computing the elevation at a specific point on the grid.

Assuming a separation of the problem into smaller pieces is possible, the high instruction throughput and memory bandwidth of the graphics processing unit (GPU) compared to the central processing unit (CPU) may be taken advantage of to severely increase the efficiency. That is, while the CPU is designed to excel at performing a sequence of operations as fast as possible, the GPU is designed to excel at performing thousands of operations in parallel (NVIDIA 2022).

The explicit nature of Michell's thin ship theory for a steady wave pattern is for this reason beneficial, requiring no additional customization to enable parallel computing. However, to realize its full potential, the additional features such as ship waves for a turning vessel and wave reflection from walls must also be developed as parallelizable.

4.6.2 High-Level Shader Language

The use of parallel computing is within Unity performed using *compute kernels*[1] written in the C-like High-Level Shader Language (HLSL) (MICROSOFT 2021).

If compared to e.g. Python, HLSL does not facilitate easy incorporation of external functions through packages. As a consequence, it is limited to the use of intrinsic functions such as `sin()`, `exp()` and `abs()` as well as custom functions. In principle, these are sufficient tools for any implementation, although it does require additional effort and knowledge when developing. An example relevant to the thesis is the implementation of a root-finding function. Using the Python-library *SciPy*, `fsolve()` is available; a root-finding function based on the numerical subroutine library MINPACK (MORÉ et al. 1980). To achieve a similarly optimised algorithm in HLSL, time and effort must be allocated to become familiar with the procedure followed by writing a new implementation. Alternatively, efficiency and robustness can be sacrificed by selecting a lesser optimized algorithm, in this case, Newtons method.

Unlike for the CPU, the accuracy of computations performed on a GPU need not satisfy the specified precision, but is to a certain degree dependent on the hardware itself. E.g., a minimal reproducible example relevant to the thesis using 32-bit floating point variables run on an *Intel(R) UHD Graphics 620* GPU gives a deviation in the order of 10^{-5} when compared to the results obtained from the more advanced *Nvidia GeForce RTX 3070 Ti* GPU. In general such deviations does not affect the solution, but as was found when computing points of stationary phase with regards to Eq. (3.33), it may introduce visible errors in the computed wave pattern. Care should therefore be taken where high-accuracy computations are necessary.

[1] A compute kernel is a routine compiled for high throughput accelerators such as GPUs.

Chapter 5

Results and discussion

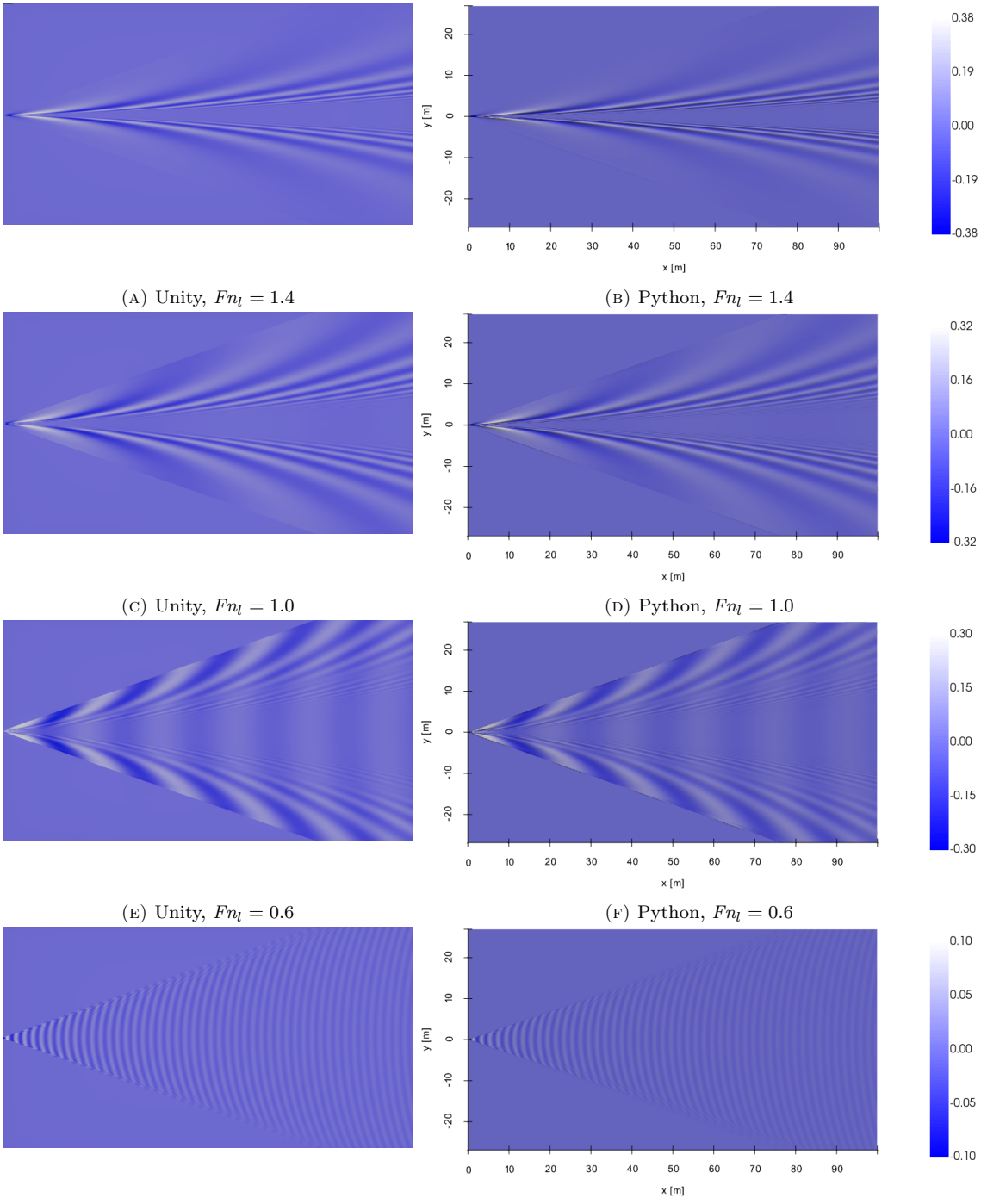
The presented results will contain visualisations using both Unity and Python as computing environments, despite the main algorithm being alike for the two. Python is mainly applied where additional processing of the wave pattern or its components is required, e.g., when computing the wave pattern, but will also serve as a verification of the results obtained from Unity. Unless otherwise stated, all visualisations are obtained from Unity, considering a main objective of the thesis being the development of a procedure applicable for this software in particular.

5.1 Ship waves for a straight course

5.1.1 Deep water

Computed wave patterns for the Wigley model described in Sec. 4.5.1 for various Froude numbers are presented in Fig. 5.1. Great agreement is found between the results obtained from Unity and Python for all length Froude numbers. The same color bar limits has been defined for both software, with the minor color difference observed being due to the ambient lightning in Unity.

The dependence on Froude number for the transverse and divergent wave systems described in Sec. 3.1.1 are found to be well captured. This is most notable for $Fn_l = 0.2$ where the transverse waves are clearly dominating, and similarly for the divergent waves at $Fn_l = 1.4$. As previously mentioned, care should be taken when evaluating results for $Fn_l >\approx 1.0$ since planing effects are not considered. The general trends are nevertheless considered representative.



(A) Unity, $Fn_l = 1.4$

(B) Python, $Fn_l = 1.4$

(C) Unity, $Fn_l = 1.0$

(D) Python, $Fn_l = 1.0$

(E) Unity, $Fn_l = 0.6$

(F) Python, $Fn_l = 0.6$

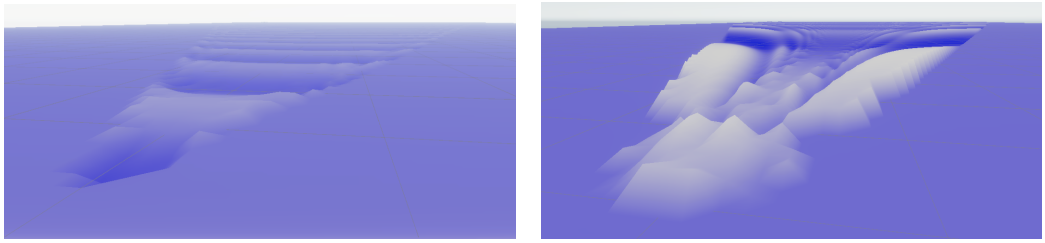
(G) Unity, $Fn_l = 0.2$

(H) Python, $Fn_l = 0.2$

FIGURE 5.1: Computed wave patterns for the Wigley model on a straight course at $Fn_l = 0.2, 0.6, 1.0, 1.4$.

The plots in Fig. 5.1 display both the near- and far-field wave pattern, despite the implemented expression for wave elevation only being valid for the far-field (actually in the limits $r \to \infty$). As a consequence, and as shown in Fig. 5.2, the resulting near-field wave pattern includes several unrealistic features such as large and abrupt changes in wave elevation. Potential solutions to this problem include the implementation of a separate near-field approximation of the wave pattern, or alternatively an artificial damping of the wave elevation in the near vicinity of the ship. The former procedure is by far the most accurate, but it is also the most computationally expensive.

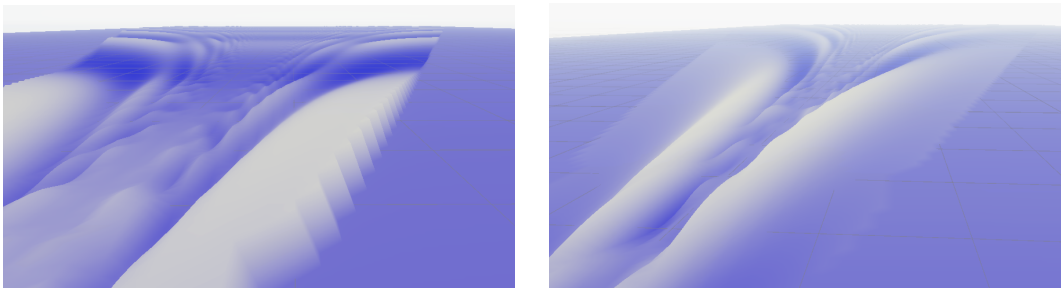
The importance of accurate modelling of the near-field ship waves in the simulator should be further investigated before a decision is made.



(A) $Fn_l = 0.2$ (B) $Fn_l = 0.6$

FIGURE 5.2: Computed near-field wave pattern for the Wigley model on a straight course. The implementation applies a far-field assumption, introducing inaccuracies in the near field.

Another undesirable feature of the present implementation is that the wave elevation becomes singular along the Kelvin lines $y = \pm x \arctan \frac{1}{3}$ and are exactly zero outside the Kelvin angle. This is illustrated in Fig. 5.3, where the singularity at the Kelvin angle has been removed using extrapolation in the close vicinity of the boundary. However, the current extrapolation does not remove the discontinuity at the Kelvin line between the wave pattern and the surrounding surface due to transverse waves. As is shown for $Fn_l = 1.0$, the divergent waves does not introduce a discontinuity, implying the issue is only present for $Fn_l < 1.0$.



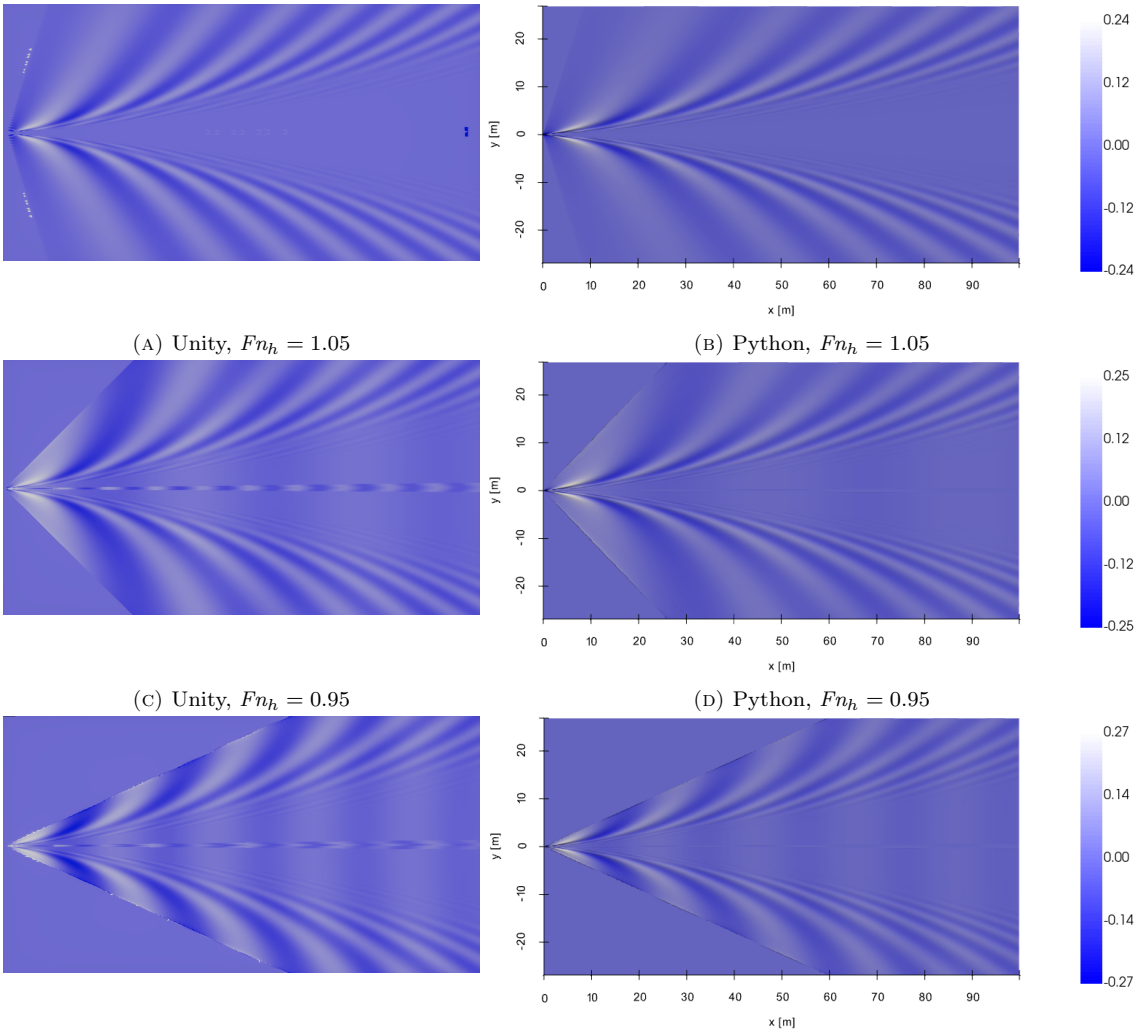
(A) $Fn_l = 0.6$ (B) $Fn_l = 1.0$

FIGURE 5.3: Computed wave pattern near the Kelvin angle for the Wigley model on a straight course.

5.1.2 Finite water

Wave patterns for the Wigley model at $Fn_h = 0.8, 0.95, 1.05$ are displayed in Fig. 5.4. For the subcritical cases, the enveloping wedge and the transverse wavelength increases with increasing depth Froude number. At the critical depth Froude number the transverse wave system ceases to exist and the enveloping wedge is at its maximum. For supercritical depth Froude numbers, here illustrated by $Fn_h = 1.05$, the enveloping wedge decreases with increasing depth Froude numbers.

Some irregularities are observed in the near vicinity of the track due to wrongful evaluation of the wave pattern for very small θ, which is especially visible for the Unity solution given that Unity uses 32-bit floating point precision versus the 64-bit precision of Python. Although not implemented, the deviations may be removed using e.g. a linear interpolation of the elevation at each side of the track. This is assumed valid since the wave system near the track is dominated by the transverse waves, with any divergent waves present being highly damped.



