FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Ocean Surface Generation and Rendering

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Thomas Gamper
Matrikelnummer 0107543

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer

Wien, 28.08.2018

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Ocean Surface Generation and Rendering

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Thomas Gamper
Registration Number 0107543

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer

Vienna, 28.08.2018

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Thomas Gamper
Lehmanngasse 25/2/12, 1230 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

I would like to thank the following people:

Winfried Auzinger for the discussions about the Fourier Transform.

Andrea Weidlich for her steady support and tons of tea.

Johannes Scharl and Peter Houska for their insights regarding the projected grid.

Alexander Kusternig for allowing me to use his mature 3D model toolchain.

All members of the *CG Club* for their moral and technical support.

The *Faculty of Computer Science* for granting me a generous scholarship for this thesis.

And finally, my family, who made this possible.

Abstract

The synthesis of a believable depiction of the ocean surface is a permanent topic of interest in computer graphics. It represents even more of a challenge for applications which require the real-time display of a water body as large as the ocean. That is because the ocean is a highly dynamic system which combines waves at all scales, ranging from millimetres to kilometres. Moreover, the ocean may be observed from several distances, ranging from close-ups to views which reach the horizon. Thus, we present a framework to generate and render the open ocean in real time, for arbitrary viewing distances and including waves at all scales. We focus our efforts on the geometry of the animated ocean surface, for which we leverage a set of wave spectrum models from oceanographic research. We discuss the intricacies of said models, as well as their fitness for real-time rendering. Moreover, we delineate in detail how to integrate distinct wave spectrum models into a coherent framework, from which one is able obtain believable, consistent and coherent results.

Kurzfassung

Die realistische Darstellung von Wasseroberflächen ist seit jeher ein beliebtes Forschungsthema in der Computergraphik. Vor allem das Rendern des weiten Ozeans in Echtzeit stellt eine besondere Herausforderung dar. Die Gründe hierfür sind vielseitig. Unter anderem ist der Ozean ein riesiges Gewässer, welches wir auf sehr unterschiedliche Art und Weise abbilden möchten, von der detaillierten Nahaufnahme bis zum weitläufigen Panorama. Zusätzlich stellt der Ozean ein sehr komplexes, dynamisches System dar, das aus einer riesigen Anzahl Wellen unterschiedlicher Größe besteht, vom riesigen Tsunami bis zur millimeterkleinen Kapillarwelle.

Wir präsentieren in dieser Arbeit Algorithmen um den weiten Ozean nicht nur in Echtzeit zu rendern, sondern auch alle dafür benötigten Daten in Echtzeit zu generieren. Unser Fokus liegt darauf, glaubhafte Ozeanoberflächen mittels Wellenspektren zu erzeugen, wobei letztere dem Forschungsbereich der Ozeanographie entspringen. Wir untersuchen unterschiedliche Wellenspektren sowohl auf ihre Eigenheiten, als auch auf ihre Eignung für die Darstellung mittels Echtzeitgraphik-Algorithmen. Zusätzlich erläutern wir die notwendigen Hintergründe um unterschiedlich Wellenspektren sinnvoll in eine gemeinsames Grundstruktur zu integrieren. Selbige ermöglicht es uns schlussendlich, unabhängig vom gewählten Wellenspektrum, nicht nur glaubhafte, sondern auch stimmige Ozeanoberflächen zu erzeugen.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Natural phenomena are a challenging topic in the field of computer graphics. Actual examples are complex structures which are to be found throughout nature, such as mountains, trees and water. The latter is especially interesting because of its highly dynamic form, which poses a variety of sophisticated problems. For computer graphics to reproduce the diverse appearance of ocean surfaces represents one such problem. Figures 1.1, 1.2 and 1.3 may give the reader an impression of the wide visual range of the ocean. Covering all the dynamics as well as the lighting of the entire ocean to reproduce such scenery would exceed the scope of this work. We may approach a more specific subject, namely the synthesis of animated ocean surfaces which are both believable and computationally feasible. Fortunately, the oceanographic research community already developed models which satisfy the better part of those requirements, as they are essential for oceanographic simulations. Still, to integrate the mathematics of said models into a coherent framework for ocean surface synthesis may pose a significant challenge. Moreover, only a representation of the oceanographic models most adequate for computer graphics algorithms may allow us to reduce computational complexity to an acceptable level, and therefore to complete the task at hand.

1.2 Problem Statement

Rendering an ocean is a demanding task for several reasons. First, consider the sheer size of a water body as large as an ocean, which in numerous viewing situations will be visible all the way to the horizon. Second, the ocean surface is dynamic, therefore it needs to be constantly updated with the passage of time even though the wave interactions that define its shape are huge in terms of complexity. Third, the optics of water are intricate. Incoming light may be reflected at the surface or may be refracted into the water body, where the ratio between both is dependent on the angle of incidence between the incoming light and the surface normal at the

Figure 1.1: Smith, Helen (Photographer). (2013, June 4). A stunningly blue and calm Arctic reflection of sea and sky divided by distant bright white ice and interrupted by ripples created by the ship [digital image]. Retrieved from Smith [2012].

point of incidence. Some of the refracted light may find its way back to the ocean's surface, either by scattering inside the water body or by reflection at the sea bottom, or possibly by a combination of both. Moreover, waves on the ocean surface may break and cause surf and foam, both of which strongly deviate in appearance from the surrounding water surface.

1.3 Scope and Focus of the Work

The scope of this thesis includes the generation, animation and rendering of the surface of an open ocean in real time. We focus our interest on the synthesis of animated ocean surface geometry, for which we will adopt a set of models from oceanographic research. Specific properties of said models allow for easy addition and reduction of detail, as well as for a range of algorithmic optimizations. The former combined with the latter gives us the opportunity to strike a well-adjusted balance between model detail and computational workload, and thereby to improve upon the status quo of current implementations. The works by Bruneton et al. [2010], Eric Bruneton [2010], and Dupuy and Bruneton [2012] represent such implementations, and because the latter two incorporate a real-time capable variant of the seminal *choppy wave algorithm* by Tessendorf [1999], we chose them as cornerstones for this thesis. With the foundation for our work at hand, we may outline the improvements we intend to make:

- Implement the constant-overhead, adaptive meshing scheme by Johanson [2004], as it has been tailored to large, animated water surfaces.
- Implement a set of distinct models from oceanographic research, and evaluate them with regard to plausibility of the obtained ocean surfaces, and with regard to suitability for the ocean lighting algorithm by Bruneton et al. [2010].

Figure 1.2: Karre, Julie (Photographer). (2013, August 7). One of the last sunsets for the first leg of the Oregon II [digital image]. Retrieved from Karre [2013].

- Extend the level-of-detail scheme by Eric Bruneton [2010] with a multi-resolution variant, and evaluate the potential performance gain.

Moreover, although not a contribution in and by itself, we will elaborate in detail on the theoretical background of the ocean surface synthesis algorithm and the respective performance optimizations. We deem this necessary, as some important aspects, such as the intricacies of the Fourier Transform, correct slope computation, and the surface tiling algorithm, may have been discussed only superficially in the original works by Eric Bruneton [2010], Tessendorf [1999], and Dupuy and Bruneton [2012].

1.4 Structure of the Work

The remainder of this work is organized as follows: Chapter 2 gives a survey of existing ocean simulation and rendering methods. Chapter 3 elaborates on the theoretical background the oceanographic models are based on, as well as on the models themselves. Chapter 4 describes in detail the synthesis of all data related to the ocean surface, including both the algorithmic optimizations and the level of detail mechanism. Furthermore, we give an overview of the rendering algorithms adopted for our implementation. Chapter 5 discusses the results of our work, as well as the improvements we were able to achieve in comparison to the state of the art. Last, Chapter 6 gives a summary of our work and suggests future improvements based on open issues of our implementation.

Figure 1.3: National Oceanic and Atmospheric Administration (Photographer). (1989, Winter). North Pacific storm waves as seen from the M/V NOBLE STAR [digital image]. Retrieved from NOAA [1989].

CHAPTER 2

Related Work

The ocean surface is an intricate phenomenon which owes its complexity to its highly dynamic nature. Be it a quiet sea or an agitated one, small turbulent waves or huge breaking ones, the underlying mechanisms are manifold and act on a variety of scales. Surface tension, wind, earth's gravity, the gravitational pull of sun and moon, all have an impact on ocean surface waves, where each force or effect governs a different range of wavelengths [Munk, 2010]. Scientists have spent centuries trying to understand and explain these mechanisms. Oceanographic researchers define the behaviour of an ocean surface not based on its influencing quantities, but based on its location [Young, 1999]. Water areas far from the coast are called *deep water*, water areas close to shore are called *shallow water*, with *intermediate* areas in between. Different water locations are governed by different parameters. Deep ocean surfaces are dominated by the interaction of wind and gravity at the interface between air and water, whereas shallow water surfaces are characterized by waves breaking near the shore [Kinsman, 2002]. Deep water may ignore changes in water depth, whereas intermediate and shallow waters need to account for how it affects surface waves. Additionally, tidal waves and tsunamis may travel unnoticed through deep water, but have a measurable impact on intermediate water areas and even more so on shallow water areas.

To date, computer graphics employs several ocean wave simulation models which may be roughly separated into three families: parametric description, spectral description, and computational fluid dynamics (CFD). The first describes the water surface by means of parametric equations which have been derived based on real-world observations [Biesel, 1952, Gerstner, 1809, Rankine, 1863]. The second family approximate the ocean surface using wave spectra which simulate the sea as a random process based on the distribution of wave energy in frequency space [Kinsman, 2002]. Third, computational fluid dynamics in combination with the Navier-Stokes equations (NSE) are potentially able to fully describe the dynamics of all kinds of fluid, including the ocean. We will omit any further discussion regarding CFD and NSE, because, one, those models are the ones with the largest computational cost and therefore the least adequate to simulate an entire ocean, and two, the low degree of control over the simulation these algorithms allow for may cause the user to be unable to get the desired results. The interested reader may consult Bridson [2015] and Ihmsen et al. [2014] though.

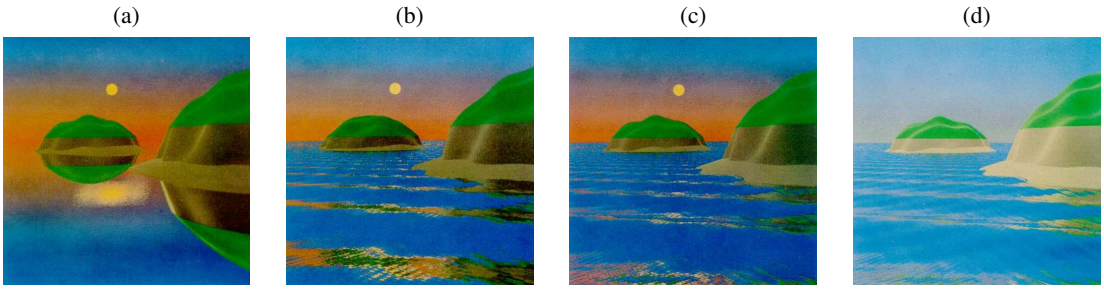


Figure 2.1: (a) Still water near sunset. (b) One reflection from waves. (c) Two reflections from waves. (d) Early afternoon, two reflections from waves. Source: Max [1981]

2.1 Parametric Models

Parametric models are generally bound to the spatial domain, where they generate and animate the ocean surface by means of a sum of periodical functions which evolve throughout time using a phase difference. One of the earliest works in this realm of computer graphics has been done by Max [1981]. The ocean surface is represented as a height map, where for each point (x, z) at time t the height is computed as a sum of sinusoids:

$$h(x, z, t) = y + \sum_{i=1}^{N} A_i \cos(l_i x + m_i z - \omega_i t) \tag{2.1}$$

where y is the mean height of the free surface, N is the number of waves, A_i is the amplitude of the ith wave, ω_i its angular frequency, and $\mathbf{k}_i = (l_i, m_i)$ its wavevector which defines the travelling direction of the wave. For the sum of waves to achieve a realistic shape, Max applies linear wave theory [Airy, 1845]: the waves on the water surface are assumed to be dominated by gravity, the wave amplitudes are assumed to be small in relation to the size of the water body, and the water body is assumed to have infinite depth. Then, according to linear wave theory, it follows that the wavenumber $k = \|\mathbf{k}\|$ and the angular frequency ω are related by $\omega^2 = kg$, where $g = 9.81$ denotes earth's gravity. Results obtained by Max are shown in Figure 2.1. Both Schachter [1980] and Perlin [1985] developed similar approaches, but instead of generating any actual geometry they simply distort the normal vectors of given surfaces.

The assumption of infinite depth restricts the above methods to deep water, they are unable to mimic the behaviour of waves as they approach the shore. The transition of waves into shallow water is the cause for phenomena such as the steepening and eventual breaking of waves, as well as for wave refraction [Mei, 1989]. The latter describes the tendency of wavefronts to align themselves parallel to the sloping beach regardless of their initial orientation. The underlying cause is the dependency of wave propagation speed on water depth, where waves move more slowly in shallow water. The part of a wave front which enters shallow water first is decelerated, the remainder of the wave still in deeper water will move faster, as a result the wave front will be turned parallel to the line of transition into shallow water.

Peachey [1986], still in the realm of linear wave theory, simulates waves near the shore through the relationship $\omega^2 = kg \tanh(kh)$, where h is the water depth. Waves in shallow water

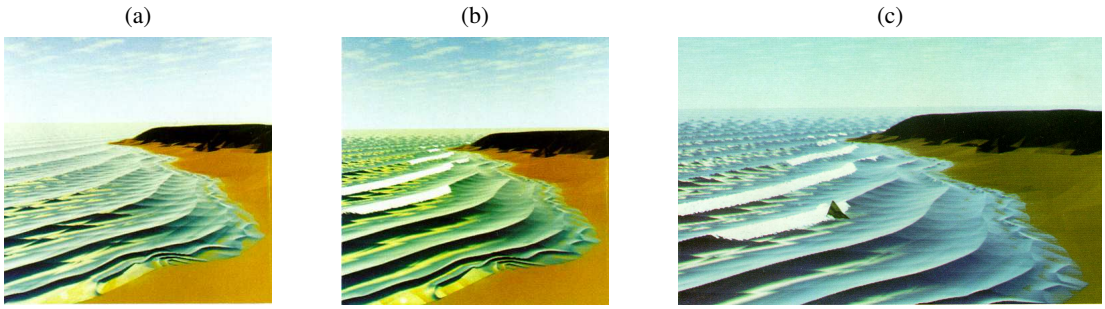


Figure 2.2: (a) Waves on the beach. (b) Breaking waves on the beach. (c) Breaking waves with obstacle. Source: Peachey [1986]

increase their steepness with declining water depth, Peachey therefore interpolates between a gentle sinusoidal wave form with long throughs and rounded crests, and short choppy waves with sharp crests. To recreate the appearance of waves near the breaking point, Peachey steepens the front of the crests of such waves even more, and stretches out the back of the crests, giving the wave profile an asymmetric form. Additionally, particle systems are added to generate spray wherever waves break or collide with partially submerged obstacles. Results of Peachey [1986] are shown in Figure 2.2.