(A) Unity, $Fn_h = 1.05$

(B) Python, $Fn_h = 1.05$

(C) Unity, $Fn_h = 0.95$

(D) Python, $Fn_h = 0.95$

(E) Unity, $Fn_h = 0.8$

(F) Python, $Fn_h = 0.8$

FIGURE 5.4: Computed wave patterns for the Wigley model on a straight course in finite water depth at $Fn_h = 0.8$, 0.95, 1.05 and $Fn_l = 0.68$.

The Unity solution is for $Fn_h = 0.8$, 1.05 in Fig. 5.4 seen to produce singularities for angles slightly larger than the Kelvin angle. This occurs when a point is wrongfully evaluated as inside the region of disturbance, hence computing the elevation where it should not. As for the case of infinite water depth, care must therefore be taken when computing the elevation near the Kelvin angle.

Fig. 5.5 gives a close-up of the wave pattern for a supercritical depth Froude number. A spike pattern is observed near the Kelvin angle, as well as close to the vessel stern. This pattern is a consequence of the claw-shaped regions of disturbance applied when evaluating where to compute the wave elevation and not given $Fn_h > 1.0$, as is illustrated in Fig. 3.7b. The distance between the spikes is dependent on the distance between the path points for which the regions of disturbance originate from, meaning that the spikes will vanish as the distance between the path points approaches zero.

As described in Sec. 4.6, the results computed in Unity are dependent on the hardware. Fig. 5.6 demonstrates how the same wave pattern as in Fig. 5.4c gives significant errors when computed using an *Intel(R) UHD Graphics 620* GPU rather than on the *Nvidia GeForce RTX 3070 Ti* GPU used elsewhere in the thesis. The errors consist of singularities, low-amplitude noise and

discontinuities along diagonal lines originating at the vessel.

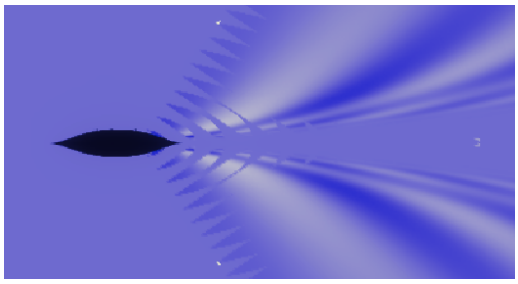


FIGURE 5.5: Region of disturbance in supercritical depth Froude numbers, here for $Fn_h = 1.2$. The "claw"-shaped disturbance regions are in accordance with those presented in Fig. 3.7.

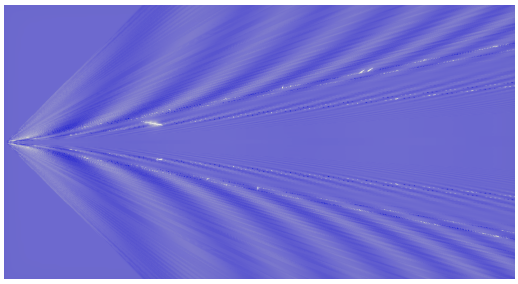
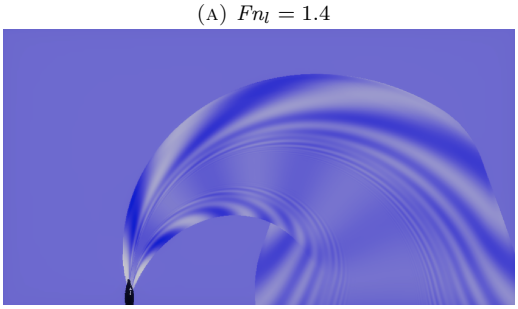


FIGURE 5.6: GPU dependent deviations in the wave pattern for $Fn_h = 0.95$. Results computed using an *Intel(R) UHD Graphics 620* GPU.

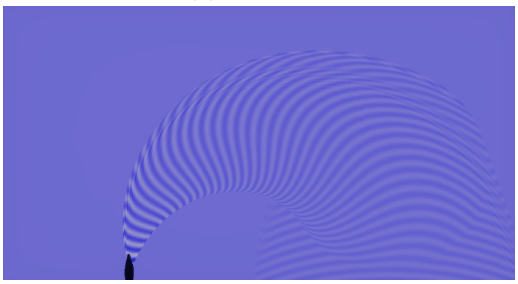
5.2 Ship waves for a circular course

(A) $Fn_l = 1.4$

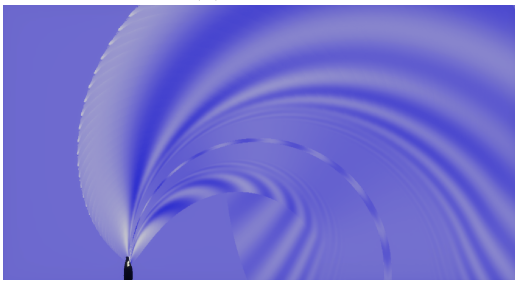
(B) $Fn_l = 1.0$



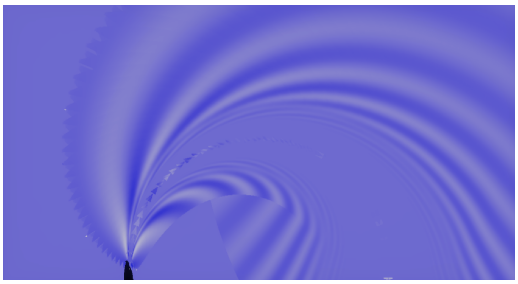
(C) $Fn_l = 0.6$



(D) $Fn_l = 0.2$



(E) $Fn_h = 0.95$, $Fn_l = 0.6$



(F) $Fn_h = 1.05$, $Fn_l = 0.6$

FIGURE 5.7: Computed wave patterns for the Wigley model on a circular course with turning radius $R = 25\,\mathrm{m}$ at $Fn_l = 0.2, 0.6, 1.0, 1.4$ and for $Fn_h = 0.95$, 1.05.

Following the procedure described in Sec. 4.2, the wave pattern for a ship on a circular course is displayed in Fig. 5.7. At first glance, the transverse and divergent wave systems are seemingly

following the expected pattern described in Fig. 3.5. The following paragraphs will however discuss the errors observed given a closer look at the plots.

Since the proposed procedure is not limited to circular courses, but rather capable of computing the wave pattern for an arbitrary course, additional visualisations for several such scenarios are given in Appendix C.

As predicted, the divergent wave systems are found to be noticeably less affected by the adaption to a circular course than the transverse wave system. To illustrate this, the shape of the transverse wave system for a Wigley model on a circular course with $Fn_l = 0.2$ is highlighted in Fig. 5.8. Instead of the crescent shapes predicted by the theory (Fig. 3.5b), the appearing shapes may be described as wavy, with the main deviations found on the inner side of the turn. Because the consequence of the deviations to the simulator (and thus the sensors observing the wave pattern) are unknown, developing a solution to the problem is not of immediate priority. The issue should nevertheless be revised in further work.

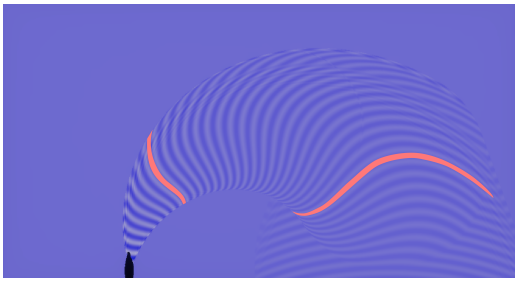
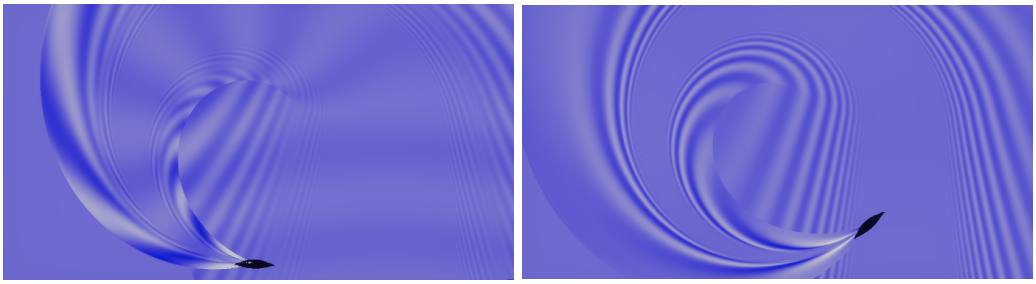


FIGURE 5.8: Transverse wave pattern for a circular course at low Froude number, $Fn_l = 0.2$.

As a consequence of the procedure approximating the wave pattern for a ship on a circular course, interference of waves originating from the same vessel is impossible. For scenarios where the vessel is moving in more or less a general direction, this fact does not pose an issue. However, it will for a vessel performing an extended turn, here illustrated by Fig. 5.9. To what degree this lack of interference affects the solution is dependent on mainly two factors; how large is the turning radius and how large are the waves. For sufficiently small wave amplitude and large turning radius, the waves generated during the first turn will be attenuated within the arrival of the next set of waves. Increasing wave amplitude and decreasing turning radius will hence reinforce the consequence of not accounting for interference, although no critical limit can be defined at this point. Considering the issue being limited to certain scenarios, it should not be of highest priority for further work.



(A) $Fn_l = 0.6$ (B) $Fn_l = 1.0$

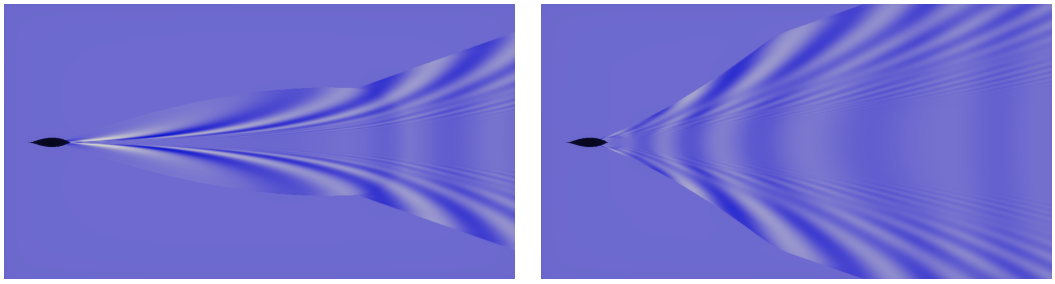
FIGURE 5.9: Computed wave patterns for the Wigley model on an extended circular course.

5.3 Ship waves for an accelerating vessel

For the wave pattern following an accelerating vessel, no relevant literature has been found for comparison. The results will therefore be discussed with regards to the simplified region of disturbance presented in Sec. 4.3. A qualitative discussion of the wave system will also be provided.

Similar to Fig. 4.3, the wave envelope is found to curve towards the track when accelerating and curve away from the track when decelerating. The rate at which the envelope curves is dependent on the magnitude of the acceleration.

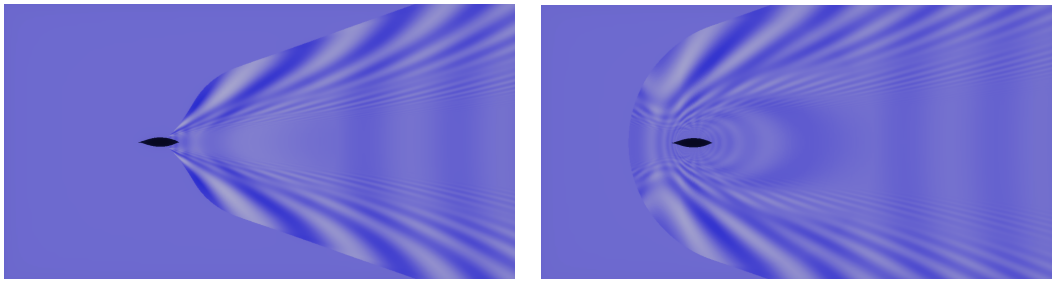
The behaviour of the transverse and divergent wave systems during acceleration may not be properly verified without experimental or analytical models. Intuitively though, as observed for the accelerating vessel in Fig. 5.10a, the transition from having both transverse and divergent waves to a divergent wave dominated system is as expected when increasing the length Froude number. On the contrary, the wake of the decelerating vessel in Fig. 5.10b is believed to be less accurate. As the vessel slows down, the chosen approach fails to consider how the wave energy catches up to the vessel and causes interference, this due to the prioritised disk of disturbance being the closest to the vessel and moving backwards.



(A) (B)

FIGURE 5.10: Ship waves for the Wigley model moving at $5.315\,\mathrm{m\,s^{-1}}$ starting to (A) accelerate at $1.0\,\mathrm{m\,s^{-2}}$ and (B) decelerate at $0.15\,\mathrm{m\,s^{-2}}$.

For a rapidly decelerating vessel, as shown in Fig. 5.11, the wave energy will quickly catch up to the vessel and surpass it. The region of disturbance is for this case assumed realistic, but the divergent waves arising in the region directly ahead of the vessel is not. They appear as a consequence of using disks of disturbance containing the steady state solution, which will always be computed based on the assumption that the vessel is ahead of the wave pattern. A potential solution could be to identify cases where the deceleration leads to wave energy surpassing the vessel, and then selectively remove the divergent wave components from the solution.



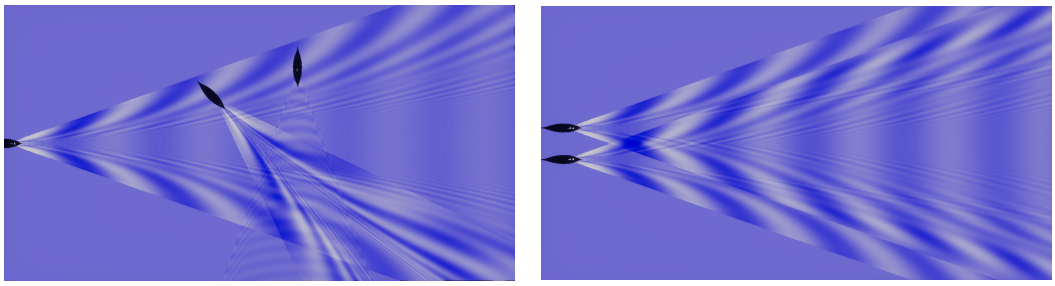
(A) (B)

FIGURE 5.11: Ship waves for a rapidly decelerating vessel. In (A), the vessel has just started decelerating, while at (B), the vessel has almost come to a full stop.

5.4 Ship waves for multiple vessels

Using the principle of superposition for linear theory, Fig. 5.12 displays the wave pattern generated by multiple vessels with different heading and velocity. Although not shown, the program is also capable of accounting for difference in water depth, path curvature and hull geometry for each individual vessel. In theory, since the contribution from each vessel to the elevation is computed using a for-loop, an unlimited number of vessels may be considered at once. In practice however, the computational cost will serve as the limiting factor.

If assuming no interaction between the hulls of a multihull vessel, it follows that the wave pattern of e.g. a catamaran may be comprised of two identical hulls with a fixed length of separation. An example using two identical Wigley hulls is shown in Fig. 5.12b.

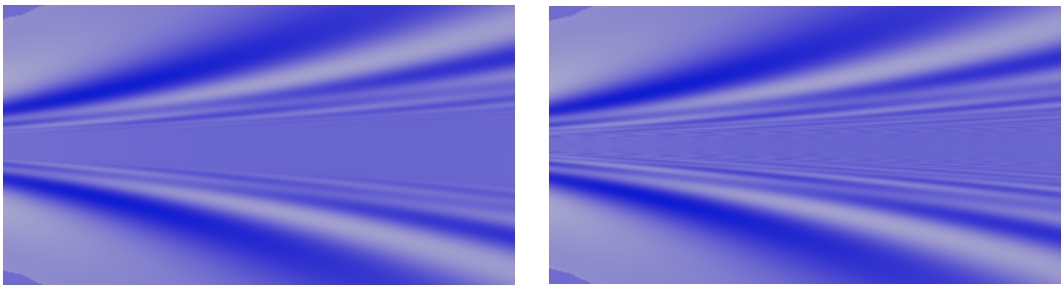


(A) (B)

FIGURE 5.12: Superposition of ship waves for multiple vessels.

5.5 Viscous correction factor

Comparing the deep water wave pattern with and without the viscous correction factor (Eq. (3.34)), Fig. 5.13 displays how the high-frequency divergent waves nearly perpendicular to the track are most affected by the damping. Due to the lightning effects affected by the rapidly changing normal vectors present in this area if not accounting for viscosity, it is considered a crucial element when aiming to correctly visualise the wave pattern using a game engine.



(A) With viscous damping (B) Without viscous damping

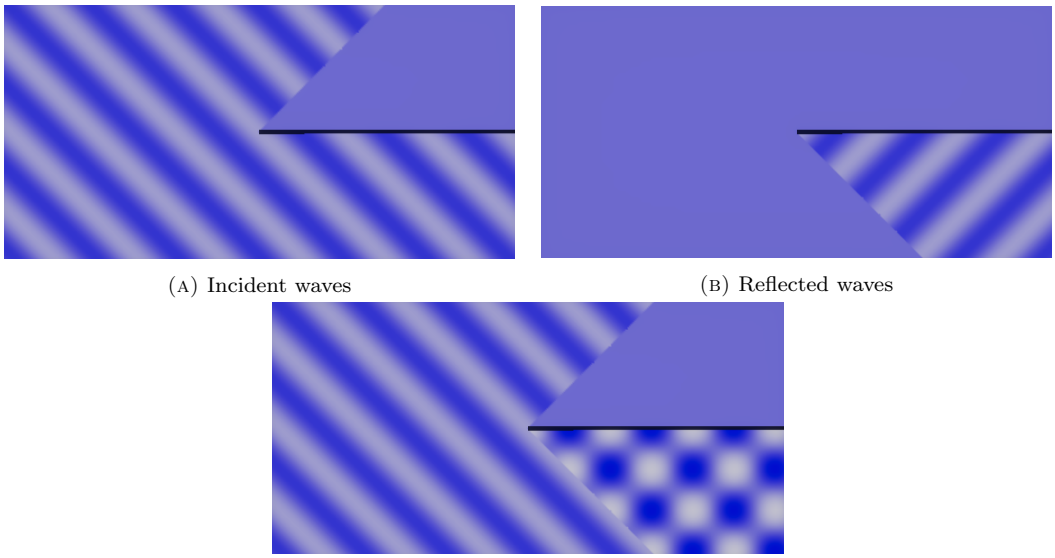
FIGURE 5.13: Comparing the deep water wave pattern with and without a viscous correction factor, $Fn_l = 0.95$.

In the derivation of Eq. (3.34) for finite water depth as found in Appendix A, an approach similar to that of deep water was applied. Intuitively, due to the non-zero velocity of the water particles at the sea bottom in finite water, this is not considered a valid assumption. As similarly discussed by LAMB (1932), the resistance due to slipping at the bottom implies that the dissipation may no longer be calculated as if the motion were irrotational. The present finite water viscous damping factor should therefore be considered an initial estimate to be revised by further work.

5.6 Reflection

5.6.1 Sine waves

To verify the implementation for reflection of ship waves, the wave pattern for sine waves incident at 45° to a breakwater is first presented in Fig. 5.14. Since diffraction is not considered, the total wave pattern is given as the sum of the incident and reflected waves. Similar to the illustration by Penney et al. (1952, p. 246), the geometric shadow is observed at the lee side of the breakwater, while the reflected and incident waves at the windward side combine to create short-crested waves travelling to the right.



(A) Incident waves (B) Reflected waves

(C) Combined wave pattern

Figure 5.14: The wave pattern for sine waves incident at 45° to a breakwater. (A) shows the incident waves, (B) the reflected waves and (C) the combined wave pattern.

A limitation of the implementation is that only the initial reflection of the incident waves are considered. Fig. 5.15 displays a scenario where two breakwaters are located next to each other, forming a canal in between where in reality, multiple consecutive reflections would occur between the canal walls. However, considering the current neglect of diffraction, it was not deemed a priority to account for multiple wall reflections.

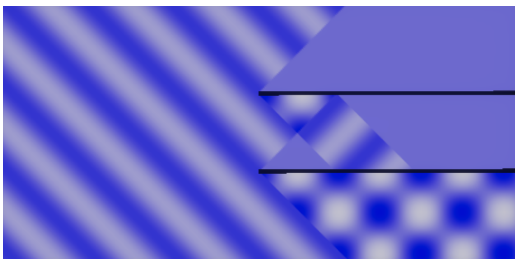


Figure 5.15: The wave pattern for sine waves incident at 45° to two consecutive breakwaters. Only the initial reflection of the incident waves are considered.

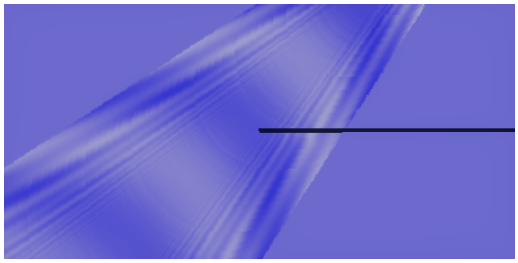
5.6.2 Ship waves

Ship wave patterns for a vessel travelling at an angle $\Psi = 0°$ and 45° to the breakwater is examined in Fig. 5.16. Disregarding the absence of diffraction waves, multiple inaccuracies are observed.

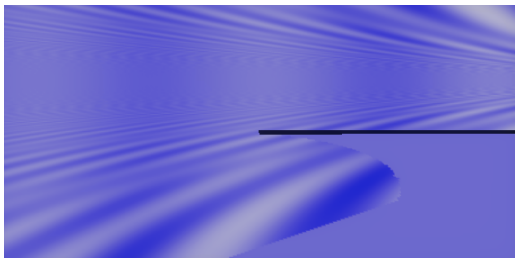
Firstly, the incident waves for $\Psi = 45°$ are shown to be unaffected by the breakwater. The cause for this behaviour lies in the assumption that the elevation may be computed as a remapping of the fully developed steady state wave pattern for a straight course. In the derivation of the steady wake in Sec. 3.1, the wave characteristics were explained using an infinite sum of long-crested linear waves with different propagation angles arising as the vessel moves. On the lee side, this assumption is invalid for most wave components originating from before the vessel passed the breakwater. I.e., the incident wave elevation present on the lee side of the breakwater is correctly shown as present, but the representation of it as fully developed is not realistic. The same issue is found for $\Psi = 0$, but the presence of a shadow zone makes it less distinct.

Another consequence of the inability to represent the wave elevation as anything but fully developed where present is an artificial increase in total energy. This is most easily observed for the combined wave pattern at $\Psi = 0°$ where the wave elevation is of similar height on both sides of the breakwater. The increase in total energy is in terms of the visualisation equivalent to an artificial increase in the observed wave elevation.

The above inaccuracies are discussed for the case of ship waves interacting with a breakwater. However, a breakwater represents an extreme when considering wave reflection and diffraction from walls, and is hence not representative for the usual environment in which the ship wave pattern is to be computed. For less complex wall arrangements, the errors introduced by the current implementation will decrease noticeably.

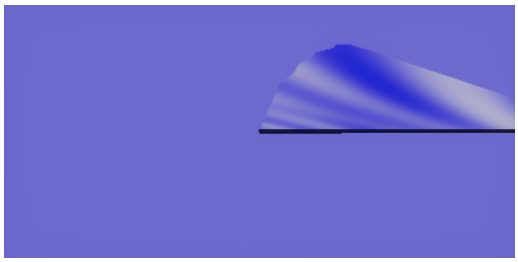


(A) Incident waves, $\Psi = 45°$

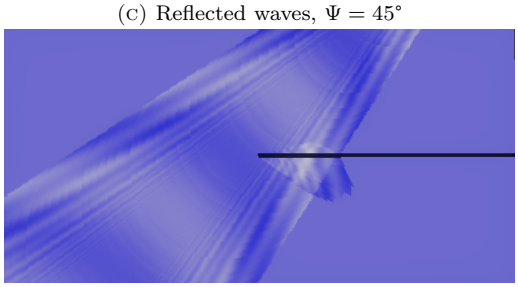


(B) Incident waves, $\Psi = 0°$

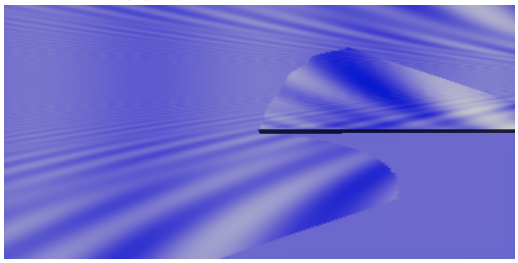
(C) Reflected waves, $\Psi = 45°$



(D) Reflected waves, $\Psi = 0°$



(E) Combined wave pattern, $\Psi = 45°$



(F) Combined wave pattern, $\Psi = 0°$

FIGURE 5.16: The wave pattern following a vessel travelling at an angle Ψ relative to the breakwater. $Fn = 0.6$ and infinite water depth is considered.

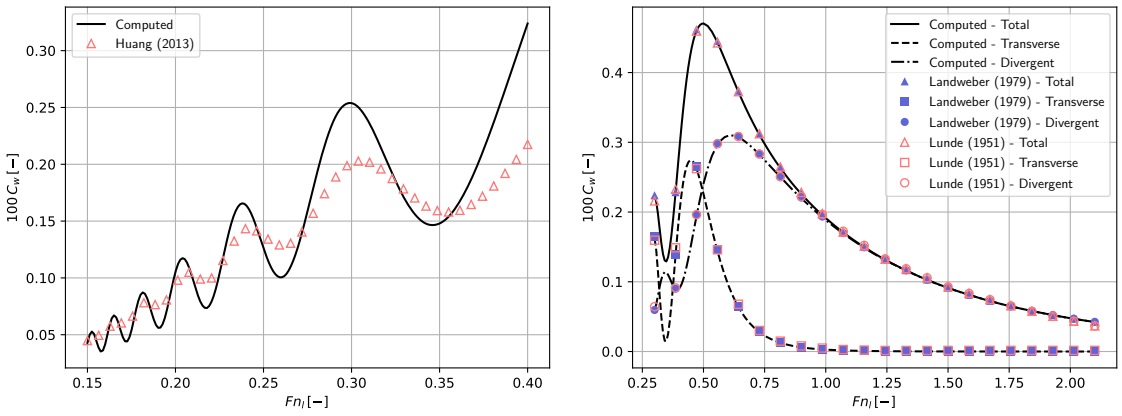
5.7 Wave resistance

Accurate computation of wave resistance is by itself not a requirement for the present implementation of ship waves. However, considering wave resistance according to Eq. (3.47) and Eq. (3.48) is dependent on the same amplitude function as the wave elevation, it may be used for the purpose of verification.

5.7.1 Deep water

Fig. 5.17 compares the computed deep water wave resistance coefficients for a Wigley hull with those obtained by Lunde (1951), Landweber (1979) and F. Huang (2013). The results of Lunde (1951) and Landweber (1979) have been selected for comparison since they too are computed using linearised theory, whereas F. Huang (2013) applies the Neumann-Michell theory; a modification of the classical Neumann-Kelvin theory. The contribution to the total wave resistance from the transverse and divergent waves are given by the waves whose direction angle varies between 0° and 35°16′ and between 35°16′ and 90°, respectively. This definition is in accordance with the intervals of θ for transverse and divergent waves derived in Sec. 3.1.1.

The comparison with the linearised theory in Fig. 5.17b shows good agreement for all Froude numbers, thus verifying the implementation as correct. When compared to the far-field wave resistance coefficient of F. Huang (2013) in Fig. 5.17a, the procedure is found to capture the overall trends, but with a larger amplitude of the oscillations. These discrepancies are assumed to be a result of the underlying differences between the theories applied, with the linearised Michell theory being the most simplified.



(A) $B/L = 0.1$ (B) $B/L = 0.75/8$

Figure 5.17: Comparison of deep water wave resistance coefficient for the Wigley model with (A) the Neumann-Michell theory of F. Huang (2013) (extracted from graph) and (B) the linearised theory of Lunde (1951)(extracted from an extended graph in Faltinsen (2005)) and Landweber (1979) (Eq. (4.18)).

The computed resistance coefficient for the DTMB 5415 model is however far from the far-field results of F. Huang (2013), as is displayed in Fig. 5.18. Especially for $Fn_l <\approx 0.25$, the resistance curve becomes unsmooth and the behaviour unrealistic. For higher Froude numbers, the computed values are off by a factor of about 2.5, but otherwise following the expected curvature. The deviations are expected to be the result of *(i)* a too coarse mesh on the hull geometry and *(ii)* limitations of the Michell theory, although it has not been verified.

Firstly, a coarse mesh is not capable of correctly accounting for the complex geometry of the DTMB 5415 hull, such as the sonar dome and its effect on the wave resistance through cancellation. Consequently, this deficiency gives the largest errors for small Froude numbers where the transverse wave system dominates, and lesser errors where the divergent waves dominate. A mesh refinement study was not completed due to the requirement of equidistant spacing in x- and z-direction for the numerical procedure described in Sec. 4.1, leading to exceedingly large computational effort required for the finer meshes. A morphing mesh could have been applied, although it would have to be paired an alternative numerical procedure.

Secondly, the thin ship assumption of Michell is less fulfilled for the destroyer than for e.g. the Wigley model. As a consequence, the computed resistance is expected to deviate from the more complex Neumann-Michell theory of F. Huang (2013) even given a sufficiently fine mesh. In addition, during the derivation and following implementation of Eq. (3.45), it was assumed that the offsets vanish at both ends, i.e., no transom stern. This is not representative for the DTMB 5415, as can be seen in Fig. 4.6.

Lastly, the experimental results of Sharma (1969) for a parabolic model in deep water are compared with computations, where the x-axis is defined as $0.5/Fn_l^2$ to better space the values. The first-order theory of Michell shows fair agreement with the experiments for all Froude numbers, regardless of the thin ship theory not being satisfied at the draft of the parabolic hull.

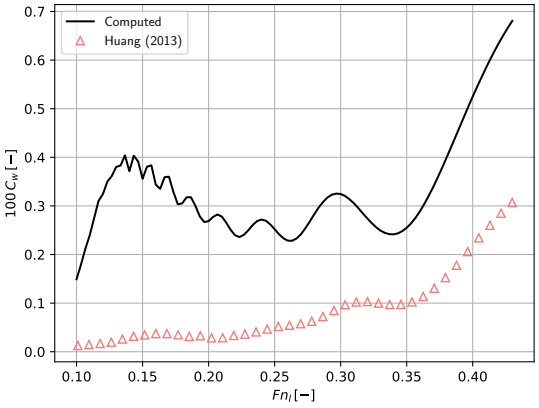


Figure 5.18: Comparison of deep water wave resistance coefficient for the DTMB 5415 model with the Neumann-Michell theory of F. Huang (2013) (extracted from graph).

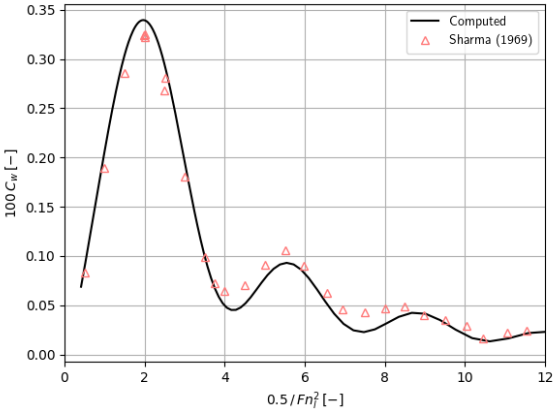


Figure 5.19: Comparison of deep water wave resistance coefficient for the parabolic model with the experiments by Sharma (1969).