Instead of using sinusoids, Fournier and Reeves [1986] build on Gerstner's work [Gerstner, 1809, Rankine, 1863] to synthesize waves. Compared to the gentle form of sinusoids in Max [1981], Gerstner waves allow for a larger variety of wave shapes, including waves with a sharp crest. Such waves are common near the shore and in stormy seas. Gerstner waves assume a water surface dominated by gravity and an incompressible water body of infinite depth. Then, a single Gerstner wave may be written as follows:

$$\mathbf{x} = \mathbf{x}_0 - \frac{\mathbf{k}}{\|\mathbf{k}\|} A \sin\left(\mathbf{k} \cdot \mathbf{x}_0 - \omega t\right) \tag{2.2}$$

$$y = y_0 - A \cos\left(\mathbf{k} \cdot \mathbf{x}_0 - \omega t\right) \tag{2.3}$$

where $\mathbf{x} = (x, z)$ denotes the horizontal coordinate and y the vertical coordinate of the wave particle at time t, with $\mathbf{x}_0$ and y_0 its horizontal and vertical position at rest respectively. As before, A is the amplitude, $\mathbf{k}$ the wavevector and ω the wave's angular frequency. Given the wavenumber $k = \|\mathbf{k}\|$, the term kA defines the sharpness of the wave crest. With $kA < 1$ the wave takes the form of an upside-down trochoid, with $kA = 1$ the form of an upside-down cycloid, and with $kA > 1$ the wave intersects itself, an undesireable effect which does not resemble real waves. See Figure 2.3 for waves with different kA. We may write the sum of

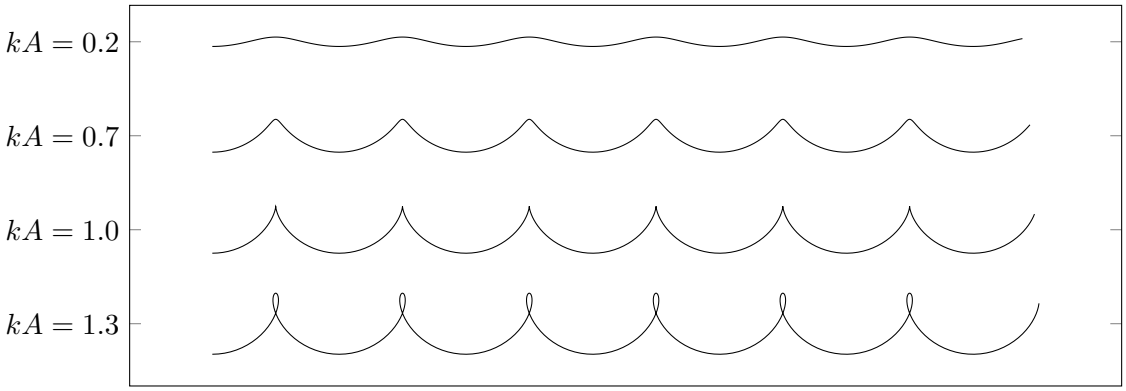


Figure 2.3: Example waves with different kA. One can see that the wave crest gets more pronounced as kA grows, with the wave intersecting itself when $kA > 1$.

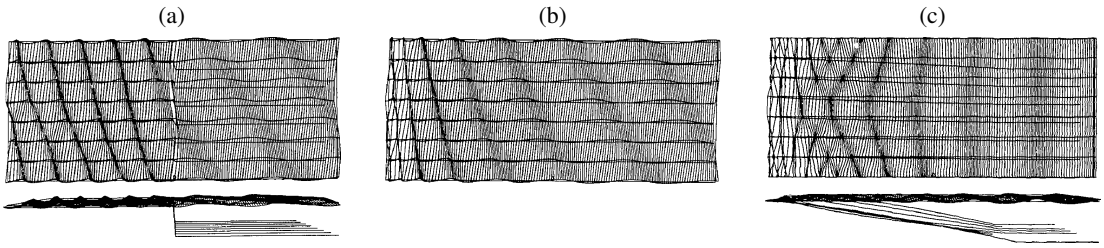


Figure 2.4: The refraction of waves. (a) Two bottoms of constant depth, the wavelengths are halved as the waves reach the shallow bottom on the left. (b) The beach slopes gently down from the left. One can see the wave fronts aligning themselves with the beach. (c) An undersea valley that affects both, the wavelengths as well as the travelling direction of the waves. Source: Fournier and Reeves [1986]

Figure 2.5: (a) Breaking waves on the shore, where the crests take the shape of the shore. (b) A large breaking wave. Source: Fournier and Reeves [1986]

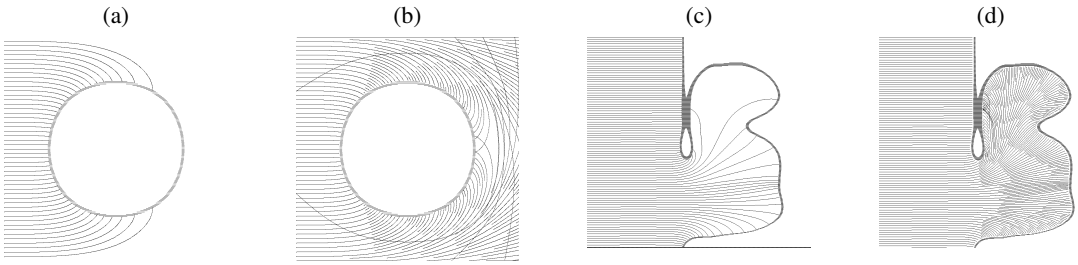


Figure 2.6: (a) Wave Trace of an island. (b) Dynamic Wave Trace of the same island. (c) Wave Trace of a bay. (d) Dynamic Wave Trace of the same bay. Source: Gonzato and Le Saëc [1997]

Gerstner waves involving N components as follows:

$$\mathbf{x} = \mathbf{x}_0 - \sum_i^N \frac{\mathbf{k}_i}{\|\mathbf{k_i}\|} A_i \sin\left(\mathbf{k}_i \cdot \mathbf{x}_0 - \omega_i t\right) \tag{2.4}$$

$$y = y_0 - \sum_i^N A_i \cos\left(\mathbf{k}_i \cdot \mathbf{x}_0 - \omega_i t\right) \tag{2.5}$$

To model waves in shallow water, Fournier and Reeves [1986] extend the Gerstner wave model in two ways. First, wave particle motion takes the underlying sea bed into account, allowing for waves to be refracted, i.e., to change their wavelength, their speed and their direction of travel based on water depth, see Figure 2.4. Second, the circular orbit of Gerstner waves approaching the shore is transformed into an elliptical orbit, which allows approximating the form of steepening and eventually breaking waves. Results of Fournier and Reeves are shown in Figure 2.5.

Ts'o and Barsky [1987] compute wave propagation near the shore with an algorithm called *wave tracing*, which launches orthogonal wave rays from the open sea along the direction of wave propagation. Wave refraction is computed based on Snell's law [Huygens, 2012], the wave trains are traced inside a uniform grid. In case rays tracing the waves' path diverge significantly from a straight line, there may remain undersampled areas, leading to parts of the surface lacking detail. Gonzato and Le Saëc [1997] improve upon this by generating new rays in undersampled areas, allowing their *dynamic wave tracing* algorithm to more densely sample the water surface surrounding islands and the water surface inside bays. See Figure 2.6 for a comparison between wave tracing and dynamic wave tracing. Moreover, Gonzato and Le Saëc modify the wave model to include plunging breaking waves. Three additional functions achieve said goal. First, a stretch function imitates Biesel's law [Biesel, 1952] by progressively stretching the wave on the crest along its major axis. Second and third, an orientation and a displacement function rotate and translate the wave's crest downwards, towards the water surface below the crest. Gonzato and Le Saëc [2000] augment their work with the inclusion of wave diffraction caused by jetties, and with the addition of capillary waves which are modeled using fractal noise.

Parametric models are well suited to model propagating water fronts, where a small number of wavenumbers may suffice to obtain satisfactory results. Because one has basically to select

the involved wavelengths and amplitudes by trial and error, it follows that parametric models are not particularly eligible to represent an agitated water surface, as it would involve a significantly higher number of waves. The user may have a hard time assembling a meaningful number of wavenumbers and amplitudes, with the result being potentially infeasible because of the accumulated computational cost.

2.2 Spectral Models

Oceanographic research employs *wave spectra* to model the deep ocean [Kinsman, 2002]. The sea is assumed as a linear superposition of sinusoids with many different wavelengths, frequencies and phases, travelling in different directions. The wave spectrum gives the distribution of wave energy among different wave frequencies or wavelengths. Given horizontal coordinate $\mathbf{x}$, according to Tessendorf [1999] we may compute the vertical coordinate at time t via an Inverse Discrete Fourier Transform as follows:

$$h(\mathbf{x}, t) = \sum_{\mathbf{k}} A(\mathbf{k}, t) \mathrm{e}^{\mathrm{i}\mathbf{k}^{\mathsf{T}}\mathbf{x}} \tag{2.6}$$

where $\mathbf{k}$ represents the wavevectors of the most significant frequency components, and $A(\mathbf{k}, t)$ the amplitudes obtained from the wave spectrum. At this point we refer to Chapter 3 for a more thorough discussion of the theoretical background of wave spectra. For now it shall suffice to give an overview of key properties of wave spectra.

First, all wave spectra are static [Kinsman, 2002]. Static in the sense, that a wave spectrum represents a certain sea state based on parameters such as wind speed and the distance from shore. To change the sea state requires the synthesis of a new wave spectrum. Thus, for example, if one would like to simulate a calm sea which gets agitated as a storm approaches, then one needs to generate a wave spectrum for each specific wind speed, and optionally interpolate between those spectra.

Second, wave spectra are compact models which require only a small number of input parameters defining the desired sea state [Kinsman, 2002, Young, 1999]. Since wave spectra are homogeneous models, all parameters are uniform for the entire ocean surface, there is no room for local variations as parametric models would allow. Wave spectra assume the ocean surface to be dominated by the interplay of wind and gravity, hence wind speed is the parameter common to all models. Another well established parameter is *fetch*, which is the distance over which the wind blows before it reaches the ocean patch the wave spectrum simulates. Water depth, if supported by the wave spectrum model, is uniform like all other parameters, hence the sea bed is assumed planar and parallel to the ocean surface at rest. It follows that no wave transitions from one water depth to another, therefore no wave refraction may take place [Mei, 1989].

Third, the Inverse Discrete Fourier Transform may leverage the Fast Fourier Transform algorithm [Cooley and Tukey, 1965], which makes the computation of Equation 2.6 most efficient. Hence, given the same computational cost, it is possible for spectral models to involve way more wavenumbers in the sum of waves than parametric models could. Moreover, thanks to the Discrete Fourier Transform we get a seamless tileable surface. The downside of the Fourier

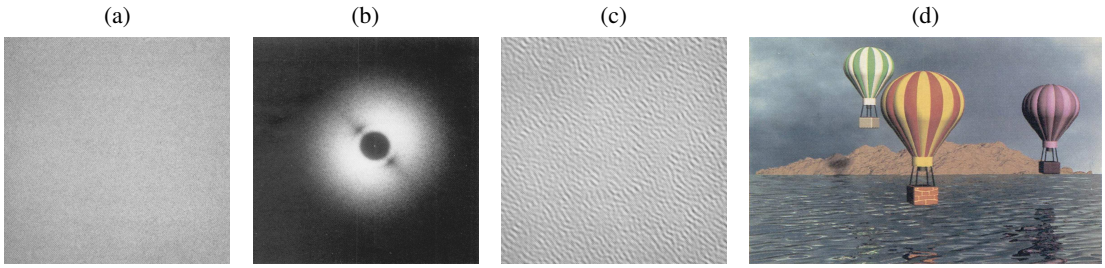


Figure 2.7: (a) White noise. (b) The Pierson-Moskowitz spectrum combined with the directional spreading function. (c) The synthesized heightfield. (d) Rendering of the sea surface. Source: Mastin et al. [1987]

Transform is that the range and uniform sampling density of spatial coordinates $\mathbf{x}$ directly defines the set of wavevectors $\mathbf{k}$ contributing to the sum of waves. In contrast to the parametric models we are neither able to cherry-pick certain wavevectors, nor to omit a range of wavevectors, nor to to exert some form of local control where we modify the set of wavevectors based on location $\mathbf{x}$. Despite those drawbacks, Mitchell [2005] and Thon et al. [2000] still argue that from an algorithmic point of view one should choose a spectral approach over a parametric one. Only by computing the sum of waves with the highly efficient Fast Fourier Transform algorithm, one is able to include the vast amount of wavenumbers necessary to synthesize a strongly agitated ocean surface.

Mastin et al. [1987] introduced wave spectra to computer graphics. Their approach is a simple one: generate white noise to represent random amplitudes and phases, then filter it with the Pierson-Moskowitz spectrum [Pierson and Moskowitz, 1964] in frequency space. The Pierson-Moskowitz spectrum gives wave energy per wavenumber, but not its distribution across wavevectors. In short, there is no information telling in which direction the waves are moving. Thus, Mastin et al. combine the Pierson-Moskowitz spectrum with a *directional spreading function*, namely the one proposed in Hasselmann et al. [1980]. A directional spreading function distributes wave energy across wavevectors, where the energy peak is usually to be found aligned with the wind direction and the remaining energy fanning out at its sides. Results of Mastin et al. [1987] are shown in Figure 2.7. Premoze and Ashikhmin [2000], on the other hand, make use of the JONSWAP spectrum [Hasselman et al., 1973], as it allows for more varied ocean surfaces than the Pierson-Moskowitz spectrum.

One of the most well-known works in computer graphics related to wave spectra is by Tessendorf [1999]. Therein, Tessendorf describes how the ocean was brought to life for productions such as *Waterworld* and *Titanic*. In contrast to Mastin et al., Tessendorf does not use white noise, but Gaussian random numbers, because waves in deep sea areas are often observed to follow a Gaussian distribution. Moreover, Tessendorf makes use of the Phillips spectrum, a spectrum devised by himself specifically for computer graphics purposes, but loosely based on oceanographic research published in Phillips [1958, 1985].