5.7.2 Finite water

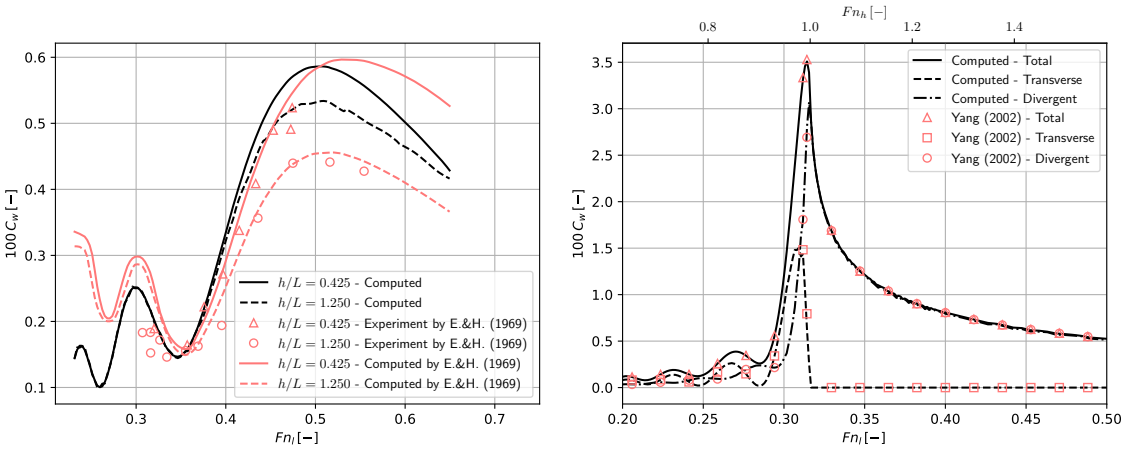
The computed finite water wave resistance coefficient is in Fig. 5.20 compared with the experiments and computations of Everest and Hogben (1969) as well as the linearised theory of Yang (2002). Note that Fig. 5.20a does not include an axis for the depth Froude number since the results of two different water depths are displayed.

As expected, the computational results for $h/L = 0.1$ are in accordance with the linear theory presented in Fig. 5.20b over the entire range of Froude numbers. At $Fn_h = 1.0$ the transverse wave system ceases to exist, with the only contribution to the wave resistance coming from the divergent waves. The peak of the wave resistance is at the critical depth Froude number.

Compared to the experimental results of Everest and Hogben (1969), in particular for $h/L = 0.425$, good agreement is found with the computed curve. For $h/L = 1.250$, despite lesser effects

of finite water depth from the sea bottom, the discrepancy between the computed curve is larger. This is partly due to inaccuracies affecting Eq. (3.46) as the water depth increases, which will be explained in the following paragraph. The computed curves of Everest and Hogben (1969) are in better agreement with their experimental values for both water depths because of their chosen computational method. Using measured waves, they fit an "equivalent generating source array", i.e., a source distribution which according to linear theory will generate the measured wave pattern. In the cases where linear theory appears valid, the source arrays may be used to compute the wave resistance. This empirical-mathematical approach is therefore not expected to exactly match Michell's formulation, despite both being based on linear theory.

Unlike for $h/L = 0.425$, the computed wave resistance for $h/L = 1.250$ is unsmooth over the entire range of Froude numbers, especially visible near $Fn_l = 0.52$. The reason for these discontinuities lies in the behaviour of the complex amplitude function in finite water for increasing water depth. The formal limit of Eq. (3.46) as h approaches infinity recover the deep water result presented in Eq. (3.45), i.e., $k \to k_0 \sec^2\theta$, $1 - k_0 h \operatorname{sech}^2 kh \to 1$ and $\cosh k(z+h)/\cosh kh \to \exp(-kz)$. However, the latter limit will in practice approach NaN since the individually computed values of $\cosh k(z+h)$ and $\cosh kh$ both returns infinity for large kh. Extra care must therefore be taken during the evaluation, e.g., using custom exceptions enforcing the correct limiting functions.



(A) $B/L = 0.1$ (B) $B/L = 0.1$, $h/L = 0.1$

Figure 5.20: Comparison of finite water wave resistance coefficient for the Wigley model with (A) experiments and computations by Everest and Hogben (1969) (extracted from graph) and (B) the linearised theory of Yang (2002) (Eq. (4.22)).

5.7.3 Deep and finite water comparison

A comparison between the wave resistance for a Wigley model in deep and finite water depth is given in Fig. 5.21. For small depth Froude numbers, the length of the transverse waves are small relative to the water depth and the resistance is hence not much affected by the sea bottom (Yang 2002). Starting from around $Fn_h = 0.6$ and peaking at the critical depth Froude number, the resistance is heavily affected by the finite water depth. This corresponds to the range where the difference in wave elevation is largest between the deep and finite water wave pattern. With further increasing Froude numbers the short-lengthened waves of the divergent wave system will begin to dominate the solution for both deep and finite water, consequently reducing the difference in wave resistance.

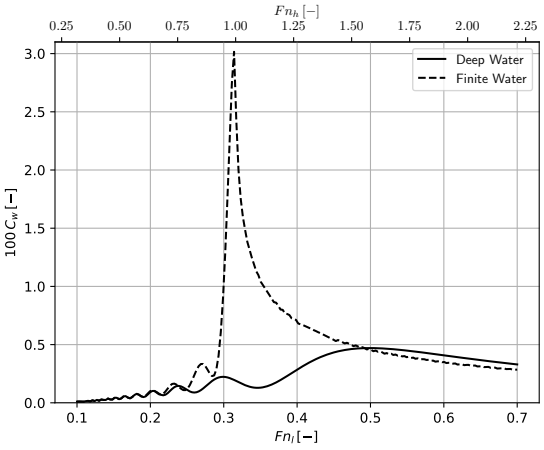


FIGURE 5.21: Comparison of deep and finite water wave resistance coefficient for the Wigley model described in Tab. 4.1 at $h/L = 0.1$. The depth Froude number is only applicable to finite water depth.

5.8 Wave cuts

There exist multiple sources in the literature where experimental or numerically computed wave cuts are presented, amongst others in the works of KAJITANI et al. (1983), LINDENMUTH et al. (1991), TUCK, SCULLEN et al. (1999b) and F. HUANG (2013). Several comparisons between the implemented approach and the literature were made, but due to large discrepancies none are presented.

It is concluded that the procedure presented in the thesis–as it stands–is not well suited for computing accurate wave cuts in the near-field of the vessel. E.g., KAJITANI et al. (1983) specifies that wave cuts should be measured for $x \in [-\frac{L}{2}, \frac{L}{2}]$ for a Wigley model, far closer to the vessel than the far-field region for which the procedure is developed for. The study of LINDENMUTH et al. (1991) extends the wave cut into the far-field, but considers the DTMB 5415 rather than the Wigley model. As discussed in Sec. 5.7, limitations in the representation of the destroyer hull will significantly affect both the near- and far-field solution, meaning that good agreement between the results are not expected.

5.9 Runtime performance

An essential part of the motivation for the thesis is to investigate the possibility of computing sufficiently accurate results using procedures that may be run in real-time. Tab. 5.1 presents the average time it takes to produce one of the wave patterns displayed in Fig. 5.1 using a *Nvidia GeForce RTX 3070 Ti*. The surface spans a range of 100 m × 54 m and consists of 1 080 000 triangular elements.

A significant reduction in the computational time is obtained when precomputing the points of stationary phase for finite water depth. This is due to the numerical procedures required to compute the finite water wave number and thus the points of stationary phase. However, since the points are dependent on the water depth, depth Froude number and angle α relative to the vessel, the required memory capacity increases significantly.

TABLE 5.1: The average computational time per frame in milliseconds and FPS for some selected wave patterns similar to those in Fig. 5.1 and Fig. 5.4. The computational time is not limited to the wave elevation only, but includes all processes run by Unity during one frame.

		[ms]	FPS
Infinite water depth		74	13.5
Finite water depth	$Fn_h \leq 1.0$	2224	0.45
Finite water depth*	$Fn_h \leq 1.0$	82	12.2
Finite water depth	$Fn_h > 1.0$	1044	0.96
Finite water depth*	$Fn_h > 1.0$	73	13.7
**when precomputing points of stationary phase*			

Despite the rapid runtime of the current implementation, there exist numerous potential improvement to be made. With regards to GPU-specific features in particular, the thesis has limited the scope to developing a working model, that is, a proof of concept, rather than dedicating effort into comprehensive optimisation of the code. That at most 14 FPS is reached is therefore not a concern, but rather a verification of the procedures potential.

Tab. 5.1 does also not consider the effect of turning and acceleration, wall reflection or multiple vessels, meaning that the runtime will increase beyond the approximate runtime provided here. Extensive performance tests has not been performed for these scenarios as they are very case-sensitive.

Chapter 6

Further work

Several issues for further work are already commented upon in the results and discussion chapter, both for the straight and circular course wave patterns. These will not be repeated in depth here.

Results for comparison As previously discussed, in particular with regards to the wave pattern for an accelerating vessel, there exist few comparable results in the existing literature. Similarly, except for the analytical approach of Stoker (1957), this can also be said for the wake from a vessel on a circular course. It is therefore recommended that further work aims to acquire wave pattern results for such cases, also containing relevant data including but not limited to: velocity, location, heading and depth throughout the experiment or computation. This will enable a full reconstruction in Unity using the procedure implemented in the thesis, thus facilitating an in-depth quantitative comparison.

Near-field The thesis only considers the far-field wave pattern, leading to large inaccuracies in the near-field. A separate near-field solution is for further work desired, e.g., following the approach proposed by Tuck, Scullen et al. (2000a).

Spray / foam White spray or foam is a non-linear phenomena arising in the near-field ship wake as the vessel moves along the surface at higher speeds, but is also found in the narrow turbulent wake following the vessel and near breaking ship waves. The striking visual aspects of the spray is easily observed, and hence increases its importance compared to many other non-linear components. Further work shall not necessarily aim to accurately model the behaviour of the spray/foam, but should investigate methods for imitating the phenomena. A criteria may here be the steepness of of the resulting wave pattern, indicating where waves are likely to break and thus cause white foam.

Transom stern In the method described for computing the amplitude function in Eq. (3.45), it is assumed that the vessel does not have a transom stern. This assumption simplifies the usage of the Filon quadrature presented in Sec. 3.4.2, but at the expense of accuracy. Further work should investigate the additional computational effort needed to account for the transom stern to decide if it's worth including. A procedure for accounting for the transom stern is outlined by amongst others Lazauskas (2009).

Current The presence of a uniform current will affect the wave pattern generated by the ship. Further work should investigate how large this effect is when considering typical current velocities

in the areas that are to be simulated. If it is sufficiently large, a routine should be developed to account for the effect using the already computed steady wave pattern for a ship on a straight course.

Viscosity The viscous correction factor presented in Eq. (3.34) is applied for both deep and finite water depth, although its validity in finite water depth is questionable. Further work should pursue a damping factor better grounded in the relevant literature.

Wall diffraction Interaction with walls does currently not consider diffraction, only reflection. Further work may investigate the consequences on the wave elevation by leaving out diffraction effects, or propose a procedure for including diffraction.

Chapter 7

Conclusion

Using Michell's thin ship theory as a foundation, a procedure has been developed to compute the far-field wave pattern for any number of vessels moving along any course in finite or infinite water depth, simultaneously accounting for vessel accelerations and wall reflections.

The highly oscillatory integral derived for the steady far-field wave pattern is solved using the method of stationary phase. To approximate the integrals obtained for the complex amplitude function as given by Michell's theory, a Filon quadrature method and a Filon-Trapezoidal method is applied.

To account for turning and acceleration of the vessel, the steady solution obtained for a straight path was adapted using the principles of Kelvin's method of stationary phase for a point source moving on the surface. The outlined regions of disturbance were translated and rotated using the instantaneous velocity and heading of the vessel at specified points along the travelled path.

The wave pattern for a Wigley hull has been presented for a straight and circular course, for an accelerating vessel, and for wall reflections, discussing benefits and drawbacks of the adaption from the steady-state solution. Although most cases lack the necessary data for an in-depth comparison, good qualitative agreement is observed with regards to the theory. In addition, deep and finite water wave resistance coefficients have been computed and verified against existing experimental and numerical results for both the Wigley hull and the DTMB 5415 destroyer hull.

The procedure is made parallelizable and implemented in the game engine Unity using the High-Level Shader Language. Thorough runtime performance tests have not been performed, but estimates performed for a mesh consisting of about 10^6 triangular elements indicate that the procedure is suitable for real-time simulation.

Further work is advised to acquire comparable results, either experimental or numerical, to validate the proposed approach for in particular turning and accelerating vessels. The procedure can also be extended to include the near-field solution, wall diffraction, the occurrence of spray and foam in the wake region and general improvements to the steady-state wave pattern.

Bibliography

Abramowitz, M. and Stegun, I. A. (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. 9th Dover printing, 10th GPO printing. Dover.

Bai, K. J. and McCarthy, J. (Jan. 1979). *Proceedings of the Workshop on Ship Wave-Resistance Computations Held at Bethesda, Maryland on 13-14 November 1979. Volume I.*

Chowdhury, S., Kv, A., S A, S. and Sundar, V. (May 2017). 'Nonlinear wave interaction with curved front seawalls'. In: *Ocean Engineering* 140, pp. 84–96. DOI: 10.1016/j.oceaneng.2017.05.015.

Daemrich, K.-F. and Kohlhase, S. (1978). 'Influence of Breakwater-Reflection on Diffraction'. In: *Coastal Engineering Proceedings* 1.16, pp. 651–663. DOI: 10.9753/icce.v16.36.

Everest, J. and Hogben, N. (1969). *An experimental study of the effect of beam variation and shallow water on "thin ship" wave predictions.*

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

Faltinsen, O. M. (2005). *Hydrodynamics of High-Speed Marine Vehicles*. Cambridge University Press, pp. 99–140. DOI: 10.1017/CBO9780511546068.

Filon, L. N. G. (1930). 'III. — On a Quadrature Formula for Trigonometric Integrals'. In: *Proceedings of the Royal Society of Edinburgh* 49, pp. 38–47. DOI: 10.1017/S0370164600026262.

Fosdick, L. D. (1968). 'A Special Case of the Filon Quadrature Formula'. In: *Mathematics of Computation* 22.101, pp. 77–81. DOI: 10.2307/2004764.

Froude, W. (1875). *The fundamental principles which govern the behaviour of fluids, with special reference to the resistance on ships*. Reprinted in *The papers of William Froude : M.A., LL.D., F.R.S. 1810-1879*. London: Institution of Naval Architects, 1955.

Froude, W. (1876). *The fundamental principles of the resistance of ships*. Reprinted in *The papers of William Froude : M.A., LL.D., F.R.S. 1810-1879*. London: Institution of Naval Architects, 1955.

Glassner, A. S. (2002). 'Duck!' In: *IEEE Computer Graphics and Applications* 22.4, pp. 88–97. DOI: 10.1109/MCG.2002.1016702.

Havelock, T. H. (1908). 'The Propagation of Groups of Waves in Dispersive Media, with Application to Waves on Water Produced by a Travelling Disturbance'. In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 81.549, pp. 398–430. Reprinted in *The collected papers of Sir Thomas Havelock on hydrodynamics*. 1965.

HAVELOCK, T. H. (1934). 'The Calculation of Wave Resistance'. In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 144.853, pp. 514–521. (Visited on 17th Apr. 2022). Reprinted in *The collected papers of Sir Thomas Havelock on hydrodynamics*. 1965.

HAVELOCK, T. H. (1965). *The collected papers of Sir Thomas Havelock on hydrodynamics*. Ed. by C. WIGLEY. Office of Naval Research, Dept. of the Navy.

HUANG, F. (2013). 'A Practical Computational Method for Steady Flow about a Ship'. PhD thesis. George Mason University. URL: https://hdl.handle.net/1920/8264.

HUANG, L., QU, Z., TAN, X., ZHANG, X., L. MICHELS, D. and JIANG, C. (2021). 'Ships, Splashes, and Waves on a Vast Ocean'. In: *ACM Transaction on Graphics* 40.6. DOI: 10.1145/3478513.3480495.

KAJITANI, H., MIYATA, H., IKEHATA, M., TANAKA, H., ADACHI, H., NAMIMATSU, M. and OGIWARA, S. (1983). *The Summary of the Cooperative Experiment on Wigley Parabolic Model in Japan*.

KHAN, R. S. (1994). 'A simple model of ship wakes'. University of British Columbia. DOI: 10.14288/1.0051431.

LAMB, H. (1932). *Hydrodynamics*. 6th ed. Cambridge: at the University Press.

LANDWEBER, L. (1979). 'Wigley Parabolic Hull Group Discussion'. In: *Proceedings of the Workshop on Ship Wave-Resistance Computations Held at Bethesda, Maryland on 13-14 November 1979. Volume I*, pp. 51–65.

LAZAUSKAS, L. V. (2009). 'Resistance, wave-making and wave-decay of thin ships, with emphasis on the effects of viscosity'. PhD thesis. University of Adelaide, School of Mathematical Sciences. URL: https://hdl.handle.net/2440/53216.

LINDENMUTH, W., RATCLIFFE, T. and REED, A. (1991). 'Comparative Accuracy of Numerical Kelvin Wake Code Predictions - 'Wake Off''. In: p. 275.

LORD KELVIN (1887a). 'On Ship Waves'. In: *Proceedings of the Institution of Mechanical Engineers* 38.1, pp. 409–434.

LORD KELVIN (1887b). 'On the waves produced by a single impulse in water of any depth, or in a dispersive medium'. In: *Philosophical Magazine* 5 (23), pp. 252–255.

LORD RAYLEIGH (1876). 'On Waves'. In: *Philosophical Magazine* 1, pp. 257–279.

LUNDE, J. K. (1951). *On the linearized theory of wave resistance for displacement ships in steady and accelerated motion*. Trondheim.

MICHELL, A. G. M. (1941). 'John Henry Michell. 1863-1940'. In: *Obituary Notices of Fellows of the Royal Society* 3.9, pp. 363–382. URL: http://www.jstor.org/stable/768895.

MICHELL, J. H. (1898). 'The Wave-Resistance of a Ship'. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 45.272, pp. 106–123. DOI: 10.1080/14786449808621111.

MICROSOFT (2021). *High-level Shader Language (HLSL)*. URL: https://docs.microsoft.com/en-us/windows/win32/direct3dhlsl/dx-graphics-hlsl.

MORÉ, J. J., GARBOW, B. S. and HILLSTROM, K. E. (Aug. 1980). *User Guide for MINPACK-1*. Tech. rep. ANL-80-74. Argonne, IL, USA: Argonne National Laboratory.

NAGERNYAK, V. (n.d.). *Stunning image of a boat, motor yacht racing through the river in sunlight.* URL: https://www.shutterstock.com/nb/image-photo/1736436425.

NEWMAN, J. N. (1977). *Marine hydrodynamics.* MIT Press.

NOBLESSE, F. (1983). 'A Slender-Ship Theory of Wave Resistance'. In: *Journal of Ship Research* 27.01, pp. 13–33. DOI: 10.5957/jsr.1983.27.1.13.

NOBLESSE, F. (2000). *Analytical Representation of Ship Waves.*

NOBLESSE, F., HUANG, F. and YANG, C. (2013). 'The Neumann–Michell theory of ship waves'. In: *Journal of Engineering Mathematics* 79. DOI: 10.1007/s10665-012-9568-7.

NVIDIA (2022). *The Benefits of Using GPUs.* URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

OLVER, S. (2008). 'Numerical Approximation of Highly Oscillatory Integrals'. PhD thesis. University of Cambridge. URL: https://www.maths.usyd.edu.au/u/olver/papers/OlverThesis.pdf.

PENNEY, W. G., PRICE, A. T., MARTIN, J. C., MOYCE, W. J., PENNEY, W. G., PRICE, A. T. and THORNHILL, C. K. (1952). 'Part I. The diffraction theory of sea waves and the shelter afforded by breakwaters'. In: *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 244, pp. 236–253. DOI: 10.1098/rsta.1952.0003.

PLESSET, M. S. and WHIPPLE, C. G. (1974). 'Viscous effects in Rayleigh-Taylor Instability'. In: *Physics of Fluids* 1.

RAMSEY, A. S. (1935). *A Treatise On Hydromechanics - Part II.* G. Bell and Sons Ltd. Reprinted in 1960.

SHARMA, S. D. (1969). 'Some Results Concerning the Wavemaking of a Thin Ship'. In: *Society of Naval Architects and Marine Engineers* 13 (1), pp. 72–81.

SIMMAN (2008). *Workshop on Verification and Validation of Ship Manoeuvring Simulation Methods - US Navy Combatant, DTMB 5415.* URL: https://www.simman2008.dk.

SIMMAN (2014). *Workshop on Verification and Validation of Ship Manoeuvring Simulation Methods - US Navy Combatant, DTMB 5415.* URL: https://simman2014.dk.

SOMMERFELD, A. (1896). 'Mathematische Theorie der Diffraction'. In: *Mathematische Annalen* 47, pp. 1432–1807. DOI: 10.1007/BF01447273.

SRETENSKI, L. N. (1946). 'Ship waves for circular courses (in Russian)'. In: *Bulletin de l'Académie des Sciences de l'URSS.*

STOKER, J. (1957). *Water Waves: The Mathematical Theory with Applications.* John Wiley & Sons, Inc. Reprinted in 1992. DOI: 10.1002/9781118033159.

SWIERKOWSKI, L., GOUTHAS, E., CHRISTIE, C. and WILLIAMS, O. (2009). 'Boat, wake and wave real-time simulation'. In: *Proceedings of SPIE - The International Society for Optical Engineering.* DOI: 10.1117/12.818430.

TUCK, E. O. (1967). 'A Simple "Filon-Trapezoidal" Rule'. In: *Mathematics of Computation* 21.98, pp. 239–241. DOI: 10.2307/2004168.

TUCK, E. O. (1974). 'The Effect of a Surface Layer of Viscous Fluid on the Wave Resistance of a Thin Ship'. In: *Journal of Ship Research* 18.04, pp. 265–271. DOI: 10.5957/jsr.1974.18.4.265.

Tuck, E. O. (1989). 'The wave resistance formula of J.H. Michell (1898) and Its significance to recent research in ship hydrodynamics'. In: *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics* 30.4, pp. 365–377. DOI: 10.1017/S0334270000006329.

Tuck, E. O., Scullen, D. C. and Lazauskas, L. (1999a). *Sea Wave Pattern Evaluation Part 1 report: Primary Code and Test Results (Surface Vessels).*

Tuck, E. O., Scullen, D. C. and Lazauskas, L. (1999b). *Sea Wave Pattern Evaluation Part 2 Report : Investigation of Accuracy.*

Tuck, E. O., Scullen, D. C. and Lazauskas, L. (2000a). *Sea Wave Pattern Evaluation Part 3 Report: Near-Field Waves.*

Tuck, E. O., Scullen, D. C. and Lazauskas, L. (2000b). *Sea Wave Pattern Evaluation Part 4 Report : Extension to Multihulls and Finite Depth.*

Tuck, E. O., Scullen, D. C. and Lazauskas, L. (2001a). *Sea Wave Pattern Evaluation Part 5 Report : Speed-up and Squat.*

Tuck, E. O., Scullen, D. C. and Lazauskas, L. (2001b). 'Ship-Wave Patterns in the Spirit of Michell'. In: *IUTAM Symposium on Free Surface Flows*. Springer Netherlands, pp. 311–318. DOI: 10.1007/978-94-010-0796-2_38.

Tuck, E. O., Scullen, D. C. and Lazauskas, L. (2002). *Sea Wave Pattern Evaluation Part 6 Report: Viscosity Factors.*

Vasstein, K. and Skarshaug, T. (Dec. 2021). *Gemini*. URL: https://github.com/Gemini-team/Gemini.

Wehausen, J. and Laitone, E. (1960). 'Surface Waves'. In: *Handbuch der Physik IX*. Ed. by S. Flugge and C. Truesdell, pp. 446–778.

Yang, Q. (2002). 'Wash and Wave Resistance of Ships in Finite Water Depth'. PhD thesis. Norwegian University of Science and Technology.

Yang, Q., Faltinsen, O. and Zhao, R. (2001). 'Wash and Wave Resistance of Ships in Finite Water Depth'. In: *Practical Design of Ships and Other Floating Structures*. Ed. by Y.-S. Wu, W.-C. Cui and G.-J. Zhou. Oxford: Elsevier Science Ltd, pp. 475–483. DOI: 10.1016/B978-008043950-1/50060-0.

Yuksel, C. (2010). 'Real-Time Water Waves with Wave Particles'. PhD thesis. Texas A&M University.

Yuksel, C., House, D. H. and Keyser, J. (2007). 'Wave Particles'. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)* 26.3. DOI: 10.1145/1276377.1276501.

Zeabuz (Apr. 2021). URL: https://zeabuz.com.

Appendix A

Viscous correction factor

Since the viscous correction factor of Tuck, Scullen et al. (2002) in Eq. (3.34) is inspired by the works of Lamb (1932, p. 624), a brief derivation of his contribution will be given first.

Let the two-dimensional velocity potential be defined as

$$\phi = \frac{\alpha g}{\omega} e^{kz} \cos(\omega t - kx), \tag{A.1}$$

where α is the wave amplitude. It follows that

$$u = \frac{\partial \phi}{\partial x} = \alpha \omega e^{kz} \sin(\omega t - kx), \tag{A.2}$$

$$w = \frac{\partial \phi}{\partial z} = \alpha \omega e^{kz} \cos(\omega t - kx), \tag{A.3}$$

$$\zeta = -\frac{1}{g} \frac{\partial \phi}{\partial t}\Big|_{z=0} = \alpha \sin(\omega t - kx). \tag{A.4}$$

For the described motion to persist in the presence of viscosity, the necessary surface forces applied must equal

$$\begin{aligned} p_{zz} &= -p + 2\mu \frac{\partial w}{\partial z}\Big|_{z=0} = -p + 2\mu\alpha\omega k \cos(\omega t - kx). \\ p_{zx} &= \mu \left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) \Big|_{z=0} = 2\mu\alpha\omega k \sin(\omega t - kx). \end{aligned} \tag{A.5}$$

where the relevant derivation of the stresses in a fluid in motion may be found in Lamb (1932, pp. 571–574) or Ramsey (1935, pp. 364–370). The rate at which the surface forces does work is

$$p_{zz} w + p_{zx} u = -p\alpha\omega \cos(\omega t - kx) + 2\mu\alpha^2\omega^2 k, \tag{A.6}$$

further reduced to $2\mu\alpha^2\omega^2 k$ when evaluating the mean value. Lamb (1932) argues that "*this must evidently be the rate at which energy is being dissipated in the free motion given by Eq. (A.2) and Eq. (A.3)*". It can be shown that the total energy of a wave per unity length is half kinetic, half potential, and expressed as $\frac{1}{2}\rho g\alpha^2$. Consequently, in the absence of the surface forces in Eq. (A.5),

$$\frac{d}{dt}\left(\frac{1}{2}\rho g \alpha^2 \right) = -2\mu\alpha^2\omega^2 k, \tag{A.7}$$

which simplifies to the ODE

$$\frac{d\alpha}{dt} = -2\nu k^2 \alpha. \tag{A.8}$$

A solution to Eq. (A.8) is

$$\alpha = \alpha_0 \exp(-2\nu k^2 t), \tag{A.9}$$

where $\exp(-2\nu k^2 t)$ represents the viscous correction factor. Alternatively, if directly transformed to the reference system of a ship moving along a straight path with constant speed in infinite water depth, the damping factor reads

$$\exp\left(-2\nu k_0^2 \frac{x}{U} \sec^4\theta\right). \tag{A.10}$$

The absence of a y-dependency in Eq. (A.10) is with regards to ship waves described as arbitrary by Tuck, Scullen et al. (2002). They also question the use of the group velocity rather than the phase velocity, which would yield a factor of four in the exponent.

An improved correction factor is by Tuck, Scullen et al. (2002) introduced through the addition of a dissipative term in the steady free-surface condition. For a reference system moving to the left at constant velocity U, it reads

$$g\phi_z + U^2\phi_{xx} + 4\nu U\phi_{xzz} = 0. \tag{A.11}$$

In accordance with the wave system described in Eq. (3.3), the deep water velocity potential holds the form

$$\exp(ik(x\cos\theta + y\sin\theta) + kz), \tag{A.12}$$

which when applied to Eq. (A.11) yields

$$gk - U^2k^2\cos^2\theta + 4\nu iUk^3\cos\theta = 0. \tag{A.13}$$

If approximating the solution for small ν using the exact solution for $\nu = 0$,

$$k = k_0\sec^2\theta + \frac{4i\nu}{U}k_0^2\sec^5\theta, \tag{A.14}$$

and the following damping factor becomes

$$\exp\left(-\frac{4\nu}{U}k_0^2\sec^4\theta(x + y\tan\theta)\right). \tag{A.15}$$

A similar derivation can be made using the finite water velocity potential

$$\frac{\cosh k(z+h)}{\cosh kh}\exp(ik(x\cos\theta + y\sin\theta)) \tag{A.16}$$

rather than Eq. (A.12), yielding the damping factor presented in Eq. (3.34). However, this approach neglects multiple key considerations related to viscous damping in finite water depth, as further discussed in Sec. 5.5.

Appendix B

Code structure

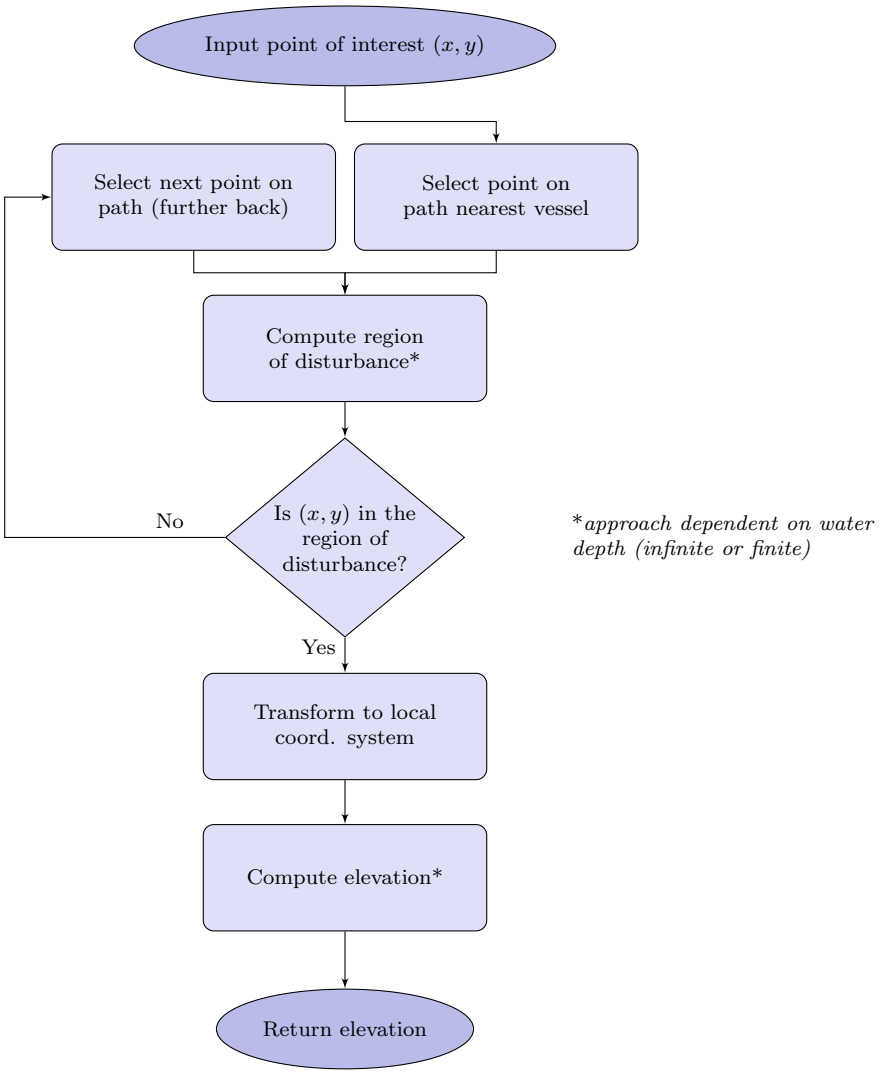


FIGURE B.1: Flow diagram indicating the flow structure for computing wave elevation.