Ocean surfaces based on wave spectra all share the same issue, namely the all too gentle form of wave troughs and wave crests. Because the surface is computed as a linear superposition of

(a) (b)

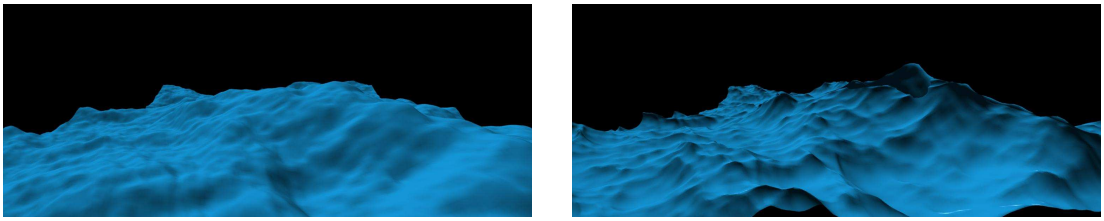


Figure 2.8: (a) Ocean surface sythesized with the Phillips spectrum. (b) The same surface with the choppy wave algorithm applied. Source: Tessendorf [1999]

(a) (b)

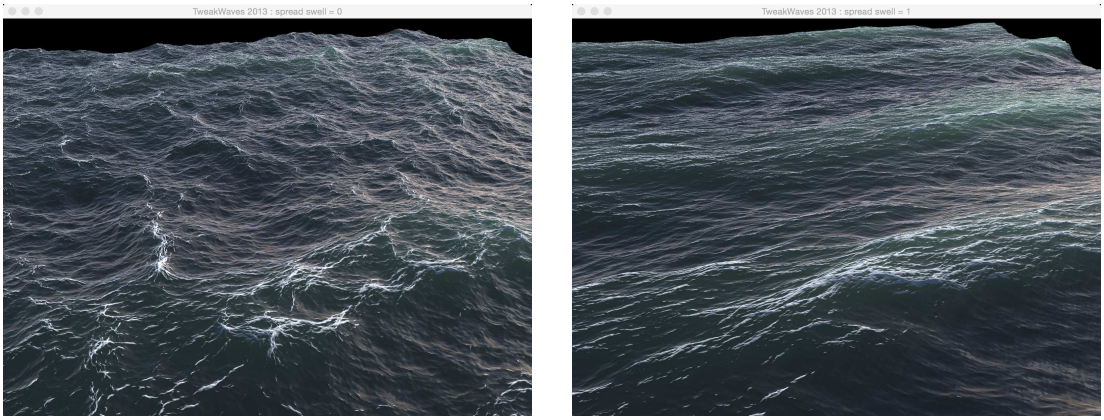


Figure 2.9: (a) The ocean without swell. (b) The same ocean, but dominated by swell. Source: Horvath [2015]

sinusoids, it is only natural that the result shares the general form of its underlying building blocks. But a sea exposed to strong winds or even a storm is characterized by steep waves with sharp crests. Tessendorf [1999] presents the *choppy wave* algorithm which overcomes this specific issue by allowing the waves to take a form similar to Gerstner waves, with deep troughs and sharp crests, see Figure 2.8.

Horvath [2015] improves upon Tessendorf's work by adding support for *swell*. Swell is defined as waves which have travelled out of their generating area. According to Ochi [2005], one may categorize fairly large waves that are observed on a sea with minor or even no wind as swell. Said waves then run across, or mix, with the local wind-generated waves. Horvath explains that the addition of swell allows for a feature often requested by artists, namely, the synthesis of a calm sea that is nevertheless dominated by elongated, parallel waves. Figure 2.9 depicts an example thereof.

(a) (b) (c)

Figure 2.10: Examples of early attempts to generate plausible water surfaces on graphics hardware. (a) Source: Schneider and Westermann [2001] (b) Source: Isidoro et al. [2002] (c) Source: Finch [2004]

2.3 Hybrid Approaches

Hybrid approaches combine parametric and spectral models where wave spectra provide a realistic description of the wave components, and the parametric part allows fine-grained control over the model. Thon et al. [2000] and Thon and Ghazanfarpour [2002] use a linear superposition of Gerstner waves, where a small set of wavevectors and corresponding amplitudes is picked from a directional Pierson-Moskowitz spectrum. Said wavevectors need to be representative for a large part of the energy of the spectrum, thus it is unlikely for short wavelengths to be selected. As a remedy, a three dimensional turbulence function [Perlin, 1985] is added to generate small scale details. Lee et al. [2007] improve upon the work of Thon et al. by using the TMA spectrum [Hughes, 1984], which incorporates both deep water waves and basic support for shallow water waves. Fréchot [2007] presents an adaptive sampling scheme, where care is taken not only to include wavevectors with the most contribution to wave energy, but wavevectors which are representative for the entire spectrum: long waves, short waves and even capillary waves. More recently, Jeschke and Wojtan [2015] have shown an algorithm which generalizes spectrum-based methods to handle varying water depth and physically correct interactions with static obstacles. The latter includes the simulation of wave reflections, wave refractions, and wave diffractions at boundaries located inside the water body, e.g., reefs, islands, or the shore. However, in deep water, and in the absence of any obstacles, the method simply reverts to a homogeneous sea as represented by a wave spectrum.

2.4 GPU Implementations

The computer graphics community has been, and still is, hard at work to make algorithms presented in the previous sections capable to render in real time. The challenge is trifold: first, the synthesis of an animated ocean surface, second, efficient rendering of vast ocean surface geometry, and third, believable shading of the ocean surface.

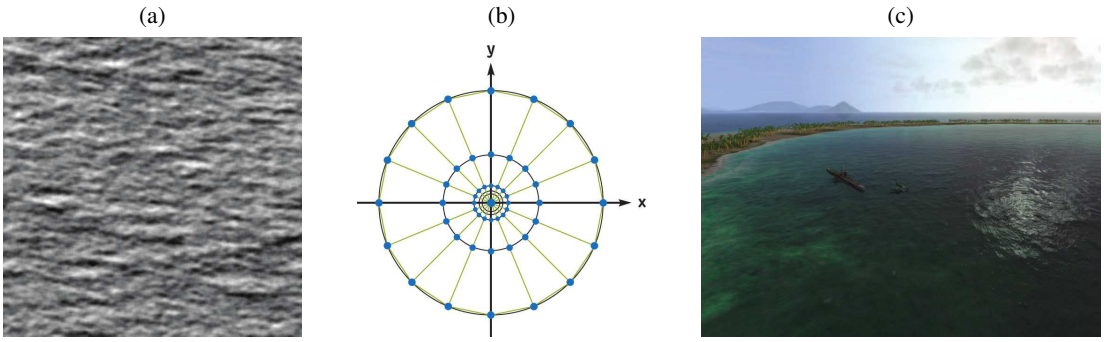


Figure 2.11: (a) A height map used for water displacement. (b) Radial grid centered at the camera position. (c) View with the ocean spanning to the horizon. Source: Kryachko [2005]

Early Work

Early attempts to generate plausible water surfaces on graphics hardware include Isidoro et al. [2002], Schneider and Westermann [2001] and Finch [2004]. Figure 2.10 depicts some of their results. To obtain interactive framerates, Schneider and Westermann compute the displacements caused by wave motion using two octaves of Perlin noise in a vertex shader on the GPU (Figure 2.10(a)). Isidoro et al. perturb an existing mesh by means of a sum of four low-frequency sinusoids computed in a vertex shader on the GPU (Figure 2.10(b)). Additional high-frequency detail is obtained by scrolling static normalmaps across the displaced surface. Finch uses a sum of four Gerstner waves to displace an existing mesh, where wavelengths that are too short to be represented by the mesh are filtered out automatically (Figure 2.10(c)). Moreover, Finch evaluates an additional sum, with more waves involved, to generate better detailed surface normals.

Adaptive Schemes

Viewing situations involving an ocean surface are varied: from close-ups where one may notice ripples on the surface, to views where the water surface may span until the horizon. Hence, to improve performance and visual fidelity, it is beneficial for rendering algorithms to focus on regions near the camera position, minimizing the number of samples according to the distance from the viewpoint.

Kryachko [2005] describes the algorithm used in an actual flight simulator. The ocean is modeled as a combination of handcrafted heightmaps tiled in space and time (Figure 2.11(a)). Four heightmaps are summed for lighting computations, where two of them with the largest scale are used to displace the ocean surface mesh. The latter is organized as a radial grid centered at the camera position, tesselated such that it provides more detail close to the viewer, see Figure 2.11(b). Radial grid positions are calculated in the vertex shader, as are displacements, allowing the algorithm to adapt the mesh vertices as well as displacement computations to the camera position automatically on the GPU for each frame. Still, one may notice that most of the radial mesh is always outside the view frustum, causing unnecessary overhead.

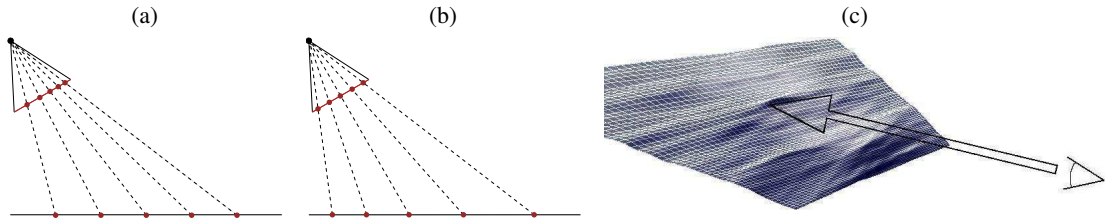


Figure 2.12: (a) Uniformly spaced vertices in world space, their projection onto the screen is not uniformly spaced though. (b) Uniformly spaced vertices in screen space projected onto world space base plane. (c) The result mesh obtained by projection of the vertices onto the base plane followed by displacement. One can see that the surface patches grow with distance from the viewer. Source: Hinsinger et al. [2002]

Hinsinger et al. [2002] present a more holistic approach, with their contribution being two-fold. One, an adaptive meshing scheme, and two, an adaptive simulation scheme. The basic idea for the adaptive surface mesh is to make sure that every surface element at rest covers the same area on screen. Thus, the screen is subdivided into quads which are projected onto the plane which represents the ocean surface at rest, see Figure 2.12. The result is a mesh which automatically adapts to the current camera position and gives more detail close to the viewer. Hinsinger et al. synthesize waves as a sum of Gerstner waves, where the vertices represent the sampling points. Derivatives are computed analytically for each vertex, resulting in better normal vectors than a finite differences approach would produce. To reduce aliasing as well as to save computation time, wavetrains smaller than a grid quad are removed from the sum of waves for that quad. Cui et al. [2004] uses the adaptive mesh by Hinsinger et al. for a marine simulator with three adjacent viewports, the wave model, however, is a simple sum of sinusoids with a small number of waves picked from a wave spectrum. Johanson [2004], on the other hand, improves upon the adaptive mesh of Hinsinger et al.. First, the projection of the mesh onto the base plane is moved from the CPU to a vertex shader on the GPU. Second, Johanson makes sure that the projection of the vertices works correctly in all possible viewing situations.

Bruneton et al. [2010] augment the work of Hinsinger et al. in various areas, the most prominent improvement being an adaptive lighting scheme. The latter is based on a hierarchical representation which combines geometry, normals and a BRDF [Ross et al., 2005], where the BRDF is specifically tailored to the statistical properties of ocean surfaces. Based on distance from the viewer, the algorithm transitions seamlessy from lighting computed with geometry and per pixel normals to lighting computed based solely on the statistical distribution of ocean surface slopes as modeled by the BRDF, see Figure 2.13. To save computation time for vertex positions, wavetrains are filtered according to the size of their associated projected grid cell in world space. For the gradient computation in the pixel shader there is an additional step, where wavetrains are filtered according to the projected size of the current pixel in world space. In addition, Bruneton et al. show the integration of the BRDF by Ross et al. into the computation of all necessary lighting terms, namely the reflected light from the sun and from the skydome, and the refracted light from the water, as seen in Figure 2.14.

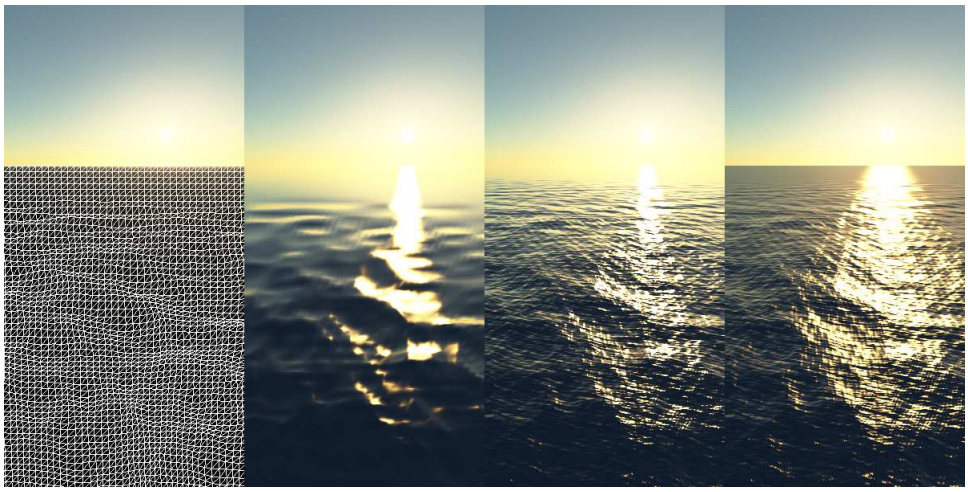


Figure 2.13: From left to right: The displaced surface mesh in screen space, lighting with geometry only, lighting with geometry and per pixel normals, lighting with geometry and per pixel normals and BRDF. Source: Bruneton et al. [2010]

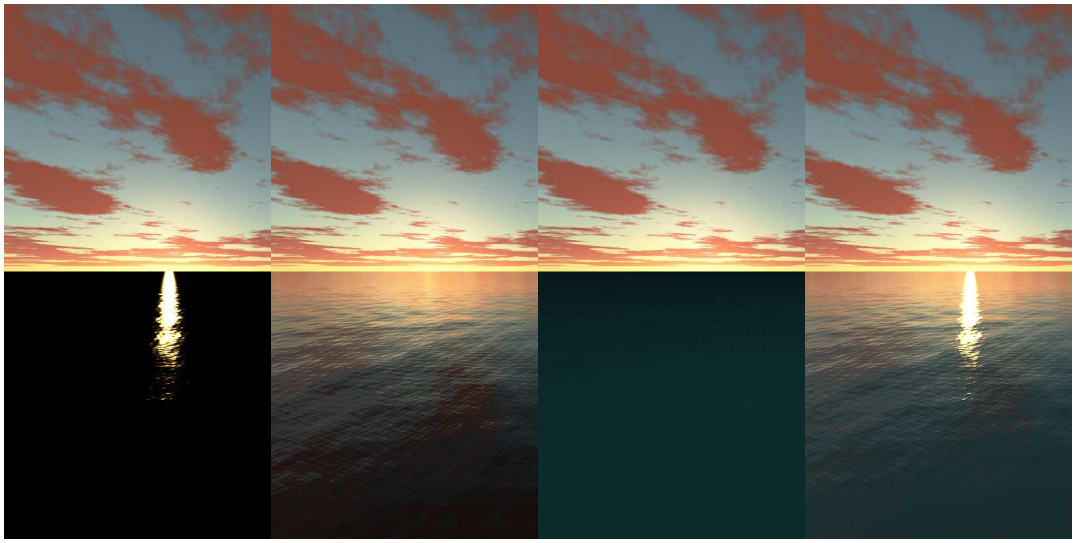


Figure 2.14: From left to right: The reflected sun light, the reflected sky light, the light refracted from the water, and the final result. Source: Bruneton et al. [2010]

Figure 2.15: A set of example ocean scenes with whitecaps. Source: Dupuy and Bruneton [2012]

Mitchell [2005] describes a partial GPU implementation of the ocean wave generation method presented in Tessendorf [1999]. To improve rendering performance, Mitchell synthesizes two water surface height maps, where one contains low frequency waveforms and the other contains low frequency and high frequency waveforms. The low frequency map is of low resolution and is used to displace the water surface mesh. The high resolution map is used to generate a normal map for shading. The result is an ocean surface which appears highly detailed while the underlying mesh is coarse.

Eric Bruneton [2010] combines the adaptive lighting scheme from Bruneton et al. [2010] with an ocean surface synthesized from the wave spectrum presented in Elfouhaily et al. [1997]. Wave heights, gradients, as well as the horizontal displacements required for the choppy wave algorithm by Tessendorf [1999] are computed entirely on the GPU. In an additional step, Dupuy and Bruneton [2012] add whitecaps to the ocean lighting model, see Figure 2.15. The algorithm by Dupuy and Bruneton is computationally expensive, as it requires four additional scalar fields to be synthesized, namely the first-order partial derivatives of the horizontal displacements. Said derivatives are then used to determine the locations of the whitecaps on the water surface. More recently, Chen et al. [2017] have shown a computationally less intensive approach, where a single additional scalar field, namely the vertical acceleration of the wave crests, serves as the criterion to locate whitecaps on the water surface.

As ocean surfaces generated from wave spectra allow for seamless tiling, one may need to make sure to reduce tiling artifacts, i.e., the viewer shall not notice that the ocean surface repeats itself in all directions. The approach taken by Rydahl [2009] and NVIDIA [2011] is to generate additional noise on the ocean surface. Eric Bruneton [2010] and Dupuy and Bruneton [2012], on the other hand, compute not just one ocean surface tile, but up to four, where all tiles are of different size and each one samples a different part of the wave spectrum. Even though such an approach is unable to entirely remove periodicity, the period is increased to the least common multiple of the different tile sizes.

Now that we have completed our discussion of related work, we may highlight the cornerstones of this thesis. First and foremost there is the lighting scheme by Bruneton et al. [2010], which has been specifically tailored to the deep water of the open ocean. Second, Eric Bruneton [2010] combines said lighting scheme with Tessendorf's choppy wave algorithm. Third, Dupuy and Bruneton [2012] extend the work of Eric Bruneton [2010] with whitecaps. All three, Bru-

neton et al. [2010], Eric Bruneton [2010], and Dupuy and Bruneton [2012] employ the meshing scheme by Hinsinger et al. [2002]. The latter severely restricts the camera's freedom of movement, therefore we will replace it with the improved version by Johanson [2004]. Moreover, as the work by Eric Bruneton [2010] contains only a single wave spectrum [Elfouhaily et al., 1997], we will evaluate an additional set of wave spectra with regard to their suitability for the above lighting scheme. Last, to improve upon the balance between model detail and computational workload, we will implement a multi-resolution variant of the choppy wave algorithm.

CHAPTER 3

Background

Ocean surface waves are generated by different kinds of forces, such as storms, earthquakes, the gravity of sun and moon, but with *wind* being the most prominent one. Each of these forces influences one or more frequency bands of the electromagnetic wave spectrum. The combination of said frequency bands constitutes the spectrum of ocean waves.

	Period Band Range (s)		Frequency Band Range (Hz)	
Wave Type	Start	End	Start	End
Capillary	0	1×10^{-1}	∞	1×10^{1}
Ultragravity	1×10^{-1}	1×10^{0}	1×10^{1}	1×10^{0}
Gravity	1×10^{0}	3×10^{1}	1×10^{0}	3.33×10^{-2}
Infragravity	3×10^{1}	3×10^{2}	3.33×10^{-2}	3.33×10^{-3}
Long Period	3×10^{2}	4.32×10^{4}	3.33×10^{-3}	2.32×10^{-5}
Tidal	4.32×10^{4}	8.64×10^{4}	2.32×10^{-5}	1.16×10^{-5}
Transtidal	8.64×10^{4}	∞	1.16×10^{-5}	0

Table 3.1: A classification of ocean surface waves by period and frequency.

Table 3.1 gives a compact overview of different kinds of ocean surface waves classified by period band as well as the corresponding frequency band. The shortest period waves are the *capillary waves*, with periods less than $0.1\,\mathrm{s}$. These are the first waves which one is able to notice when the wind starts to blow on the ocean surface, they appear like a fine structure of small ripples of nearly capillary dimension. Next are the *ultragravity waves* with periods ranging from $0.1\,\mathrm{s}$ to $1\,\mathrm{s}$. The ordinary *gravity waves* observed at sea have periods varying from $1\,\mathrm{s}$ to $30\,\mathrm{s}$ and are composed of two types of waves, namely *sea waves* and *swell*. Sea waves are gravity waves directly generated and affected by local winds, and after the wind ceases to blow they are called swell. More generally, swell consists of wind-generated gravity waves that are not significantly affected by the local wind, therefore they have been generated elsewhere or at some time in the past. Beyond the ordinary gravity waves there are *infragravity waves*, with

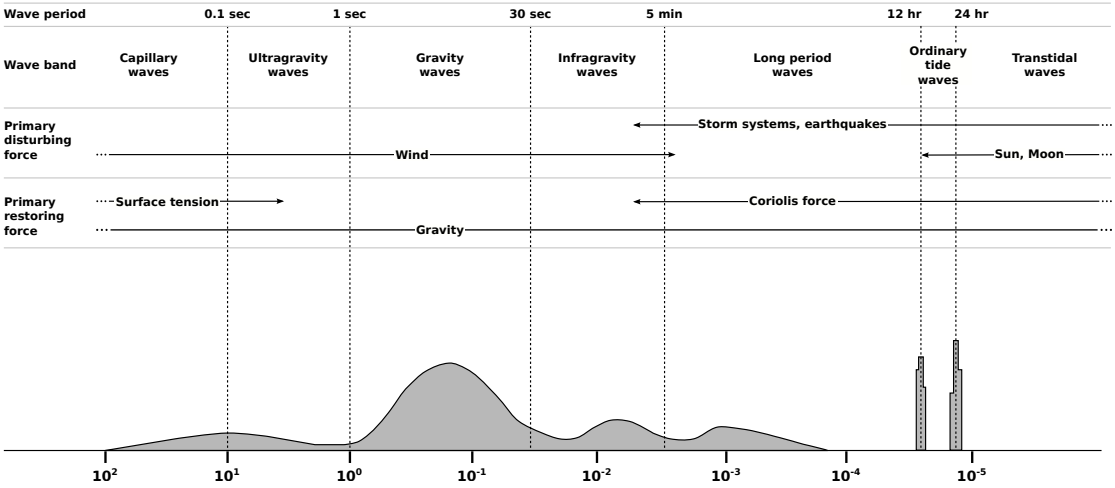


Figure 3.1: A tentative overview of which forces influence which wave frequency band, as well as the relative amount of energy each wave frequency band contains. Source: Munk [2010]

periods between $30\,\mathrm{s}$ and $5\,\mathrm{min}$. Then there are the *long-period waves* with periods in the range $5\,\mathrm{min}$ to $12\,\mathrm{h}$. Their representatives in the sea are tsunamis and storm surges. The *ordinary tidal waves* represent the astronomical tides with fixed wave periods of about $12\,\mathrm{h}$ and $24\,\mathrm{h}$. At the end of the spectrum there are the *transtidal waves* with periods greater than $24\,\mathrm{h}$. These include the longer period components of astronomical tides.

Ocean surface waves are also classified by their major governing forces. First, the disturbing force which causes the actual water displacement, and second, the restoring force which brings the water back to its position at rest. Winds are responsible for the generation of capillary waves, ultragravity waves, gravity waves and infragravity waves, whereas long-period waves like tsunamis are generated by storms and earthquakes. The ordinary tides are caused by the gravitational pull of sun and moon, transtidal waves, on the other hand, by storms, sun and moon. As soon as the water has been displaced, the force to bring capillary waves back to their position at rest is exerted by surface tension. Ultragravity waves, gravity waves and infragravity waves have gravity as their major restoring force. Gravity together with coriolis act as the restoring force for long-period waves, ordinary tidal waves and transtidal waves. Figure 3.1 pictures the entire spectrum of ocean surface waves and the forces responsible for various portions of the spectrum. As one can see, gravity is the main restoring force, therefore ocean surface waves are classified as *surface gravity waves*. Most of the energy of the spectrum is contained in two of the period ranges: the ordinary gravity waves, and the ordinary tidal waves. Ordinary tidal waves, infragravity waves, long period waves, as well as transtidal waves do not contribute significantly to the actual optical appearance of the ocean in deep water, therefore we may omit them. The remaining types of waves have wind as the common disturbing force and are therefore grouped under the term *wind generated waves*, or in short, *wind waves*. Wind waves still contain a large part of the energy of the spectrum, and range from ripples with less than a centimeter in height to huge waves more than thirty meters high.

In order to be able to model the surface of such an intricate fluid system as the ocean, one has to reduce complexity. Research fields such as ocean engineering or coastal engineering employ the *Airy wave theory* [Airy, 1845] to model the properties of the sea. Airy wave theory, also known as *linear wave theory*, describes the propagation of gravity waves on the surface of a homogeneous fluid with a set of *linear* equations. These equations give a decent approximation of the wave dynamics and kinematics with enough accuracy to model the state of the sea over a limited amount of time.

The remainder of this chapter is organized as follows: Section 3.1 gives an introduction to linear wave theory and its capability to model the ocean surface. Section 3.2 presents the wave spectrum concept, whereas Section 3.3 outlines the steps necessary to synthesize an ocean surface based on a wave spectrum. Last, Section 3.4 discusses the wave spectra which we implemented as part of this work.

3.1 Linear Theory of Ocean Surface Waves

Linear wave theory makes a set of assumptions about the properties of the fluid, such as its viscosity, compressibility and curl. Because fluid dynamics, associated fluid properties, and the derivation of the linear wave theory go beyond the scope of this thesis, we will refer the interested reader to Airy [1845], Batchelor [2000] and Kinsman [2002]. *But* we will mention two core assumptions of linear wave theory which are easy to picture:

- The water body has a uniform mean depth.
- The wave amplitudes are small in relation to the size of the water body.

Suppose we observe an idealized ocean in which exactly one sine wave is travelling constantly. The parameters defining the sine wave – amplitude, frequency, length and direction – are all fixed. To simplify the mathematics we assume a two-dimensional ocean, with one dimension representing the wave's direction of travel and the other the wave's vertical displacement. With these assumptions we can describe surface elevation as a sinusoid:

$$\eta(x,t) = A\cos(kx - \omega t) \tag{3.1}$$

with

$$k = \frac{2\pi}{\lambda} \qquad \omega = 2\pi f \qquad f = \frac{1}{T}$$

where A is the amplitude, k is called the *wavenumber*, x is the horizontal position, ω is the *wave frequency* in radians per second, t represents the time, λ is the *wavelength*, f is the *wave frequency* in Hertz (Hz) and T is the wave period in seconds. According to Airy wave theory, wavenumber k and angular frequency ω are not independent parameters, but are coupled by the *dispersion relation*. We will elaborate on the dispersion relation at a later point, though.

We may picture the wave form as defined by Equation 3.1 in an easy way if we either focus on a fixed position x or a fixed point in time t, see Figure 3.2. If we assume a fixed position, we can see the surface elevation evolve through time at that position. On the other hand, if we

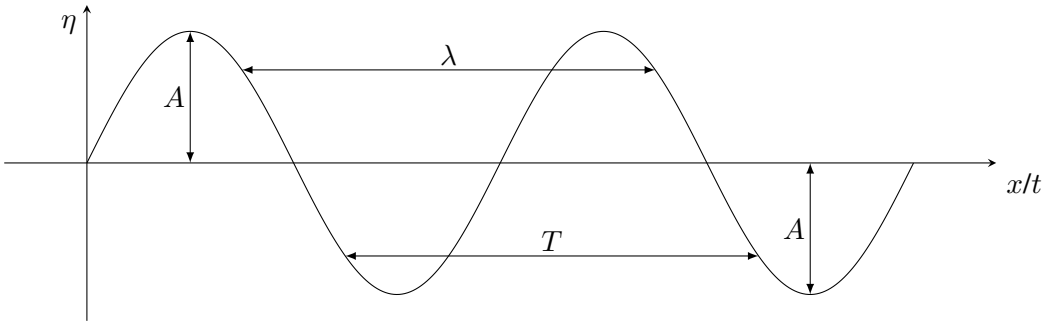


Figure 3.2: A sinusoidal wave with amplitude A, wavelength λ, and wave period T.

assume a fixed point in time, we can observe surface elevation at all positions at that specific time. The wave's high point is called a *crest*, its low point a *trough*. Because surface elevation is described by a cosinus function, all crests and all troughs will have the same elevation. Another effect of the cosinus function is that the surface elevation is limited by the amplitude A. We will call the difference in elevation between crest and trough the *wave height*. It is easy to see that wave height H is defined as $H = 2A$.

Phase Velocity

In linear wave theory, the rate at which a particular phase of a wave propagates in space is called the *phase velocity*. The phase velocity is a vector and has an associated direction, the *phase speed* on the other hand refers only to the magnitude of the phase velocity. The most comprehensible example of phase speed is the rate of propagation of the wave crest. During one wave period T the wave crest travels a distance equal to the wavelength λ. We may generalize this to other phases than the wave crest. Given a constant phase

$$const = kx - \omega t$$

the surface elevation η will always be the same. We rewrite the term to

$$x = \frac{\omega}{k}t + \frac{const}{k}$$

which gives us all positions x where the wave has the same elevation. These positions are time-dependent, differentiation gives us phase speed c:

$$c = \frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\omega}{k} = \frac{\lambda}{T}$$

The Dispersion Relation

In the context of water surface waves, *dispersion* refers to *frequency dispersion*, which describes the effect of waves at different wavelengths travelling at different phase speeds. We have seen that phase speed $c = \lambda/T$ involves wavelength and frequency, and alternatively $c = \omega/k$, which

includes angular frequency and wavenumber. We focus on the latter formulation for phase speed, because linear wave theory defines a functional relationship between the two factors involved:

$$\omega^2 = gk \tanh(kd) \tag{3.2}$$

where g is the acceleration of earth's gravity and d is the water depth. Equation 3.2 is called the *dispersion relation*. There exist two useful approximations of the equation which do not involve the $\tanh$ term:

- *Shallow water approximation* – the water depth d is much smaller than the wave length λ. Assume $d \ll \lambda$, then $0 \leq kd \ll 1$ and $\tanh(kd) \approx kd$.

- *Deep water approximation* – the water depth d is much larger than the wave length λ. Assume $d \gg \lambda$, then $kd \gg 1$ and $\tanh(kd) \approx 1$.

The reduced dispersion relations are as follows:

$\omega^2 = gk^2d$	with	$d < \frac{\lambda}{20}$	Shallow water dispersion relation
$\omega^2 = gk$	with	$d > \frac{\lambda}{2}$	Deep water dispersion relation

Given the dispersion relation approximations for shallow and deep water, the corresponding phase speeds are:

$$c = \sqrt{gd} \qquad \text{Shallow water phase speed} \tag{3.3}$$

$$c = \sqrt{\frac{g}{k}} = \frac{g}{\omega} \qquad \text{Deep water phase speed} \tag{3.4}$$

According to Equation 3.3, the phase speed in shallow water depends exclusively on water depth and gravitational acceleration, there is no connection to wavelength or frequency. We may conclude that waves with different wavelengths travel with the same phase speed, it follows that shallow water is not dispersive. Phase speed in deep water, on the other hand, does not relate to water depth, but to wavelength and frequency, which makes deep water dispersive. We can expand Equation 3.4 to:

$$c = \sqrt{\frac{g\lambda}{2\pi}} = \frac{gT}{2\pi} \tag{3.5}$$

Looking at Equation 3.5 we can conclude that phase speed in deep water increases with wavelength λ *and* with wave period T. Waves in deep water with large wavelengths/periods travel faster than those with smaller ones.

Note: In the context of this work we concern ourselves only with the open ocean, where the water is assumed to be deep. Should occasions arise where we are required to derive terms and formulas based on phase speed or the dispersion relation, then we will rely exclusively on the deep water phase speed and the deep water dispersion relation.