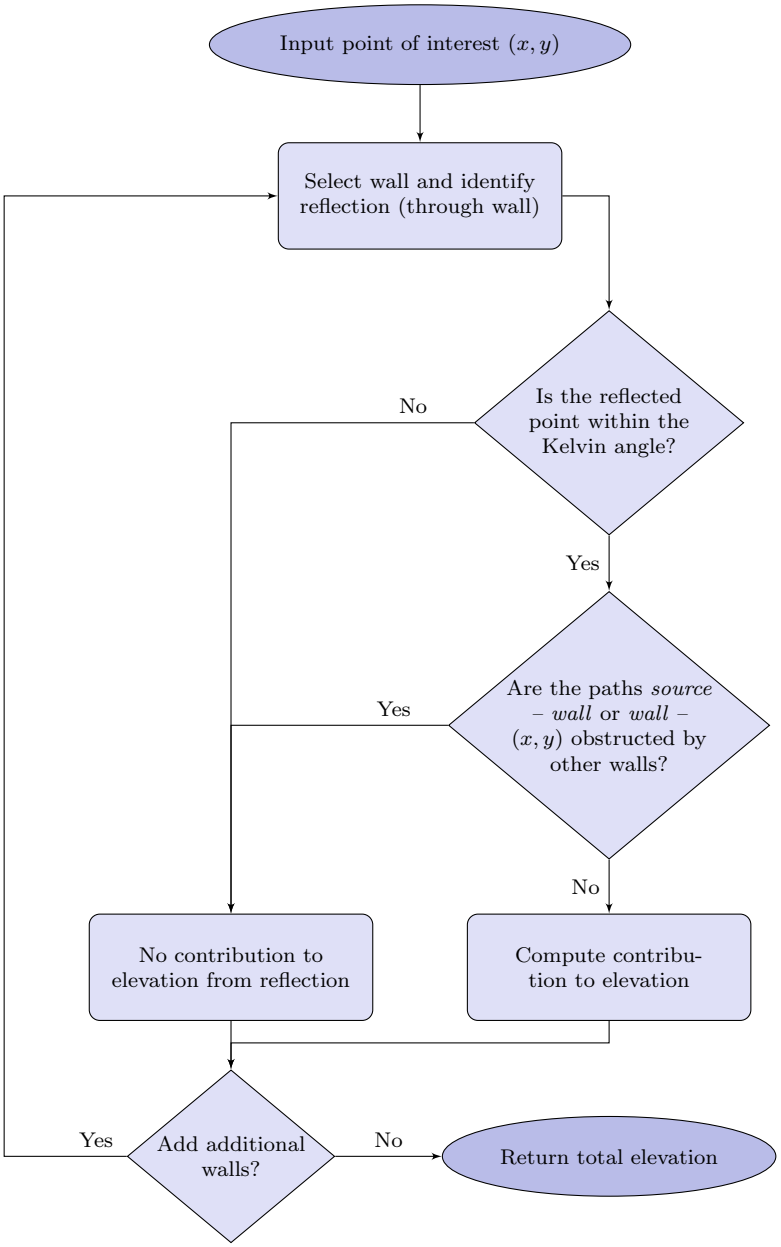
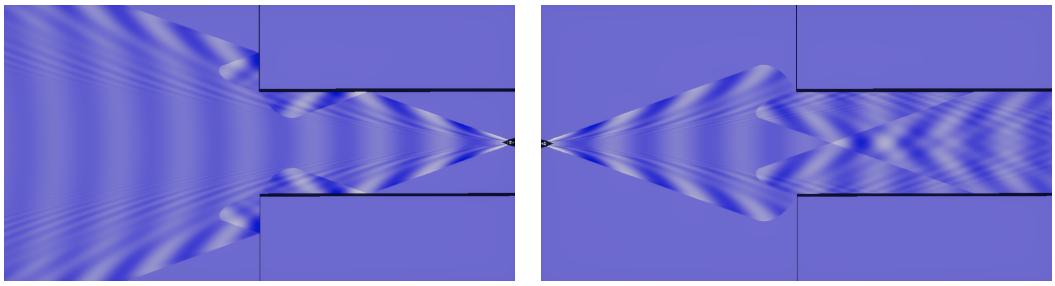


FIGURE B.2: Flow diagram indicating code structure for including wall reflections.

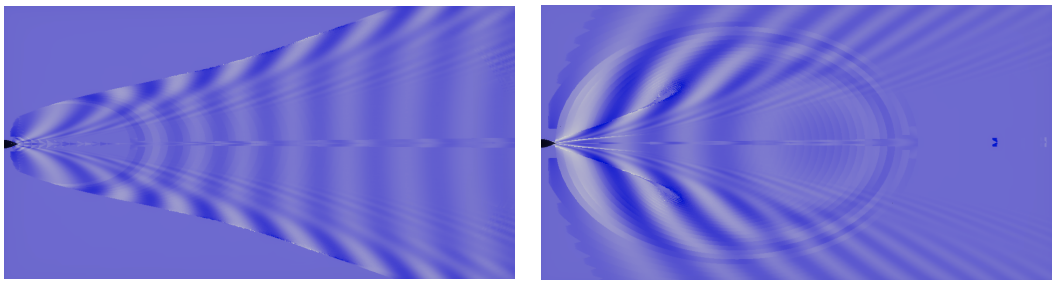
Appendix C

Ship waves for an arbitrary course



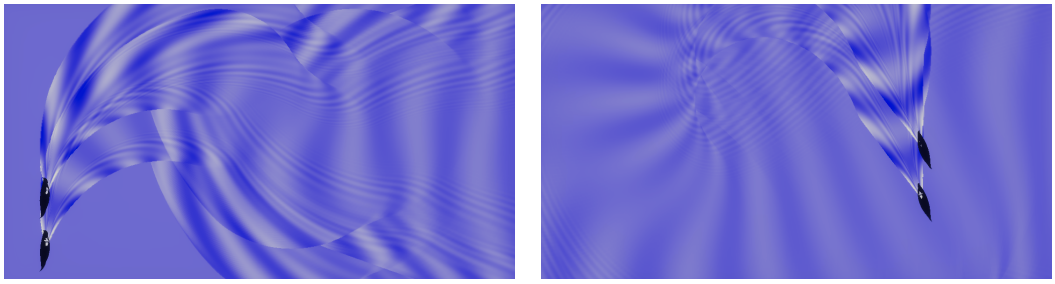
(A) (B)

FIGURE C.1: Ship waves for a canal entry and exit at $Fn_l = 0.5$ in infinite water depth.



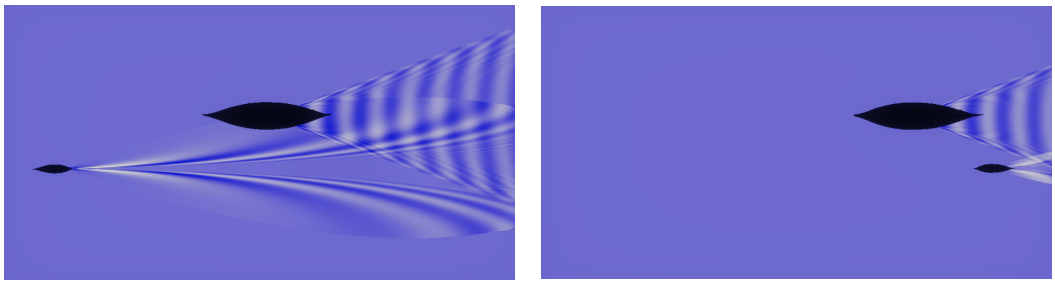
(A) (B)

FIGURE C.2: For a vessel moving from left to right at $Fn_l = 0.5$, (A) gives the ship waves as $Fn_h = 0.4 \rightarrow 1.2$ and (B) gives the ship waves as $Fn_h = 1.2 \rightarrow 0.4$, where both transitions are linear with regards to the depth Froude number. The Wigley model is used, with the entire length of the image covering 100m. The rings observed for both cases shows how the distance between the path points affects the solution through their disks of disturbance. (B) also gives the appearance of a sudden forming of an additional Kelvin angle, where the wave energy created as $Fn_h \approx 1.0$ propagates ahead of it due to the high group velocity near the critical depth.



(A) (B)

FIGURE C.3: Two vessels sailing nearby each other at $Fn_l = 0.5$ in infinite water depth.



(A) t_0 (B) $t_0 - t$

FIGURE C.4: A vessel accelerating past another vessel three times its size. The larger vessel moves at $Fn_l = 0.2$, and both in infinite water depth.

Appendix D

Code

The complete Unity project with all relevant code is located at https://github.com/JonArntK/ShipWaves. However, note that the project does not include a user interface nor tutorial, as the scope of the thesis is limited to a proof of concept. The HLSL implementations relevant to the theory and method described are for convenience included in the appendix.

```
#pragma kernel ComputeWaterSurfaceKernel

#include "ComputeElevationGlobal.hlsl"
#include "GenerateMesh.hlsl"
#include "VesselGeometryStruct.hlsl"
#include "VesselPathStruct.hlsl"
#include "WallReflection.hlsl"

#define GROUP_SIZE_X 64
#define GROUP_SIZE_Y 1
#define GROUP_SIZE_Z 1

// Define buffers for storing the vertices, normals and texture coordinates of the mesh.
RWByteAddressBuffer _VertexBuffer;
RWByteAddressBuffer _NormalBuffer;
RWByteAddressBuffer _TexcoordBuffer;

// Mesh properties.
int _QuadCount, _XQuadCount, _ZQuadCount, _XOrigin, _ZOrigin;
float _Time, _XStep, _ZStep;

// Number of vessels.
int _NumVessels;

// Vessel geometry properties.
StructuredBuffer<float3> _VesselCoord;
int2 _VesselNxNy;

// Vessel path properties.
StructuredBuffer<float2> _VesselPathCoord;
StructuredBuffer<float> _VesselPathTime, _VesselPathHeading, _VesselPathDepth;
int _VesselPathNumPoints;

// Stationary points for use in computation of ship waves in finite water depth.
// Pre-calculated to increase performance.
StructuredBuffer<float2> _FiniteWaterStationaryPoints;
float4 _FnhInfo, _HInfo, _AlphaInfo;

// Wall properties
StructuredBuffer<float4> _Walls;
int _NumWalls;

float ComputeElevation(Vertex v, VesselGeometryStruct vgs, VesselPathStruct vps)
```

```
{
    // Compute wave elevation.

    float elevation = 0.0;  // Initialize wave elevation.
    for (int i = 0; i < _NumVessels; i++)   // For each vessel.
    {
        elevation += ComputeShipWaveElevationGlobal(v.position.xz, i, vgs, vps, false, _Walls, 0, _NumWalls);   // Add contribution before reflection from walls.
        elevation += ComputeWallReflection(v.position.xz, i, vgs, vps, _Walls, _NumWalls);  // Add contribution after reflection from walls.
    }
    // Return total elevation.
    return elevation;
}


[numthreads(GROUP_SIZE_X, GROUP_SIZE_Y, GROUP_SIZE_Z)]
void ComputeWaterSurfaceKernel(uint3 threadID : SV_DispatchThreadID)
{
    // Initialize structs.
    VesselGeometryStruct vesselGeometryInfo = InitializeVesselGeometry(_VesselCoord, _VesselNxNy);
    VesselPathStruct vesselPathInfo = InitializeVesselPath(_VesselPathNumPoints, _VesselPathCoord, _VesselPathTime, _VesselPathHeading, _VesselPathDepth,
        _FiniteWaterStationaryPoints, _FnhInfo, _HInfo, _AlphaInfo);

    uint id = threadID.x + threadID.y * GROUP_SIZE_X;
    if ((int) id >= _QuadCount)
        return;
    uint idx1 = id * 6;
    uint idx2 = id * 6 + 1;
    uint idx3 = id * 6 + 2;
    uint idx4 = id * 6 + 3;
    uint idx5 = id * 6 + 4;
    uint idx6 = id * 6 + 5;

    // Generate quad.
    Vertex v1 = GenerateQuad(idx1, _XQuadCount, _XStep, _ZStep, _XOrigin, _ZOrigin);
    Vertex v2 = GenerateQuad(idx2, _XQuadCount, _XStep, _ZStep, _XOrigin, _ZOrigin);
    Vertex v3 = GenerateQuad(idx3, _XQuadCount, _XStep, _ZStep, _XOrigin, _ZOrigin);
    Vertex v6 = GenerateQuad(idx6, _XQuadCount, _XStep, _ZStep, _XOrigin, _ZOrigin);

    // Compute surface elevation.
    v1.position.y = ComputeElevation(v1, vesselGeometryInfo, vesselPathInfo);
    v2.position.y = ComputeElevation(v2, vesselGeometryInfo, vesselPathInfo);
    v3.position.y = ComputeElevation(v3, vesselGeometryInfo, vesselPathInfo);
    v6.position.y = ComputeElevation(v6, vesselGeometryInfo, vesselPathInfo);

    // ----- Triangular mesh from quads -----
    // Quads are generated from the 'id' in the order displayed below.
    //    p3, p5        p6
    //       ------------
    //      |\          |
    //      |  \        |
    //      |    \      |
    //      |      \    |
    //      |        \  |
    //      |__________\|
    //     p1           p2, p4
    //
    // Hence triangles are made as (p1, p2, p3) and (p4, p5, p6),
    // which is equivalent to      (p1, p2, p3) and (p2, p3, p6),
    // avoiding the computation of p4 and p5.

    StoreTriangle(idx1, idx2, idx3, v1, v3, v2, _VertexBuffer, _NormalBuffer, _TexcoordBuffer);
    StoreTriangle(idx4, idx5, idx6, v2, v3, v6, _VertexBuffer, _NormalBuffer, _TexcoordBuffer);
}
```

```
#ifndef __COMPUTEELEVATIONGLOBAL_HLSL__
#define __COMPUTEELEVATIONGLOBAL_HLSL__

#include "ComputeElevationGlobalDeepWaterFunctions.hlsl"
#include "ComputeElevationGlobalFiniteWaterFunctions.hlsl"
#include "ComputeElevationLocalDeepWater.hlsl"
#include "ComputeElevationLocalFiniteWater.hlsl"
#include "VesselGeometryStruct.hlsl"
#include "VesselPathStruct.hlsl"
#include "WallReflectionFunctions.hlsl"


float GetVelocity(VesselPathStruct vps, int index)
{
    // U = dS / dt, i.e., distance divided by time.
    float U = sqrt(pow(vps.coord[index].x - vps.coord[index + 1].x, 2) + pow(vps.coord[index].y - vps.coord[index
    ↪  + 1].y, 2))
        / abs(vps.time[index] - vps.time[index + 1]);
    return U;
}

float2 GlobalToLocalCoord(float2 XZ, float t, VesselPathStruct vps, int index, float U)
{
    // Find point P.
    float XP = vps.coord[index].x, ZP = vps.coord[index].y, tP = vps.time[index], heading = vps.heading[index];

    // Transform point P from the global coordinate system to the local.
    float2 rotatedCoord = RotationMatrix(XZ.x, XZ.y, -heading + PI, XP, ZP); //
    float XRotated = rotatedCoord.x, ZRotated = rotatedCoord.y;

    float x = (t - tP) * U + (XRotated - XP);
    float z = ZRotated - ZP;

    // Return the local coordinate equivalent of (X, Z).
    return float2(x, z);
}

bool IsPointValid(float X, float Z, float U, float t, float fnh, VesselPathStruct vps, float index,
    bool isWallReflection, StructuredBuffer<float4> walls, int currentWall, int numWalls)
{
    // A regular point is valid if it is:
        // - within the region of disturbance, which is dependent on water depth.
        // - is not obstructed by any wall wrt. to the source point.
    // A wall reflection point is valid if it is:
        // - within the region of disturbance, which is dependent on water depth.
        // - special consideration for obstructions by walls.

    if (!IsFiniteWater(fnh) &&
        IsPointInRegionDeepWater(X, Z, vps.coord[index].x, vps.coord[index].y, U, t, vps.time[index],
        ↪  vps.heading[index]) &&
        !IsPointObstructedByWall(float2(X, Z), float2(vps.coord[index].x, vps.coord[index].y), walls, currentWall,
        ↪  numWalls, isWallReflection))
    {
        return true;
    }
    else if (IsFiniteWater(fnh) &&
        IsPointInRegionFiniteWater(X, Z, vps.coord[index].x, vps.coord[index].y, U, t, vps.time[index],
        ↪  vps.heading[index], fnh) &&
        !IsPointObstructedByWall(float2(X, Z), float2(vps.coord[index].x, vps.coord[index].y), walls, currentWall,
        ↪  numWalls, isWallReflection))
    {
        return true;
    }

    return false;
}