Three-Dimensional Wave

Until now we were discussing a two-dimensional ocean defined by exactly one sinus wave. As a first step to get to a more realistic ocean surface representation, we will extend Equation 3.1 to form a wave in three-dimensional space. In contrast to Equation 3.1, where the wave's travelling direction is fixed, the wave has to be able to travel in an arbitrary direction on a plane. Moreover, we need to handle two-dimensional input positions. The sinusoid describing surface elevation in three-dimensional space is defined as follows:

$$\eta(\mathbf{x}, t) = A\cos(\mathbf{k}^{\mathsf{T}}\mathbf{x} - \omega t) \tag{3.6}$$

where $\mathbf{x} = (x, z)$ denotes the observed point on a plane and $\mathbf{k} = (k_x, k_z)$ represents the travelling direction of the wave. The wavenumber k of *wavevector* $\mathbf{k}$ is determined by the wavevector's magnitude:

$$k = \|\mathbf{k}\|$$

Because of the dispersion relation, all waves of same magnitude k share the same angular frequency ω.

Complex Exponential Wave Form

Before we start handling a large number of sinusoids, we will improve our notation to make it more compact. In order to do that we employ *Euler's formula* [Euler, 1748]:

$$e^{ix} = \cos x + i\sin x \qquad x \in \mathbb{R} \tag{3.7}$$

$$e^{-ix} = \cos x - i\sin x \qquad x \in \mathbb{R} \tag{3.8}$$

We may extract both the real and the imaginary part separately:

$$\cos x = \mathcal{R}\{e^{ix}\}$$
$$\sin x = \mathcal{I}\{e^{ix}\}$$

where $\mathcal{R}$ denotes the real part of the exponential and $\mathcal{I}$ the complex one. Alternatively we may solve Equations 3.7 and 3.8 as a pair in the variables $\cos x$ and $\sin x$:

$$\cos x = \frac{e^{ix} + e^{-ix}}{2}$$
$$\sin x = \frac{e^{ix} - e^{-ix}}{2i}$$

Now we are able to rewrite our current formulation of a sinusoid in three-dimensional space from Equation 3.6 as an exponential:

$$\eta(\mathbf{x}, t) = \mathcal{R}\{A\mathrm{e}^{\mathrm{i}[\mathbf{k}^{\mathsf{T}}\mathbf{x} - \omega t]}\} \tag{3.9}$$

or equivalently

$$p = A\mathrm{e}^{\mathrm{i}[\mathbf{k}^{\mathsf{T}}\mathbf{x}-\omega t]}$$
$$\eta(\mathbf{x},t) = \frac{p+p^*}{2}$$

where p^* denotes the *complex conjugate* of p. From now on we will omit the explicit $\mathcal{R}$ in equations expressing η, because surface elevation is a real number.

Sum of Waves

Now that we have a description of waves in three-dimensional space, we need to enhance the description of the ocean surface. It is obvious that the ocean does not consist of only one wave travelling, but of an infinite number of waves with different lengths and directions. Because we still find ourselves in the realm of *linear* wave theory, we may reach an adequate solution by *combination*. A linear combination of solutions constitutes a valid solution as well.

Each combination of amplitude A, wavevector $\mathbf{k}$ and wave frequency ω identifies one particular wave. As a next step we get rid of the restriction of one single amplitude A for all wave components. We replace the constant amplitude A with $a(\mathbf{k})$ in order to create a dependency between wavevector and wave amplitude. Furthermore, because of the dispersion relation, we can express frequency as a function of the wavevector, $\omega = \omega(\mathbf{k})$. Now we are able to rewrite Equation 3.9 as follows:

$$\eta(\mathbf{x},t) = a(\mathbf{k})\mathrm{e}^{\mathrm{i}(\mathbf{k}^{\mathsf{T}}\mathbf{x}-\omega(\mathbf{k})t)} \tag{3.10}$$

We already mentioned that we seek a solution by linear combination. Based on Equation 3.10 we may express surface elevation as an infinite sum of sinusoids:

$$\eta(\mathbf{x},t) = \iint_{\mathbf{k}} a(\mathbf{k})\ \mathrm{e}^{\mathrm{i}(\mathbf{k}^{\mathsf{T}}\mathbf{x}-\omega(\mathbf{k})t)}\ \mathrm{d}\mathbf{k} \tag{3.11}$$

where $a(\mathbf{k})$ is called the *amplitude spectrum* of $\eta(\mathbf{x},t)$. According to Kinsman [2002] it is more expedient to represent the vertical displacement of the water surface by *Generalized Fourier Transforms* [Lighthill, 1958]. Thus, Kinsman transforms Equation 3.11 into a sea surface representation with two spatial dimensions and one time dimension:

$$\eta(\mathbf{x},t) = \iint_{\mathbf{k}}\int_{\omega} B(\mathbf{k},\omega)\ \mathrm{e}^{\mathrm{i}(\mathbf{k}^{\mathsf{T}}\mathbf{x}-\omega t)}\ \mathrm{d}\mathbf{k}\ \mathrm{d}\omega \tag{3.12}$$

where the $\mathbf{k}$ integration is over all wavenumber space and the ω integration is over all frequencies. $B(\mathbf{k},\omega)$ is called the three-dimensional amplitude spectrum of $\eta(\mathbf{x},t)$. Notice the missing dependency of ω on $\mathbf{k}$ in the exponent, it is the responsibility of $B(\mathbf{k},\omega)$ to model that relationship. We may integrate separately over all frequencies to obtain the two-dimensional amplitude spectrum $B(\mathbf{k},t)$:

$$B(\mathbf{k},t) = \int_{\omega} B(\mathbf{k},\omega)\ \mathrm{e}^{-\mathrm{i}\omega t}\ \mathrm{d}\omega \tag{3.13}$$

Alternatively, we may integrate over all wavenumbers to obtain the one-dimensional amplitude spectrum $B(\mathbf{x}, \omega)$:

$$B(\mathbf{x}, \omega) = \iint_{\mathbf{k}} B(\mathbf{k}, \omega)\, \mathrm{e}^{\mathrm{i}\mathbf{k}^{\mathsf{T}}\mathbf{x}}\, \mathrm{d}\mathbf{k} \tag{3.14}$$

Thus, we may rewrite Equation 3.12 in terms of Equation 3.13 and Equation 3.14:

$$\eta(\mathbf{x}, t) = \iint_{\mathbf{k}} B(\mathbf{k}, t)\, \mathrm{e}^{\mathrm{i}\mathbf{k}^{\mathsf{T}}\mathbf{x}}\, \mathrm{d}\mathbf{k} = \int_{\omega} B(\mathbf{x}, \omega)\, \mathrm{e}^{-\mathrm{i}\omega t}\, \mathrm{d}\omega \tag{3.15}$$

3.2 The Specification of a Random Sea

Equation 3.15 may be able to describe the ocean surface, but at the cost of deducing an amplitude spectrum for each observed point on the sea surface, a both tedious and unfeasible task. At this point we are in need of a general description of the amplitude spectrum which is valid at all points $\mathbf{x}$ and times t. Fortunately, modern physical oceanography is based on exactly such a formulation, which has been developed by Neumann and Pierson Jr [1966].

Neumann and Pierson Jr combine oceanography, physics and stochastics in order to describe the ocean surface. A fundamental assumption of their theory is that surface disturbance is formed from many contributions caused by relatively unrelated forces at different times. Therefore, it is reasonable to presume that the elements in the summation of Equation 3.15 are statistically independent. Neumann and Pierson Jr model the wind-driven ocean surface as a *spatially homogeneous* and *temporally stationary* Gaussian random process [Grimmett and Stirzaker, 2001]. The random process is said to be *homogeneous*, because it is independent of the origin selected for positions $\mathbf{x}$, furthermore it is *stationary* since absolute time is irrelevant, only time difference matters.

The Energy Spectrum

Given a spatially homogeneous and temporally stationary wave field, and a respective three-dimensional amplitude spectrum $B(\mathbf{k}, \omega)$, we may write:

$$\Theta(\mathbf{k}, \omega) = \frac{B(\mathbf{k}, \omega)^2}{2} = \overline{B(\mathbf{k}, \omega) B^*(\mathbf{k}, \omega)} \tag{3.16}$$

where the $\overline{B}$ operator denotes the mean and $\Theta(\mathbf{k}, \omega)$ represents the *mean square displacement* of wave amplitude $B(\mathbf{k}, \omega)$. The mean square displacement of a wave is connected to wave energy E_w as follows:

$$E_w = \rho g\, \Theta(\mathbf{k}, \omega) \tag{3.17}$$

where ρ is the water density and g the accleration of earth's gravity. Equation 3.17 is the reason $\Theta(\mathbf{k}, \omega)$ in literature is referred to as the three-dimensional *energy spectrum* associated with $B(\mathbf{k}, \omega)$. Now we are able to define mean surface energy E_s based on combined wave energy:

$$E_s = \rho g \iint_{\mathbf{k}} \int_{\omega} \Theta(\mathbf{k}, \omega)\, \mathrm{d}\mathbf{k}\, \mathrm{d}\omega \tag{3.18}$$

Looking back at Equation 3.17, which makes wave energy dependent on the mean square displacement of wave amplitudes, we may deduce that mean surface energy is dependent on the *mean square surface displacement*, denoted as $\overline{\eta^2}$. Therefore, we may write:

$$\overline{\eta^2} = \iint_{\mathbf{k}} \int_{\omega} \Theta(\mathbf{k}, \omega) \, \mathrm{d}\mathbf{k} \, \mathrm{d}\omega \tag{3.19}$$

Later on, we will be in need of energy spectrum formulations which have a lower dimensionality than the three-dimensional $\Theta(\mathbf{k}, \omega)$. As such, we introduce the two-dimensional energy spectrum, in literature known as *wavenumber spectrum*, which is defined as follows:

$$\Theta(\mathbf{k}) = \int_{\omega} \Theta(\mathbf{k}, \omega) \, \mathrm{d}\omega \tag{3.20}$$

The wavenumber spectrum gives the contribution to the wave energy arising from wave components with wavenumber k dependent on wavevector $\mathbf{k}$, irrespective of the frequencies ω associated with that wavenumber. In contrast, the one-dimensional energy spectrum, in literature known as *frequency spectrum*, is defined as follows:

$$\Theta(\omega) = \int_{\mathbf{k}} \Theta(\mathbf{k}, \omega) \, \mathrm{d}\mathbf{k} \tag{3.21}$$

The frequency spectrum gives the contributions to the energy coming from each frequency ω, irrespective of the wavenumbers associated with that frequency.

Note: The energy spectrum as presented is severely restricted in its capabilities to model a *dynamic* ocean. Dynamic, in this context, means an ocean which may start as a perfectly flat surface, is actuated to a fully developed sea by wind, decays back into a near motionless state, and so on and so forth. The energy spectrum as discussed only models waves on an ocean with infinite extent over which a uniform wind has been blowing forever. In context of this document, such a model of the ocean is sufficient, there is no need to simulate the actual generation and decay of waves over time.

The Random Process

Based on the wave energy spectrum, Neumann and Pierson Jr [1966] describe the sea surface as a Gaussian random process. As such, we are not dealing with one concrete ocean surface η, but with an *ensemble* of different sea surfaces $\{\eta\}$ which share the same statistics. Because the random process is a Gaussian one, it has an associated Gaussian Distribution with mean $\mathrm{E}[\eta]$ and variance $\mathrm{Var}[\eta] = \sigma^2$. Considering a sea surface in combination with linear wave theory, where a symmetry between troughs and crests is a given, the mean is easily found:

$$\mathrm{E}[\eta] = 0$$

Moreover, all spectral components $B(\mathbf{k}, \omega)$ are assumed to be *statistically independent*, each with mean zero and variance $\Theta(\mathbf{k}, \omega)$, therefore the variance of all components combined is:

$$\mathrm{Var}[\eta] = \sigma^2 = \iint_{\mathbf{k}} \int_{\omega} \Theta(\mathbf{k}, \omega) \, \mathrm{d}\mathbf{k} \, \mathrm{d}\omega$$

which is the same as Equation 3.19. The variance σ^2 is equal to the mean square surface elevation $\overline{\eta^2}$. With mean and variance at hand, we may write the Gaussian Distribution that the values of sea surface elevation are drawn from:

$$pr[\eta] = \frac{1}{\sqrt{2\pi\sigma^2}} \, \mathrm{e}^{-\frac{\eta^2}{2\sigma^2}} \tag{3.22}$$

Note that neither position nor time appear in the distribution, because the process is modeled as spatially and temporally invariant. As of now, we may formulate surface elevation as a realisation of a stochastic process:

$$\eta(\mathbf{x}, t) = \iint_{\mathbf{k}} \int_{\omega} \xi \, \mathrm{e}^{\mathrm{i}(\mathbf{k}^{\mathsf{T}}\mathbf{x} - \omega t)} \, \mathrm{d}\mathbf{k} \, \mathrm{d}\omega \tag{3.23}$$

where ξ is a Gauss distributed variable with mean $\mathrm{E}[\eta] = 0$, and variance $\mathrm{Var}[\eta] = \sigma^2$ as defined by the energy spectrum $\Theta(\mathbf{k}, \omega)$. One may realize that in context of this formulation of sea surface elevation, the energy spectrum is the key to believable ocean surfaces, as it needs to be able to model the underlying physical processes to give realistic results.

From an algorithmic point of view, we should not use three-dimensional energy spectra $\Theta(\mathbf{k}, \omega)$ to synthesize the ocean surface, as the computational cost of Fourier Transforms in three dimensions would simply be too high. On the other hand, we are unable to employ one-dimensional frequency spectra $\Theta(\omega)$ as-is, because they do not contain any information about the travelling direction of waves, and therefore are not able to compute surface elevation with two-dimensional positions $\mathbf{x}$ as input. Hence, we have to settle for the two-dimensional wavenumber spectrum $\Theta(\mathbf{k})$ as basis. We may write surface elevation as follows:

$$\eta(\mathbf{x}, t) = \iint_{\mathbf{k}} \xi \, \mathrm{e}^{-\mathrm{i}\omega(\mathbf{k})t} \, \mathrm{e}^{\mathrm{i}\mathbf{k}^{\mathsf{T}}\mathbf{x}} \, \mathrm{d}\mathbf{k} \tag{3.24}$$

where, in line with Equation 3.23, ξ is a Gauss distributed variable with mean $\mathrm{E}[\eta] = 0$ and variance $\mathrm{Var}[\eta] = \sigma^2$ defined by the energy spectrum $\Theta(\mathbf{k})$. Moreover, for animation purposes, we add the term $\mathrm{e}^{-\mathrm{i}\omega(\mathbf{k})t}$ to explicitely handle time t, because the two-dimensional wavenumber spectrum has no notion of it. Equation 3.24 represents a two-dimensional Inverse Fourier Transform of Fourier components $\xi \mathrm{e}^{-\mathrm{i}\omega(\mathbf{k})t}$. For the remainder of the document, Equation 3.24 will be the basis for all further discussions of specific wave spectrum models. Moreover, in Section 3.4 we will discuss the conversion of the more common one-dimensional frequency spectra $\Theta(\omega)$ into two-dimensional wavenumber spectra $\Theta(\mathbf{k})$ for our practical usage.