```

```
64  float ComputeShipWaveElevationGlobal(float2 XZ, int vesselNum, VesselGeometryStruct vgs, VesselPathStruct vps,
65      bool isWallReflection, StructuredBuffer<float4> walls, int currentWall, int numWalls)
66  {
67      // Define number of points in the vessel path.
68      int vesselPathNumPoints = vps.numPoints;
69
70      // Define starting index for current vessel: vesselNum.
71      int vesselPathIndexStart = vesselNum * vesselPathNumPoints;
72
73      // Initialize values.
74      float X0, Z0, R, U, fnh;
75      int index = 0;      // Used to store the index of the correct path point when found.
76      bool computeElevationFlag = false;
77
78      // Get vesselPathTime from the struct.
79      float t = vps.time[vesselPathIndexStart + vesselPathNumPoints - 1];
80
81      // Loop over all points in the vessel path, starting with the point closest to the vessel (placed at the back
        ↪  of the array).
82      for (int i = vesselPathNumPoints - 2; i >= 0; i--)
83      {
84          int refIndex = vesselPathIndexStart + i;
85
86          // Compute velocity based on the current and previous path locations and the time difference between them.
87          U = GetVelocity(vps, refIndex);
88
89          // Compute the depth Froude number for the current vessel path point (is dependent on U and h).
90          fnh = Fnh(U, vps.depth[refIndex]);
91
92          // Check if point is valid, i.e., if it is inside the region on disturbance and not obstructed by any
            ↪  walls w.r.t. the source point.
93          if (IsPointValid(XZ.x, XZ.y, U, t, fnh, vps, refIndex, isWallReflection, walls, currentWall, numWalls))
94          {
95              if (!isWallReflection ||
96                  (isWallReflection && isWallReflectionValid(XZ, vps.coord[refIndex].xy, walls[currentWall])))
97              {
98                  index = i;
99                  computeElevationFlag = true;
100                 break;
101             }
102         }
103     }
104
105     // Initalize elevation 'y' as 0;
106     float y = 0;
107
108     if (computeElevationFlag)   // If a point is found, meaning that the elevation should be computed.
109     {
110         // Get local coordinate equivalent to (X, Z).
111         float2 xz = GlobalToLocalCoord(XZ, t, vps, vesselPathIndexStart + index, U);
112
113         // Compute the elevation using the local coordinate system.
114         if (IsFiniteWater(fnh))
115         {
116             float depth = vps.depth[vesselPathIndexStart + index];
117             y = ComputeShipWaveElevationLocalFiniteWater(xz.x, xz.y, vesselNum, vgs, U, depth, vps);
118         }
119         else
120         {
121             y = ComputeShipWaveElevationLocalDeepWater(xz.x, xz.y, vesselNum, vgs, U);
122         }
123     }
124
125     return y;
126 }
127
128 #endif // __COMPUTEELEVATIONGLOBAL_HLSL__
```

```hlsl
#ifndef __COMPUTEELEVATIONGLOBALDEEPWATERFUNCTIONS_HLSL__
#define __COMPUTEELEVATIONGLOBALDEEPWATERFUNCTIONS_HLSL__

// Check if a point (x, z) is inside a circle with radius r and origo at (x0, z0).
bool IsPointInCircle(float x, float z, float x0, float z0, float r)
{

    return pow(x - x0, 2) + pow(z - z0, 2) < pow(r, 2);
}

float3 GetCircleGlobalDeepWater(float XP, float ZP, float U, float t, float tP, float heading)
{
    float dt = t - tP; // Time difference from when at point P and now.

    float r = 0.25 * U * dt; // Circle radius, equal to half the group velocity with theta = 0.

    float X0 = XP + r * cos(heading); // Center of circle in global coordinate system, x-component.
    float Z0 = ZP + r * sin(heading); // Center of circle in global coordinate system, z-component.

    return float3(X0, Z0, r);
}

bool IsPointInRegionDeepWater(float X, float Z, float XP, float ZP, float U, float t, float tP, float heading)
{
    // Get circular disk of disturbance for deep water in global coordinate system.
    float3 globalCircle = GetCircleGlobalDeepWater(XP, ZP, U, t, tP, heading);
    float X0 = globalCircle.x, Z0 = globalCircle.y, R = globalCircle.z;

    // Check if point is inside disk of disturbance.
    if (IsPointInCircle(X, Z, X0, Z0, R))
    {
        return true;
    }
    return false;
}

#endif // __COMPUTEELEVATIONGLOBALDEEPWATERFUNCTIONS_HLSL__
```

```hlsl
#ifndef __COMPUTEELEVATIONGLOBALFINITEWATERFUNCTIONS_HLSL__
#define __COMPUTEELEVATIONGLOBALFINITEWATERFUNCTIONS_HLSL__

#include "HLSLMath.hlsl"

// Compute depth Froude number.
float Fnh(float U, float h)
{
    return U / sqrt(g * h);
}

// Check if the depth satisfies criteria for finite water.
bool IsFiniteWater(float fnh)
{
    if (fnh > 0.4)
    {
        return true;
    }
    return false;
}

// Check if a point (x, z) is inside an ellipse with width 2a, height 2b, rotated at 'angle' radians with center
↪  at (x0, z0).
bool IsPointInEllipse(float x, float z, float x0, float z0, float a, float b, float angle)
{
    return pow((cos(angle) * (x - x0) + sin(angle) * (z - z0)), 2.0) / pow(a, 2.0) +
           pow((sin(angle) * (x - x0) - cos(angle) * (z - z0)), 2.0) / pow(b, 2.0) <= 1.0;
}

```

```
29  float2 GetEllipseWidthHeight(float fnh, float rDeepWater)
30  {
31      // The ellipse width and height is defined relative to the radius of the corresponding circle for deep water.
32      // The values in 'aa' and 'bb' are fitted to match the shape of the ellipse for Fnh in range [0.4, 1.0] and
        ↪  [1.0, 1.3].
33      //
34      // This simplification reduces the necessary computational effort signficantly.
35
36      float a, b; // Initialize.
37      if (fnh <= 1.0)  // Valid for Fnh in [0.4, 1.0].
38      {
39          float aa[5] = { -5.68386249, 17.51752173, -15.7408244, 5.60288223, 0.30324627 };
40          float bb[5] = { -0.84655427, 6.64047585, -9.57806052, 4.99030651, 0.12666546 };
41          a = aa[0] * pow(fnh, 4.0) + aa[1] * pow(fnh, 3.0) + aa[2] * pow(fnh, 2.0) + aa[3] * fnh + aa[4];
42          b = bb[0] * pow(fnh, 4.0) + bb[1] * pow(fnh, 3.0) + bb[2] * pow(fnh, 2.0) + bb[3] * fnh + bb[4];
43      }
44      else {  // Valid for Fnh in [1.0, 1.3].
45          float aa[6] = { 1.20982757e+04, -3.42756971e+03, 3.61976109e+02, 2.35844458e+01, 4.82020395e+00,
            ↪  2.09440310e-02 };
46          float bb[6] = { -6.99986025e+01, 5.49934890e+02, -1.55589753e+02, 2.43251754e+01, 1.00362626e+00,
            ↪  4.76700487e-03 };
47
48          float a0 = 2.0189910997505467;
49          float b0 = 1.3237123674048987;
50
51          fnh -= 1.0; // Was an assumption used when creating the polygons approximating 'a' and 'b' below.
52
53          a = aa[0] * pow(fnh, 5.0) + aa[1] * pow(fnh, 4.0) + aa[2] * pow(fnh, 3.0) + aa[3] * pow(fnh, 2.0) + aa[4]
            ↪  * fnh + aa[5] + a0;
54          b = bb[0] * pow(fnh, 5.0) + bb[1] * pow(fnh, 4.0) + bb[2] * pow(fnh, 3.0) + bb[3] * pow(fnh, 2.0) + bb[4]
            ↪  * fnh + bb[5] + b0;
55      }
56      return float2(a * rDeepWater, b * rDeepWater);
57  }
58
59  float4 GetEllipseGlobalFiniteWater(float XP, float ZP, float U, float t, float tP, float heading, float fnh)
60  {
61      float dt = t - tP; // Time difference from when at point P and now.
62
63      float rDeepWater = 0.25 * U * dt; // Circle radius for deep water, equal to half the group velocity with theta
        ↪  = 0.
64
65      float2 ellipseWidthHeight = GetEllipseWidthHeight(fnh, rDeepWater);
66
67      float X0 = XP + ellipseWidthHeight.x * cos(heading); // Center of circle in global coordinate system,
        ↪  x-component.
68      float Z0 = ZP + ellipseWidthHeight.x * sin(heading); // Center of circle in global coordinate system,
        ↪  z-component.
69
70      return float4(X0, Z0, ellipseWidthHeight.x, ellipseWidthHeight.y);
71  }
72
73  bool IsPointInRegionFiniteWater(float X, float Z, float XP, float ZP, float U, float t, float tP, float heading,
    ↪  float fnh)
74  {
75      // Get ellipse-shaped disk of disturbance for finite water in global coordinate system.
76      float4 globalEllipse = GetEllipseGlobalFiniteWater(XP, ZP, U, t, tP, heading, fnh);
77
78      float X0 = globalEllipse.x, Z0 = globalEllipse.y;
79      float ellipseWidth = globalEllipse.z, ellipseHeight = globalEllipse.w;
80
81      if (fnh <= 1.0 && IsPointInEllipse(X, Z, X0, Z0, ellipseWidth, ellipseHeight, heading))  // For Fnh <= 1.0,
        ↪  the ellipse-shaped region
82          // of disturbance covers the whole ellipse.
83      {
84          return true;
85      }
86      else if (fnh >= 1.0 && IsPointInEllipse(X, Z, X0, Z0, ellipseWidth, ellipseHeight, heading)) // For Fnh > 1.0,
        ↪  a minimum angle theta
```

```
87          // defines the part of the ellipse which holds the region of disturbance (ref. group velocity for Fnh >
            ↪  1.0).
88      {
89          float thetaMark = atan2(Z - ZP, X - XP);
90          float theta = heading - thetaMark;
91          theta = Mod(theta, 2.0 * PI);   // Ensure that we are operating with a value below 2pi.
92          float thetaMin = acos(1.0 / fnh);
93          if (abs(theta) > thetaMin && abs(theta) < 2.0 * PI - thetaMin)
94              return true;
95      }
96      return false;
97  }
98
99  #endif // __COMPUTEELEVATIONGLOBALFINITEWATERFUNCTIONS_HLSL__
```

```
1   #ifndef __COMPUTEELEVATIONLOCALDEEPWATER_HLSL__
2   #define __COMPUTEELEVATIONLOCALDEEPWATER_HLSL__
3
4   #include "HLSLMath.hlsl"
5   #include "VesselGeometryStruct.hlsl"
6
7   #define KELVIN_ANGLE 0.3398369095
8
9   float2 ComplexAmplitudeFunctionDeepWater(int vesselNum, VesselGeometryStruct vgs, float theta, float U)
10  {
11      // Compute the deep water complex amplitude function.
12
13      int vesselNx = vgs.vesselNxNy[0];
14      int vesselNy = vgs.vesselNxNy[1];
15
16      int vesselCoordIndexStart = vesselNum * vesselNx * vesselNy;    // To ensure the correct vessel geometry
        ↪  coordinates are used.
17
18      float dx = vgs.coord[vesselCoordIndexStart + vesselNy].x - vgs.coord[vesselCoordIndexStart + 0].x;
19      float dy = abs(vgs.coord[vesselCoordIndexStart + 1].y - vgs.coord[vesselCoordIndexStart + 0].y);
20
21      float P = 0.0;
22      float Q = 0.0;
23
24      float k0 = g / pow(U, 2.0);
25
26      float K = k0 / pow(cos(theta), 2.0) * dy;
27      float omega0 = (exp(K) - 1.0 - K) / pow(K, 2.0);
28      float omegaNy = (exp(-K) - 1.0 + K) / pow(K, 2.0);
29      float omegaJ = (exp(K) + exp(-K) - 2.0) / pow(K, 2.0);
30
31      float K2 = k0 / cos(theta) * dx;
32      float omegaEven = (3.0 * K2 + K2 * cos(2.0 * K2) - 2.0 * sin(2.0 * K2)) / pow(K2, 3.0);
33      float omegaOdd = 4.0 * (sin(K2) - K2 * cos(K2)) / pow(K2, 3.0);
34
35      for (int i = 0; i < vesselNx; i++)
36      {
37          int refIndex = vesselCoordIndexStart + i * vesselNy;
38
39          float F = 0.0;
40          F += omega0 * vgs.coord[refIndex + vesselNy - 1].z * exp(k0 * vgs.coord[refIndex + vesselNy - 1].y /
            ↪  pow(cos(theta), 2.0)) * dy;
41          F += omegaNy * vgs.coord[refIndex + 0].z * exp(k0 * vgs.coord[refIndex + 0].y / pow(cos(theta), 2.0)) *
            ↪  dy;
42          for (int j = 1; j < vesselNy - 1; j++)
43          {
44              F += omegaJ * vgs.coord[refIndex + j].z * exp(k0 * vgs.coord[refIndex + j].y / pow(cos(theta), 2.0)) *
                ↪  dy;
45          }
46
47          if (Mod(i, 2) == 0.0)
48          {
49              P += omegaEven * F * cos(k0 * vgs.coord[refIndex].x / cos(theta)) * dx;
```

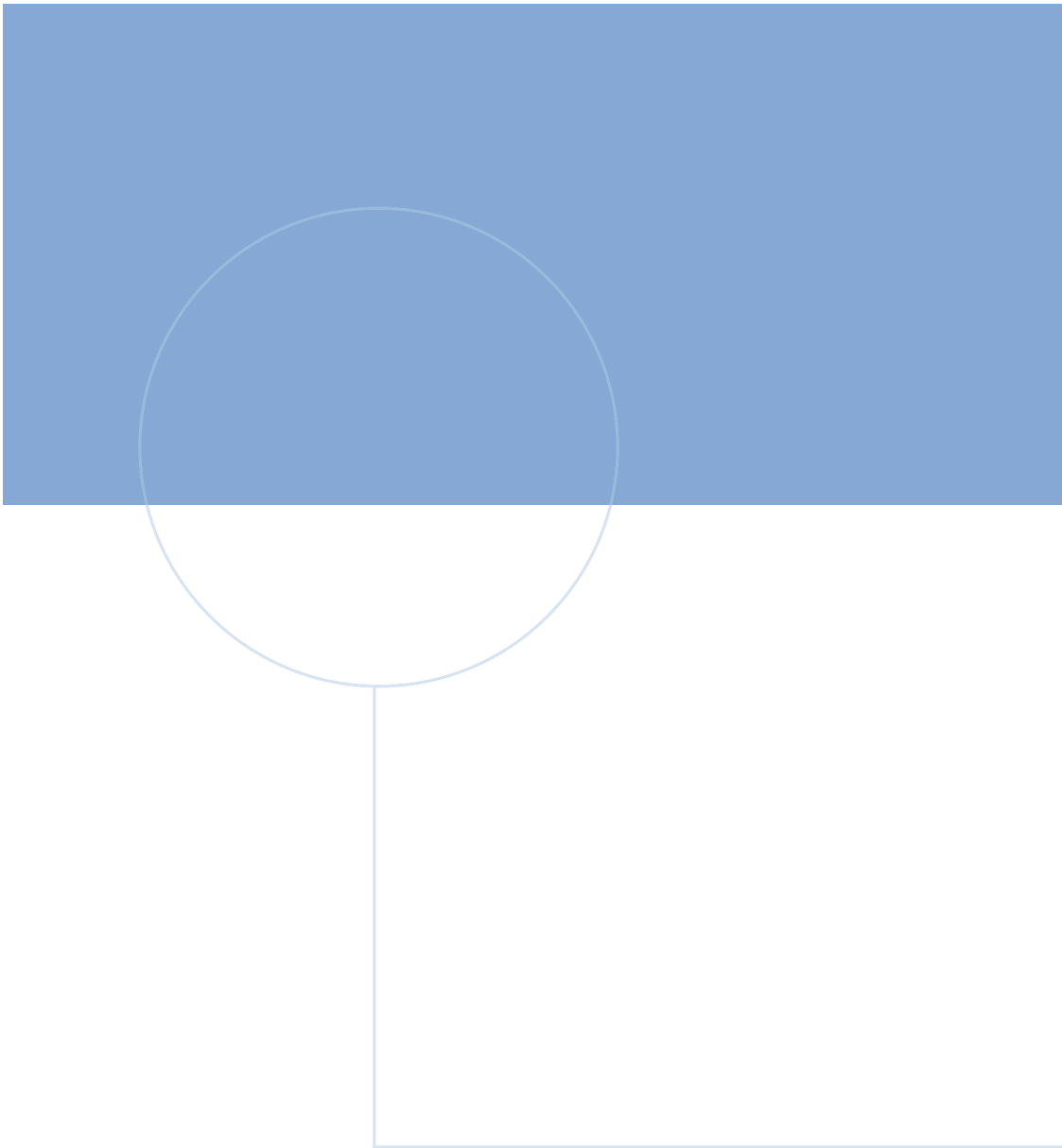
```
            Q += omegaEven * F * sin(k0 * vgs.coord[refIndex].x / cos(theta)) * dx;
        }
        else
        {
            P += omegaOdd * F * cos(k0 * vgs.coord[refIndex].x / cos(theta)) * dx;
            Q += omegaOdd * F * sin(k0 * vgs.coord[refIndex].x / cos(theta)) * dx;
        }
    }

    float2 amp = float2(0.0, -2.0 / PI * pow(k0, 2.0) / pow(cos(theta), 4.0));  // The amplitude is imaginary.
    return c_mul(amp, float2(P, Q));
}

float ComputeShipWaveElevationLocalDeepWater(float x, float z, int vesselNum, VesselGeometryStruct vgs, float U)
{
    float k0 = g / pow(U, 2.0);

    // Compute polar coordinate equivalent to (x, z).
    float r = sqrt(pow(x, 2.0) + pow(z, 2.0));
    float alpha = atan2(z, x);
    alpha = abs(alpha);     // Solution is symmetric about the x-axis.

    // If alpha is above the Kelvin half angle, the wave elevation is zero.
    float deltaBoundary = 0.01;        // To avoid singularities at boundary equal to Kelvin angle.

    if (alpha >= KELVIN_ANGLE - deltaBoundary)     // In deep water, no elevation is assumed outside the Kelvin
    ↪  angle.
    {
        alpha = KELVIN_ANGLE - deltaBoundary;
    }

    // For the method of stationary phase, dG/dtheta gives two solutions within the interval [-PI/2, PI/2] for
    ↪  theta.
    float2 theta = float2(alpha / 2.0 - 0.5 * asin(3.0 * sin(alpha)),
                          -PI / 2.0 + alpha / 2.0 + 0.5 * asin(3.0 * sin(alpha)));


    // Each theta has its own amplitude (transverse and divergent wave amplitude).
    float2 A1 = ComplexAmplitudeFunctionDeepWater(vesselNum, vgs, theta.x, U); // float2 since complex ->
    ↪  float2(real, imaginary)
    float2 A2 = ComplexAmplitudeFunctionDeepWater(vesselNum, vgs, theta.y, U); // float2 since complex ->
    ↪  float2(real, imaginary)

    // Check if amplitude is nan (not a number) or inf (infinite). If so, set as zero.
    if (isnan(abs(A1.x)) || isnan(abs(A1.y)) || isinf(abs(A1.x)) || isinf(abs(A1.y)))
    {
        A1 = float2(0.0, 0.0);
    }
    if (isnan(abs(A2.x)) || isnan(abs(A2.y)) || isinf(abs(A2.x)) || isinf(abs(A2.y)))
    {
        A2 = float2(0.0, 0.0);
    }

    // Compute wave elevation.
    float amp = sqrt(2.0 * PI / k0 / r) * pow(abs(1.0 - 9.0 * pow(sin(alpha), 2.0)), -0.25);

    A1.x *= pow(abs(cos(theta.x)), 3.0 / 2.0);
    A1.y *= pow(abs(cos(theta.x)), 3.0 / 2.0);
    float2 temp1 = c_mul(A1, c_exp(float2(0.0, k0 * r * cos(theta.x - alpha) / pow(cos(theta.x), 2.0) + PI /
    ↪  4.0)));

    A2.x *= pow(abs(cos(theta.y)), 3.0 / 2.0);
    A2.y *= pow(abs(cos(theta.y)), 3.0 / 2.0);
    float2 temp2 = c_mul(A2, c_exp(float2(0.0, k0 * r * cos(theta.y - alpha) / pow(cos(theta.y), 2.0) - PI /
    ↪  4.0)));

    // Include viscous correction factor.
    float nu = 0.0002;
    float v1 = exp(-4.0 * pow(k0, 2.0) / U * nu / pow(cos(theta.x), 4.0) * (x + z * tan(theta.x)));
```

```
113     float v2 = exp(-4.0 * pow(k0, 2.0) / U * nu / pow(cos(theta.y), 4.0) * (x + z * tan(theta.y)));
114
115
116     float zeta = amp * (v1 * temp1.x + v2 * temp2.x);     // Want the real part of the elevation.
117     return zeta;
118 }
119
120 #endif // __COMPUTEELEVATIONLOCALDEEPWATER_HLSL__
```

```
1   #ifndef __COMPUTEELEVATIONLOCALFINITEWATER_HLSL__
2   #define __COMPUTEELEVATIONLOCALFINITEWATER_HLSL__
3
4   #include "ComputeElevationLocalDeepWater.hlsl"
5   #include "ComputeElevationGlobalFiniteWaterFunctions.hlsl"
6   #include "FiniteWaterDispersionRelation.hlsl"
7   #include "HLSLMath.hlsl"
8   #include "StationaryPhaseFiniteWater.hlsl"
9   #include "VesselGeometryStruct.hlsl"
10
11
12
13  float2 ComplexAmplitudeFunctionFiniteWater(int vesselNum, VesselGeometryStruct vgs, float theta, float U, float
    ↪  fnh)
14  {
15      // Computes the complex amplitude function for a vessel on a straight path in finite water depth.
16
17      float h = (float) pow(U / fnh, 2.0) / g;
18
19      int vesselNx = vgs.vesselNxNy[0];
20      int vesselNy = vgs.vesselNxNy[1];
21
22      int vesselCoordIndexStart = vesselNum * vesselNx * vesselNy;
23
24      float dx = vgs.coord[vesselCoordIndexStart + vesselNy].x - vgs.coord[vesselCoordIndexStart + 0].x;
25      float dy = abs(vgs.coord[vesselCoordIndexStart + 1].y - vgs.coord[vesselCoordIndexStart + 0].y);
26
27      float k = FiniteWaterDispersionRelation(fnh, h, theta);
28      float k0 = g / (float) pow(U, 2.0);
29
30      float P = 0.0;
31      float Q = 0.0;
32
33      float K = k * dy;
34      float omega0 = ((float) exp(K) - (float) 1.0 - K) / (float) pow(K, 2.0);
35      float omegaNy = ((float) exp(-K) - (float) 1.0 + K) / (float) pow(K, 2.0);
36      float omegaJ = ((float) exp(K) + (float) exp(-K) - 2.0) / (float) pow(K, 2.0);
37
38      float K2 = k * (float) cos(theta) * dx;
39      float omegaEven = (3.0 * K2 + K2 * (float) cos(2.0 * K2) - 2.0 * (float) sin(2.0 * K2)) / (float) pow(K2,
        ↪  3.0);
40      float omegaOdd = 4.0 * ((float) sin(K2) - K2 * (float) cos(K2)) / (float) pow(K2, 3.0);
41
42      for (int i = 0; i < vesselNx; i++)
43      {
44          int refIndex = vesselCoordIndexStart + i * vesselNy;
45
46          float F = 0.0;
47          F += omega0 * vgs.coord[refIndex + vesselNy - 1].z * (float) cosh(k * (vgs.coord[refIndex + vesselNy -
            ↪  1].y + h)) / (float) cosh(k * h) * dy;
48          F += omegaNy * vgs.coord[refIndex + 0].z * (float) cosh(k * (vgs.coord[refIndex + 0].y + h)) / cosh(k * h)
            ↪  * dy;
49          for (int j = 1; j < vesselNy - 1; j++)
50          {
51              F += omegaJ * vgs.coord[refIndex + j].z * (float) cosh(k * (vgs.coord[refIndex + j].y + h)) / cosh(k *
                ↪  h) * dy;
52          }
53
54          if (Mod(i, 2) == 0.0)
```

```
55          {
56              P += omegaEven * F * (float) cos(k * vgs.coord[refIndex].x * cos(theta)) * dx;
57              Q += omegaEven * F * (float) sin(k * vgs.coord[refIndex].x * cos(theta)) * dx;
58          }
59          else
60          {
61              P += omegaOdd * F * (float) cos(k * vgs.coord[refIndex].x * cos(theta)) * dx;
62              Q += omegaOdd * F * (float) sin(k * vgs.coord[refIndex].x * cos(theta)) * dx;
63          }
64      }
65
66      float2 amp = float2(0.0, -2.0 / PI * (float) pow(k, 2.0) / (1.0 - k0 * h * (float) pow(1 / cos(theta), 2.0) *
        ↪  (float) pow(1.0 / cosh(k * h), 2.0)));
67      return c_mul(amp, float2(P, Q));
68  }
69  float ComputeShipWaveElevationLocalFiniteWater(float x, float z, int vesselNum, VesselGeometryStruct vgs, float U,
    ↪  float h, VesselPathStruct vps)
70  {
71      float fnh = Fnh(U, h);
72
73      // Compute polar coordinate equivalent to (x, z).
74      float r = sqrt(pow(x, 2.0) + pow(z, 2.0));
75      float alpha = atan2(z, x);
76      alpha = abs(alpha); // Solution is symmetric about the x-axis.
77
78      // If alpha is above the Kelvin half angle, the wave elevation is zero.
79      float deltaBoundary = 0.02; // To avoid singularities at boundary.
80
81      if (alpha >= PI * 0.5 - deltaBoundary)     // No elevation is present ahead of the vessel.
82      {
83          return float(0.0);
84      }
85
86      float2 theta = GetPointsOfStationaryPhaseFiniteWaterBuffer(vps, fnh, h, alpha); //
87      //float2 theta = GetPointsOfStationaryPhaseFiniteWater(float2(-PI * 0.5 + 0.02, 0.00), fnh, h, alpha); //
        ↪  Select this option to use precomputed values for the points of stationary phase.
88
89
90      // Each theta has its own amplitude (transverse and divergent wave amplitude). Then compute wave elevation.
91      float2 A1 = float2(0.0, 0.0), temp1 = float2(0.0, 0.0);
92      if (fnh < 1.0)  // Only compute amplitude for the transverse waves when at subcritical depth Froude number.
93      {
94          A1 = ComplexAmplitudeFunctionFiniteWater(vesselNum, vgs, theta.x, U, fnh); // float2 since complex ->
            ↪  float2(real, imaginary)
95          A1.x *= (float) sqrt(2.0 * PI / (r * abs(ddG(theta.x, fnh, h, alpha))));
96          A1.y *= (float) sqrt(2.0 * PI / (r * abs(ddG(theta.x, fnh, h, alpha))));
97          temp1 = c_mul(A1, c_exp(float2(0.0, r * G(theta.x, fnh, h, alpha) + PI / 4.0)));
98      }
99      float2 A2 = ComplexAmplitudeFunctionFiniteWater(vesselNum, vgs, theta.y, U, fnh); // float2 since complex ->
        ↪  float2(real, imaginary)
100     A2.x *= (float) sqrt(2.0 * PI / (r * abs(ddG(theta.y, fnh, h, alpha))));
101     A2.y *= (float) sqrt(2.0 * PI / (r * abs(ddG(theta.y, fnh, h, alpha))));
102     float2 temp2 = c_mul(A2, c_exp(float2(0.0, r * G(theta.y, fnh, h, alpha) - PI / 4.0)));
103
104     // Check for singularities or NaN and remove if present.
105     if (isnan(temp1.x) || isinf(temp1.x))
106         temp1.x = 0.0;
107     else if ((isnan(temp2.x) || isinf(temp2.x)))
108         temp2.x = 1.0;
109
110     // Include viscous correction factor.
111     float nu = 0.0002;
112     float k1 = FiniteWaterDispersionRelation(fnh, h, theta.x);
113     float v1 = exp(-4.0 * nu * U * pow(k1, 3) * cos(theta.x) * (x * cos(theta.x) + abs(z) * sin(theta.x)));
114
115     float k2 = FiniteWaterDispersionRelation(fnh, h, theta.y);
116     float v2 = exp(-4.0 * nu * U * pow(k2, 3) * cos(theta.y) * (x * cos(theta.y) + abs(z) * sin(theta.y)));
117
118
```

```
119     float zeta = v1 * temp1.x + v2 * temp2.x; // Want the real part of the elevation.
120     return zeta;
121 }
122
123 #endif // __COMPUTEELEVATIONLOCALFINITEWATER_HLSL__
```



NTNU

Norwegian University of
Science and Technology