3.3 Discretisation

For computer graphics applications, we are in need of a discrete approximation of surface elevation as defined by Equation 3.24, which gives acceptable results as well as computes said results in a reasonable amount of time. We may write a rough sketch of such a discrete formulation like the following:

$$\eta(\mathbf{x}, t) = \sum_{\mathbf{k}} \xi \, \mathrm{e}^{-\mathrm{i}\omega(\mathbf{k})t} \, \mathrm{e}^{\mathrm{i}\mathbf{k}^{\mathsf{T}}\mathbf{x}} \tag{3.25}$$

Equation 3.25 represents an Inverse Discrete Fourier Transform of Fourier components $\xi e^{-i\omega(\mathbf{k})t}$, which we are able to compute in a very efficient manner by means of the Fast Fourier Transform, *FFT* in short. The interested reader may find extensive discussions of the Discrete Fourier Transform and the Fast Fourier algorithm in Bracewell [2000] and Press [2007].

Employing the FFT algorithm may ease our concerns with regard to performance, but we still have to make sure we compute an as good as possible approximation of Equation 3.24. Since our model for surface elevation is based on wave energy, the best way forward seems to be to choose a finite range of wavevectors $\mathbf{k}$ such that they are representative of the whole energy $\mathrm{Var}[\eta]$ as defined by $\Theta(\mathbf{k})$. Because wavevectors $\mathbf{k}$ are related to wavelength, they are also related to the spatial domain, and as such to positions $\mathbf{x}$. Hence, the quality of our approximation depends on how we discretise both the spatial domain into a finite set of positions $\mathbf{x}$ and the wavevector domain into a finite set of wavevectors $\mathbf{k}$.

Domain Discretisation

We define the sea surface we want to synthesize as a rectangular area, with L and K denoting width and height in meters, respectively. Moreover, we discretise said rectangle into a grid with horizontal resolution N and vertical resolution M, where both N and M shall be an integer which can be represented as a power of two. Therefore we will define sea surface elevation at $N \times M$ discrete points. We will denote the horizontal and vertical distance between consecutive grid points as follows:

$$\Delta x = \frac{L}{N} \qquad\qquad \Delta z = \frac{K}{M}$$

We compute surface elevation at the following points:

$$\mathbf{x} = (x, z) \ \in \ \{(\alpha\Delta x, \beta\Delta y) | -\frac{N}{2} \leq \alpha < \frac{N}{2}, -\frac{M}{2} < \beta \leq \frac{M}{2}\} \tag{3.26}$$

where α and β are integers. At first glance the choice of interval for positions $\mathbf{x}$ may seem arbitrary, but it will allow us to make a direct connection to the wavevector domain later on. To simplify matters, we will handle a square area instead of a rectangular one, thus:

$$N = M \qquad\qquad L = K \qquad\qquad \Rightarrow \Delta x = \Delta z$$

Based on the *Nyquist-Shannon sampling theorem* [Bracewell, 2000] we are able to compute the minimum and maximum wavelengths which are reconstructible without loss of information. We may write said minimum and maximum wavelengths as follows:

$$\lambda_{min} = 2\Delta x \qquad\qquad \lambda_{max} = \sqrt{2}\,\Delta x\,\frac{N}{2}$$

Figure 3.3 depicts the spatial domain as defined by resolution and size, including horizontal and vertical grid point distances, as well as minimum and maximum wavelengths.

With λ_{min} and λ_{max} at hand, we may compute the corresponding wavenumbers:

$$k_{min} = \frac{2\pi}{\lambda_{min}} \qquad\qquad k_{max} = \frac{2\pi}{\lambda_{max}} \tag{3.27}$$

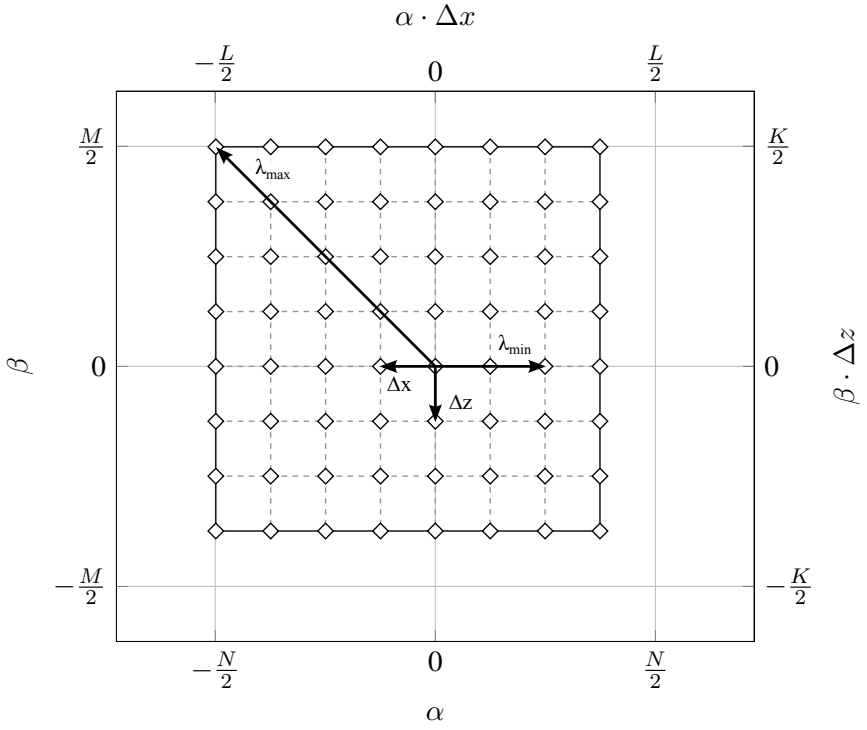


Figure 3.3: The spatial domain in terms of index pairs (α, β) and associated coordinate pairs $(\alpha\Delta x, \beta\Delta z)$.

Analogous to the spatial domain, we have to define the horizontal and vertical distance between consecutive grid points for the wavevector domain. We denote said distances as follows:

$$\Delta k_x = \Delta k_z = \frac{2\pi}{L} = \frac{2\pi}{K} \tag{3.28}$$

Given Δk_x and Δk_z, we will define wavevectors $\mathbf{k}$ in the same fashion as Equation 3.26 defines spatial positions $\mathbf{x}$:

$$\mathbf{k} = (k_x, k_z) \ \in \ \{(\alpha\Delta k_x, \beta\Delta k_z) | -\frac{N}{2} \leq \alpha < \frac{N}{2}, -\frac{M}{2} < \beta \leq \frac{M}{2}\} \tag{3.29}$$

where α and β are integers. We know that the relation between wavelength and wavenumber is an inverse one, while $\|\mathbf{k}\|$ increases, wavelength λ decreases. Therefore we find the larger wavelengths near the domain center, and the smaller ones near the domain border. Figure 3.4 shows the wavevector domain as defined by Equation 3.29, including wavevectors which are exact representatives for wavenumbers k_{min} and k_{max}. As expected, k_{min} is found at the domain border, and k_{max} near the domain center. We may rewrite wavenumbers k_{min} and k_{max} from

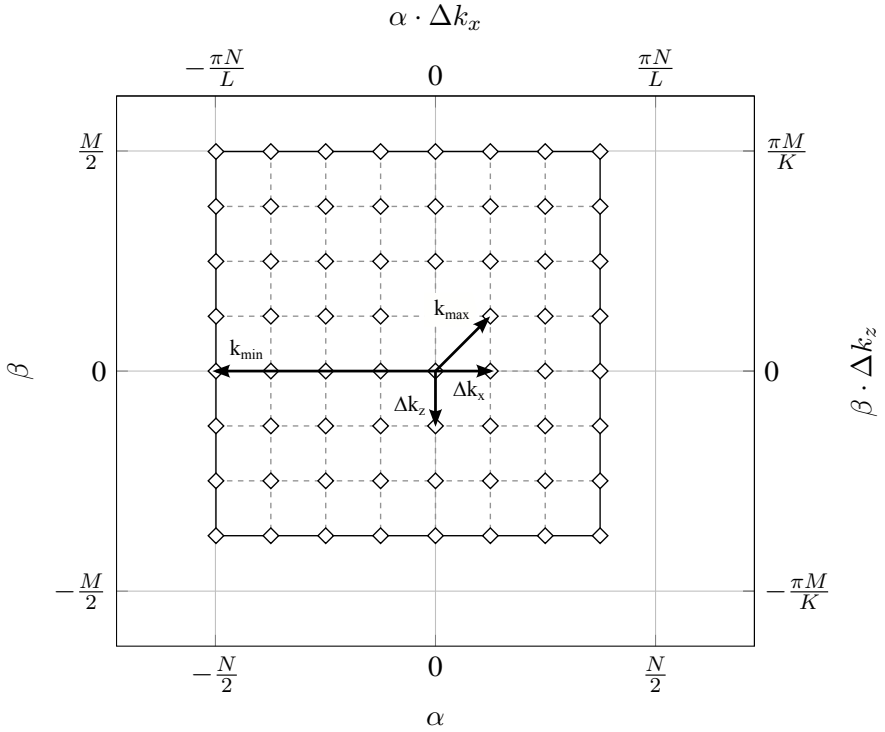


Figure 3.4: The wavevector domain in terms of index pairs (α, β) and associated coordinate pairs $(\alpha\Delta k_x, \beta\Delta k_z)$.

Equation 3.27 to the following:

$$k_{max} = \sqrt{2}\Delta k_x = \sqrt{2}\Delta k_z = \sqrt{2}\frac{2\pi}{L} = \sqrt{2}\frac{2\pi}{K}$$
$$k_{min} = \frac{\pi N}{L} = \frac{\pi M}{K}$$

Hence, by choosing surface dimensions L and K we implicitely determine the maximum wavelength we are able to include in our computation. With surface dimensions L and K fixed, we choose the number of samples N and M, knowing that they will define the smallest wavelength representable by wavevectors $\mathbf{k}$.

Later on we will see that the distribution of wave energy among wavevectors is highly dependent on the actual wave spectrum, as well as on the input parameters to the wave spectrum. Wave energy may be focused in a very narrow range of wavevectors, or be wide spread among a large portion of the entire wavevector spectrum. As a consequence, sometimes it is necessary to choose multiple sets of wavevectors $\mathbf{k}$ which complement one another, and to combine the results in order to get satisfactory results.

Energy Discretisation

We have discretised the wavevector domain into a finite set of wavevectors, where $\pm k_{min}$ represent the largest positive and negative wavenumbers on the horizontal and vertical axes. Since we seek a representative approximation of whole wave energy $\mathrm{Var}[\eta]$ based on our finite set of wavevectors, we may start by limiting the respective integral with $\pm k_{min}$:

$$\mathrm{Var}[\eta] = \iint_{\mathbf{k}} \Theta(\mathbf{k})\ \mathrm{d}\mathbf{k} \approx \int_{-k_{min}}^{k_{min}} \int_{-k_{min}}^{k_{min}} \Theta(\mathbf{k})\ \mathrm{d}\mathbf{k} \tag{3.30}$$

We may rewrite the approximation in Equation 3.30 as a sum of integrals:

$$\mathrm{Var}[\eta] = \int_{-k_{min}}^{k_{min}} \int_{-k_{min}}^{k_{min}} \Theta(\mathbf{k})\ \mathrm{d}\mathbf{k} = \sum_{\alpha} \sum_{\beta} \int_{\Delta k_x} \int_{\Delta k_z} \Theta(\mathbf{k})\ \mathrm{d}\mathbf{k} \tag{3.31}$$

where α and β are the indices from Equation 3.29, and Δk_x and Δk_z represent the distances between grid points as defined by Equation 3.28. Alternatively, we may compute an approximation of wave energy in a pure discrete way. Let $\mathbf{k}_{\alpha,\beta}$ be the wavevectors defined by Equation 3.29, then:

$$\mathrm{Var}[\eta] = \sum_{\alpha} \sum_{\beta} \Theta(\mathbf{k}_{\alpha,\beta}) \tag{3.32}$$

By combining Equation 3.31 and 3.32 we get:

$$\Theta(\mathbf{k}_{\alpha,\beta}) = \int_{\Delta k_x} \int_{\Delta k_z} \Theta(\mathbf{k})\ \mathrm{d}\mathbf{k} \approx \Theta(\mathbf{k}_{\alpha,\beta}) \Delta k_x \Delta k_z \tag{3.33}$$

Now we may write our approximation of the whole wave energy as a simple sum:

$$\mathrm{Var}[\eta] = \sum_{\alpha} \sum_{\beta} \Theta(\mathbf{k}_{\alpha,\beta}) \Delta k_x \Delta k_z \tag{3.34}$$

Random Amplitudes

Recall that wave energy is defined as the mean square of the wave amplitude, so, given wave energy $\Theta(\mathbf{k})$, we may compute wave amplitude $B(\mathbf{k})$ as follows:

$$B(\mathbf{k})^2 = 2\Theta(\mathbf{k})$$

We may convert to a discrete formulation based on Equation 3.34:

$$B(\mathbf{k}_{\alpha,\beta})^2 = 2\Theta(\mathbf{k}_{\alpha,\beta}) \Delta k_x \Delta k_z$$

As described in Section 3.2, in order to generate random wave amplitudes, we employ a Gaussian Distribution with mean $\mathrm{E}[\eta] = 0$ and variance $\mathrm{Var}[\eta]$ defined by the energy spectrum $\Theta(\mathbf{k})$. Since all Gaussian Distributions may be written based on the Standard Normal Distribution, we are able to write surface elevation as follows:

$$\eta(\mathbf{x}, t) = \sum_{\alpha} \sum_{\beta} \frac{1}{\sqrt{2}} (\xi_r + \mathrm{i}\xi_i) \sqrt{B(\mathbf{k}_{\alpha,\beta})^2}\ \mathrm{e}^{-\mathrm{i}\omega(\mathbf{k}_{\alpha,\beta})t}\ \mathrm{e}^{\mathrm{i}\mathbf{k}_{\alpha,\beta}^{\mathsf{T}}\mathbf{x}}$$

where ξ_r and ξ_i are random scalars drawn from the Standard Normal Distribution. We will drop indices α and β and give a formulation in terms of wave energy $\Theta(\mathbf{k})$ and our finite set of wavevectors $\mathbf{k}$:

$$\eta(\mathbf{x}, t) = \sum_{\mathbf{k}} \frac{1}{\sqrt{2}}(\xi_r + \mathrm{i}\xi_i)\sqrt{2\Theta(\mathbf{k})\Delta k_x \Delta k_z}\, \mathrm{e}^{-\mathrm{i}\omega(\mathbf{k})t}\, \mathrm{e}^{\mathrm{i}\mathbf{k}^{\mathsf{T}}\mathbf{x}} \tag{3.35}$$

We have found a discrete approximation of surface elevation which is fast to compute and is able to give satisfactory results as long as we choose one or more sets of wavevectors $\mathbf{k}$ wisely. We will now move on and discuss parametric wave energy models, because those constitute the basis for the generation of visually pleasing as well as plausible ocean surfaces.

3.4 Wave Spectra

Oceanographic literature provides us with a large set of different wave spectra to choose from [Komen et al., 1996]. For our implementation, we decided to pick a small number of wave spectrum models which, one, are well established in oceanographic research, and, two, allow us to reproduce a wide variety of ocean surfaces. We will elaborate on the latter point in Chapter 6, where we discuss the visual results obtained by the different wave spectra. For now let us focus on the models themselves. We will start with a pair of one- dimensional frequency spectra most often found and referenced in oceanographic literature. As said frequency spectra are one dimensional, we need to convert them into two-dimensional wavenumber spectra, as only the latter form allows us to actually synthesize a finite area of a virtual ocean surface. Thus, first, we follow up with a description of how to augment one-dimensional frequency spectra with directional information, and second, we show how to make sure to preserve integral equality between different wave spectrum domains, i.e. how to convert from frequency based wave energy to wavenumber based energy with correct scaling. Last, we discuss three more wave spectra, where each of them already includes a distinct model for the directional distribution of wave energy.

Pierson Moskowitz Spectrum

The Pierson Moskowitz spectrum [Pierson and Moskowitz, 1964] is a well known and widely used one-dimensional frequency spectrum based on data aquired by weather ships in the North Atlantic Ocean. At the time it was introduced, it was expressed in terms of wind speed as follows:

$$\Theta(\omega) = \frac{\alpha g^2}{\omega^5}\, \exp\left[-\beta\left(\frac{g}{U_{19.5}}\frac{1}{\omega}\right)^4\right] \tag{3.36}$$

with

$$\alpha = 8.1 \times 10^{-3} \qquad\qquad \beta = 74 \times 10^{-1}$$

where $U_{19.5}$ is the wind speed at a height of 19.5 meters above the sea surface, the height of the anemometers used by Pierson and Moskowitz. Further analysis of the data [Alves et al., 2003] brought to light the subsequent relationship between wind speed and peak frequency ω_p:

$$\beta \left(\frac{g}{U_{19.5}} \right)^4 = \frac{5}{4} \omega_p^4$$

which led to a rephrasing of Equation 3.36 in terms of peak frequency:

$$\Theta(\omega) = \frac{\alpha g^2}{\omega^5} \exp \left[-\frac{5}{4} \left(\frac{\omega_p}{\omega} \right)^4 \right] \tag{3.37}$$

with

$$\omega_p = \sqrt[4]{\beta \frac{4}{5}} \frac{g}{U_{19.5}} \approx 0.877 \frac{g}{U_{19.5}} \tag{3.38}$$

Most spectral models published after the Pierson Moskowitz spectrum use wind speed at ten meters height above the sea surface, U_{10}, as a standard. We may convert between $U_{19.5}$ and U_{10} as follows [Alves et al., 2003]:

$$U_{10} \approx 1.026 \times U_{19.5} \tag{3.39}$$

Now we are able to rewrite peak frequency ω_p in terms of U_{10}:

$$\omega_p \approx 0.855 \frac{g}{U_{10}} \tag{3.40}$$

which completes our conversion of the Pierson Moskowitz model to standard wind speed U_{10}. Figure 3.5 depicts how the Pierson Moskowitz frequency spectrum changes depending on wind speed U_{10}, peak frequency ω_p as well as numeric constant α.

The proper use of the Pierson Moskowitz spectrum is restricted to *fully developed seas*. A fully developed sea is a sea for which the energy input to the waves from the wind is in equilibrium with the transfer of energy among different wave components, and with the dissipation of energy by wave breaking. All wind-generated waves are as large as they can be under the current conditions. Moreover, a fully developed sea is independent of *fetch*, which is the distance over which the wind blows, usually limited by the upwind distance to shore.

Fetch-limited wave growth

Before we move on to dicuss wave spectra which model a *developing sea*, we introduce some common terms shared by all those models. A developing sea is a sea where, given a constant wind blowing over the sea surface, waves may still grow, either based on distance from shore, e.g., fetch, or based on time. We will focus on wave spectra which model *fetch-limited wave growth*, leaving aside models which simulate *duration-limited wave growth*. This is a reasonable choice, because the former has been investigated more thoroughly than the latter, both in the laboratory and the field [Young, 1999].

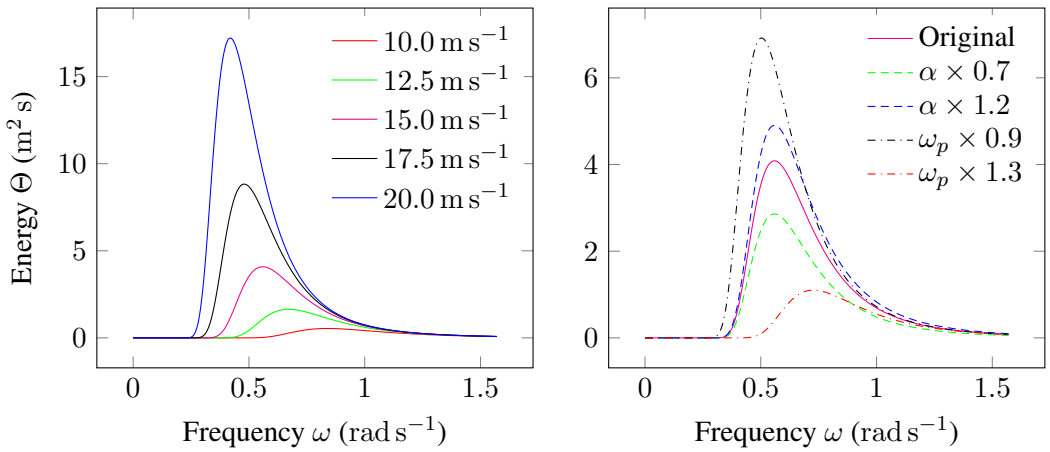


Figure 3.5: Left: The Pierson Moskowitz frequency spectrum with different wind speed values U_{10}. As the wind speed increases, the spectral peak grows and moves to lower frequencies. Right: Influence of the constant α and the peak frequency ω_p on the resulting spectrum ($U_{10} = 15.0\,\mathrm{m\,s^{-1}}$). The parameter ω_p represents the frequency where the spectrum has its peak, while the constant α affects the magnitude of the entire spectrum.

Fetch-limited wave growth occurs when a wind of constant magnitude and direction blows perpendicular to a long and straight coastline. The water is assumed deep and the wind blows for a sufficiently long time for the wave field to reach steady state (independent of time). As a consequence, given wind speed U_{10}, the wave spectrum becomes a function of fetch F.

Sverdrup and Munk [1947] as well as Kitaigorodskii [1962, 1970] identified the following quantities to be of importance for fetch-limited wave growth:

- σ^2 - the variance of the water surface elevation
- U_{10} - the wind speed measured at a reference height of 10 meters
- F - the fetch
- g - the gravitational acceleration
- f_p - the frequency of the spectral peak

Sverdrup, Munk and Kitaigorodskii went on to define the following non-dimensional groupings of said quantities:

$$\epsilon = \frac{\sigma^2 g^2}{U_{10}^4} \quad \text{- the non-dimensional energy} \tag{3.41}$$

$$\nu = \frac{f_p U_{10}}{g} \quad \text{- the non-dimensional peak frequency} \tag{3.42}$$

$$\chi = \frac{gF}{U_{10}^2} \quad \text{- the non-dimensional fetch} \tag{3.43}$$

Since the wave spectrum is assumed to be a function of fetch, the following must hold:

$$\epsilon = f_1(\chi) \qquad\qquad \nu = f_2(\chi) \tag{3.44}$$

where f_1 and f_2 are functions to be determined. Finding instances of these two functions which fit both laboratory and field data has proven to be a key aspect of oceanographic research. We will see that the energy spectra presented throughout this document, which actually originate from oceanographic literature, employ dimensionless variables in combination with functions f_1 and f_2, be it implicitely or explicitely. Note, that we will omit any detailed discussion regarding dimensionless energy ϵ and function f_1, because, contrary to dimensionless fetch ν and function f_2, those are not a prerequisite to be able to actually synthesize the energy spectrum. Function f_2 is of fundamental importance, because, given wind speed and fetch, it allows us to compute peak frequency f_p and consequently peak angular frequency ω_p. By combining Equation 3.42 and Equation 3.44, we may write the peak frequency as follows:

$$f_p = \frac{g\nu}{U_{10}} = \frac{gf_2(\chi)}{U_{10}} \tag{3.45}$$

$$\omega_p = 2\pi f_p \tag{3.46}$$

Growth limits

The first significant attempts to predict fetch-limited waves were ventured by Sverdrup and Munk [Hydrographic Office, 1944, 1945], based on data from several sources, including both field and laboratory. Bretschneider [1952] augmented said data set and refined the results. The combined findings of Sverdrup, Munk and Bretschneider, a set of non-dimensional growth curves, are known as the *SMB growth curves* [Coastal Engineering Research Center, 1977]. These curves show that for large values of dimensionless fetch χ, both dimensionless energy ϵ and dimensionless peak frequency ν reach asymptotic limits. These limits indicate that all further development ceases at that point, the sea has reached a fully developed state. We may write said limits as follows:

$$\epsilon = 5 \times 10^{-3} \quad \text{for large } \chi$$
$$\nu = 0.133 \quad \text{for large } \chi$$

Pierson and Moskowitz, on the other hand, provide the limits of dimensionless energy and dimensionless peak frequency based on data retrieved solely from fully developed seas. These limits are as follows:

$$\epsilon = 3.64 \times 10^{-3} \tag{3.47}$$

$$\nu = 0.13 \tag{3.48}$$

where ϵ and ν are independent of fetch. We may compute the peak angular frequency of the Pierson Moskowitz spectrum by combining Equations 3.45, 3.46 and 3.48 as follows:

$$\omega_p = 2\pi \frac{0.13g}{U_{10}}$$

which matches the Pierson Moskowitz definition of peak angular frequency given earlier in Equation 3.40.

Based on peak angular frequency ω_p and a constant wind of speed U_{10}, we may introduce the *inverse wave age*, Ω, which describes the state of development of a sea, and is defined as follows:

$$\Omega = \frac{U_{10}}{c_p} \tag{3.49}$$

where c_p is the phase speed at peak angular frequency ω_p. Recall that in deep water, the phase speed is g/ω, therefore $c_p = g/\omega_p$. By combining Equations 3.45, 3.46 and 3.49, we are able to rewrite the inverse wave age in deep water in terms of dimensionless peak frequency:

$$\Omega = \frac{U_{10}}{c_p} = \frac{U_{10}}{\frac{g}{\omega_p}} = 2\pi\nu \tag{3.50}$$

The Pierson Moskowitz spectrum has a constant inverse wave age, $\Omega \approx 0.82$, which represents the lower limit of Ω, reached only by a fully developed sea. The upper limit of the inverse wave age for a spectrum which simulates fetch-based wave growth is usually set at $\Omega = 4$ or $\Omega = 5$. Since the lower limit of the inverse wave age is actually less than one, we conclude that at full development, the phase speed of the waves may actually exceed the speed of the generating wind.

JONSWAP Spectrum

The JONSWAP spectrum is a one-dimensional frequency spectrum which has been developed by Hasselman et al. [1973] based on data collected during the *Joint North Sea Wave Project*. The JONSWAP spectrum simulates fetch based wave growth and is expressed as a combination of the Pierson Moskowitz spectrum and an additional *peak enhancement factor*. A compact form of the JONSWAP spectrum often found in literature looks as follows:

$$\Theta(\omega) = \frac{\alpha g^2}{\omega^5}\ \exp\left[-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4\right]\gamma^r \tag{3.51}$$

with

$$r = \exp\left[-\frac{\left(\omega-\omega_p\right)^2}{2\sigma^2\omega_p^2}\right]$$

$$\alpha = 0.076\left(\frac{gF}{U_{10}^2}\right)^{-0.22}$$

$$\omega_p = 22\left(\frac{U_{10}F}{g^2}\right)^{-0.33}$$

$$\sigma = \begin{cases} 0.07 & \omega \leq \omega_p \\ 0.09 & \omega > \omega_p \end{cases}$$

$$\gamma = 3.3$$

where F denotes the fetch in meters, and U_{10} the wind speed in meters per second at a height of ten meters above the sea surface. Moreover, γ^r is called the peak enhancement factor, σ the *peak width parameter*, α the *equilibrium range parameter* and ω_p the angular frequency of the spectral peak.

In contrast to the Pierson Moskowitz spectrum, which exclusively models fully developed seas, the JONSWAP spectrum solely models developing seas. Hasselman et al. state that in context of their model the sea is never fully developed, the spectrum does not converge to the Pierson Moskowitz spectrum as fetch increases. Figure 3.6 gives a tentative overview of the development of a JONSWAP modeled sea, including a sketch which shows the impact of different parameters on the resulting spectrum. One may notice that the spectral peak of the JONSWAP spectrum may take a more narrow form than the already pronounced peak of the Pierson Moskowitz spectrum. Moreover, Figure 3.7 depicts the inabilty of the the JONSWAP spectrum to converge to the Pierson Moskowitz spectrum, be it with decreasing inverse wave age or with increasing fetch.

Although Equation 3.51 is a compact representation of the JONSWAP spectrum, it does omit explicit use of dimensionless variables. The JONSWAP spectrum simulates fetch limited wave growth, therefore it has to specify how dimensionless peak frequency ν is dependent on dimensionless fetch χ. Hasselman et al. provide the following relationship:

$$\nu = 3.5\chi^{-0.33} \tag{3.52}$$

Because Equation 3.52 does not attempt to accomodate a transition to full development, it is not applicable at large values of χ. Based on the JONSWAP datasets, Hasselman et al. confined their model to $\chi < 1 \times 10^4$. Given dimensionless peak frequency ν, based on Equations 3.45, 3.46 and 3.50 we may write peak angular frequency ω_p as follows:

$$\omega_p = 2\pi\frac{g\nu}{U_{10}} = \Omega\frac{g}{U_{10}} \tag{3.53}$$

Moreover, Hasselman et al. also base the equilibrium range parameter α on dimensionless fetch:

$$\alpha = 0.076\chi^{-0.22} \tag{3.54}$$

This completes our conversion of the JONSWAP spectrum to make explicit use of dimensionless peak frequency and dimensionless fetch. Before we move on to other wave spectra, we may clarify how we intend to convert the presented one-dimensional frequency spectra into two-dimensional wavenumber spectra.

Directional Spreading Function

As we have seen in Section 3.2, the one-dimensional frequency spectrum does not hold any positional or directional information, which makes it the easiest one to measure. But in order to be able to synthesize a finite area of a virtual ocean surface, we depend on directional wave information. We are in need of a *directional spreading function*, which directionally distributes the energy provided by $\Theta(\omega)$ as well as filters wave trains which move in the opposite direction of the wind. We may call the product of a normal one-dimensional frequency spectrum and

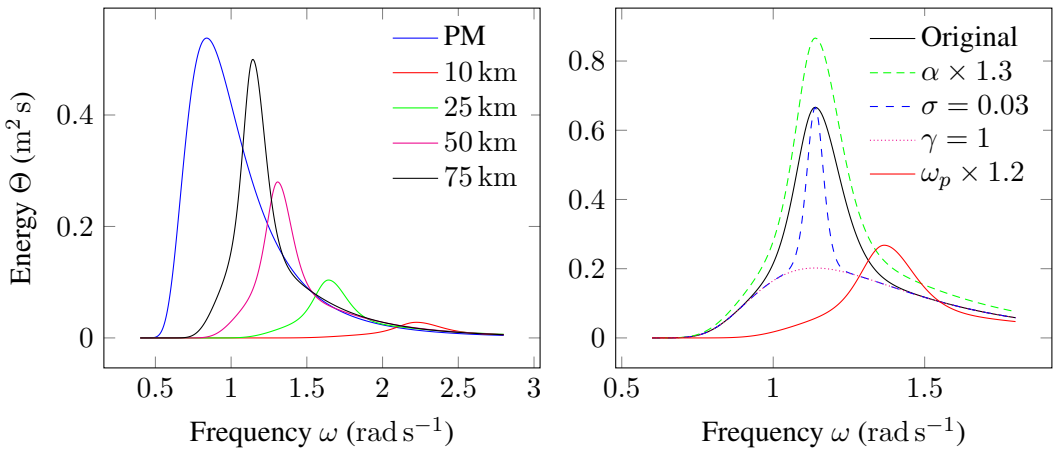


Figure 3.6: Left: JONSWAP frequency spectra with different fetch values F, Pierson Moskowitz spectrum for comparison ($U_{10} = 10\,\mathrm{m\,s^{-1}}$). Right: Influence of the involved parameters on the resulting JONSWAP spectrum ($U_{10} = 15\,\mathrm{m\,s^{-1}}$, $F = 50\,\mathrm{km}$).

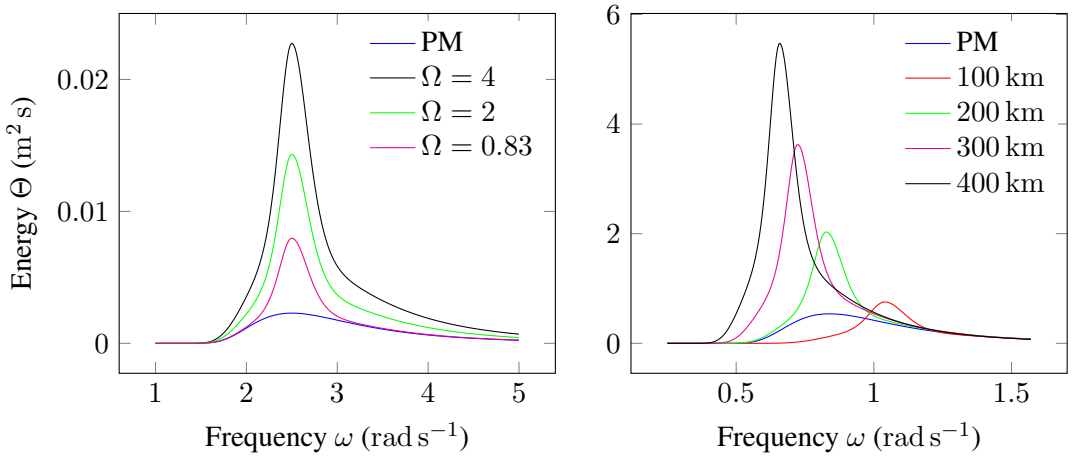


Figure 3.7: The JONSWAP spectrum does not converge towards a fully-developed sea as represented by the Pierson Moskowitz spectrum, neither based on inverse wave age, nor based on fetch. Left: Influence of inverse wave age Ω on the resulting JONSWAP spectrum, Pierson Moskowitz spectrum for comparison ($\omega_p = 2.5$). Right: JONSWAP frequency spectra with large fetch values F, Pierson Moskowitz spectrum for comparison ($U_{10} = 10\,\mathrm{m\,s^{-1}}$).

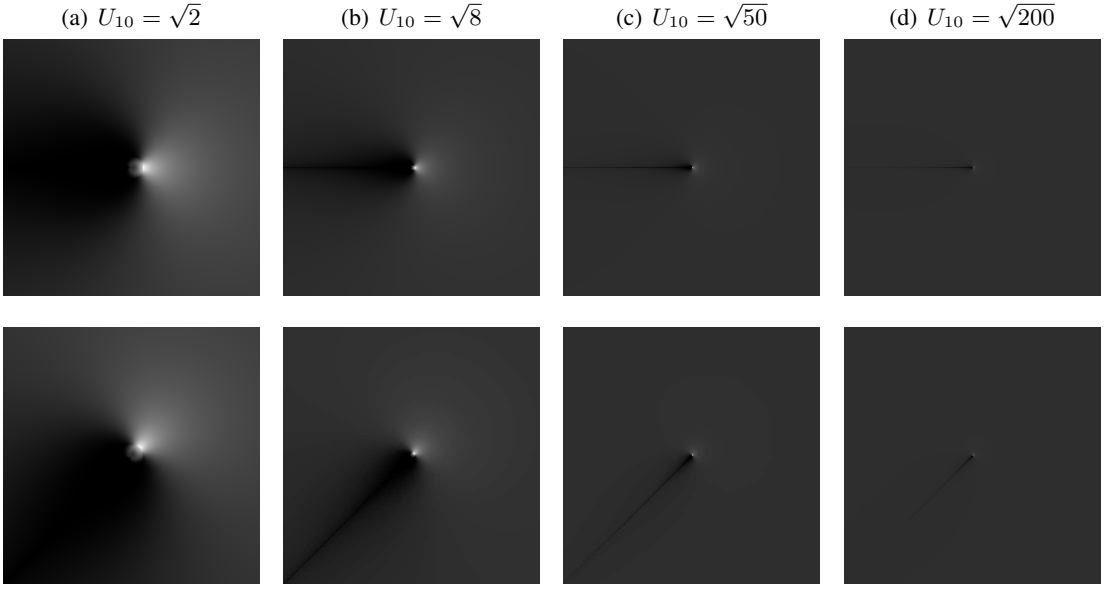


Figure 3.8: A set of example instances of the directional spreading function introduced by Mitsuyasu et al. [1975] and extended by Hasselmann et al. [1980]. Each column corresponds to a different wind speed, starting with the weakest one at the left. In the top row, the wind direction points to the right, in the bottom row to the upper right.

a directional spreading function a *directional frequency spectrum*. We may write a directional frequency spectrum as follows:

$$\Theta(\omega, \theta) = \Theta(\omega) D(\omega, \theta) \tag{3.55}$$

with

$$\int_{\theta=-\pi}^{\pi} D(\omega, \theta)\, \mathrm{d}\theta = 1 \qquad\qquad D(\omega, \theta) \geq 0 \tag{3.56}$$

where θ is the direction of the wave. Equation 3.56 makes sure the amount of energy per frequency does not change. Therefore the following relation holds:

$$\Theta(\omega) = \int_{\theta=-\pi}^{\pi} \Theta(\omega, \theta)\, \mathrm{d}\theta$$

Literature provides us with a large set of different directional spreading functions to choose from [Young, 1999]. We picked a *Cosine-2s* variant based on the work of Mitsuyasu et al. [1975] and Hasselmann et al. [1980]. Expressed in terms of angular frequency ω it is defined as follows:

$$D(\omega, \theta) = \frac{2^{2s-1}}{\pi} \frac{\Gamma^2(s+1)}{\Gamma(2s+1)} \left| \cos\left(\frac{\theta - \theta_p}{2}\right) \right|^{2s} \tag{3.57}$$

with

$$\frac{s}{s_p} = \begin{cases} (\frac{\omega}{\omega_p})^{-2.5} & \omega \geq \omega_p \\ (\frac{\omega}{\omega_p})^{5} & \omega < \omega_p \end{cases}$$

where, according to Hasselmann et al. [1980]

$$s_p = \begin{cases} 9.77 & \omega \geq \omega_p \\ 6.97 & \omega < \omega_p \end{cases}$$

Γ is the *gamma function*, s is called the *spreading factor*, ω_p represents the angular frequency the frequency spectrum has its peak at, and θ_p is the direction of the waves at said peak. It is assumed that θ_p is equal the wind direction. Figure 3.8 depicts example instances of Equation 3.57.

Integral Domain Conversion

Recall that we transform from the wavenumber domain to the spatial one. Thus, as a prerequisite, we are in need of a two-dimensional wavenumber spectrum $\Theta(\mathbf{k})$, but Equation 3.55 gives us a directional frequency spectrum $\Theta(\omega, \theta)$. We will convert one spectrum form to the other according to the change of variables theorem, with the deep water dispersion relation $\omega^2 = gk$ serving as a link between angular frequency ω and wavenumber k. To avoid possible confusion we will add subscripts to each occurence of Θ in order to explicitely show the parameters it takes as input. As a first step we may transform the one-dimensional frequency spectrum $\Theta_\omega(\omega)$ to the one-dimensional wavenumber spectrum $\Theta_k(k)$ as follows:

$$\int \Theta_k(k)\,\mathrm{d}k = \int \Theta_\omega(\omega)\,\mathrm{d}\omega = \int \Theta_\omega(\omega(k))\frac{\mathrm{d}\omega}{\mathrm{d}k}\,\mathrm{d}k = \int \Theta_\omega(\sqrt{gk})\frac{1}{2}\sqrt{\frac{g}{k}}\,\mathrm{d}k$$

which gives us

$$\Theta_k(k) = \Theta_\omega(\sqrt{gk})\frac{1}{2}\sqrt{\frac{g}{k}} \tag{3.58}$$

Given $\Theta_k(k)$, we are able to rewrite the directional frequency spectrum $\Theta_{\omega,\theta}(\omega, \theta)$ from Equation 3.55 in terms of wavenumber k:

$$\Theta_{k,\theta}(k, \theta) = \Theta_k(k)D(\sqrt{gk}, \theta) \tag{3.59}$$

As the last step we need to convert from polar coordinate pairs (k, θ) to Cartesian wavenumber coordinates $\mathbf{k}$. Said conversion is accomplished as follows:

$$\int \Theta_{\mathbf{k}}(\mathbf{k})\,\mathrm{d}\mathbf{k} = \iint \Theta_{k,\theta}(k, \theta)\frac{1}{k}\,\mathrm{d}\theta\,\mathrm{d}k \tag{3.60}$$

By combining Equations 3.55, 3.58 and 3.60 we get the two-dimensional wavenumber spectrum:

$$\Theta_{\mathbf{k}}(\mathbf{k}) = \Theta_{\omega,\theta}(\sqrt{gk}, \theta)\frac{1}{2k}\sqrt{\frac{g}{k}} \qquad \theta = \arctan\left(\frac{k_z}{k_x}\right) \tag{3.61}$$

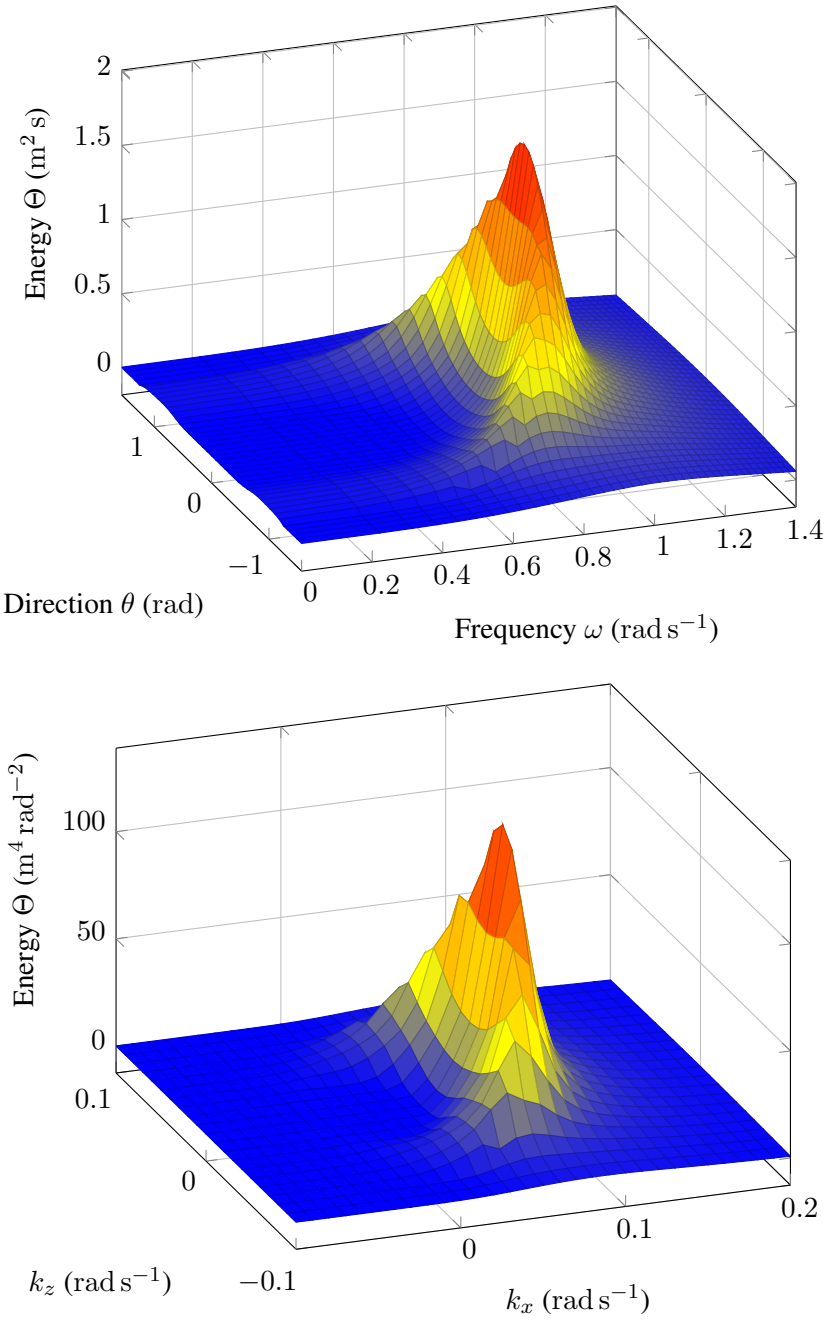


Figure 3.9: The JONSWAP frequency spectrum with wind speed $U_{10} = 15\,\mathrm{m\,s^{-1}}$ and fetch $F = 100\,\mathrm{km}$, located in two different domains. Top: The polar coordinate domain $\omega \times \theta$, the wind direction points in the direction of the positive ω axis. Bottom: The wavevector domain $k_x \times k_z$, the wind direction points in the direction of the positive k_x axis. Note the difference in scale between the two vertical axes.

An instance of a directional frequency spectrum as well as its equivalent directional wavenumber spectrum are depicted in Figure 3.9.

Now we are able to preserve integral equality between different representations of Θ, as long as we use the deep-water dispersion relation. One may do the math for another dispersion relation, given that we have laid out the path one has to take. Still, in the context of this work, we only concern ourselves with waves in deep water.

Two-Dimensional Wave Spectra

Beforehand we introduced the combination of a one-dimensional frequency spectrum with a directional spreading function in order to generate a two-dimensional frequency spectrum. We have seen that there are different forms of two-dimensional wave spectra, such as the directional frequency spectrum $\Theta(\omega, \theta)$, the wavenumber spectrum in polar form $\Theta(k, \theta)$, and the wavenumber spectrum in Cartesian form $\Theta(\mathbf{k})$. Moreover, given a dispersion relation, we are able to convert between all these different forms of a spectrum. Now, we may move on to wave spectra which have been explicitly introduced in a two-dimensional form by oceanographic research.

Donelan Spectrum

The Donelan spectrum is a directional frequency spectrum which has been developed by Donelan et al. [1985] based on data collected both on Lake Ontario and at the *Canadian Centre for Inland Waters*'s wind-wave flume. At its core, the Donelan spectrum still is a one-dimensional frequency spectrum combined with a directional spreading function, but it improves upon certain key aspects compared to the models discussed beforehand. The most prominent of said improvements is the ability of the Donelan spectrum to model the transition between developing and fully developed seas. Other improvements include the possibility for wind direction and peak wave direction to differ, as well as a directional spreading function which both matches existing data more closely and is simpler in formulation than the one introduced by Mitsuyasu et al. [1975].

The energy given by the one-dimensional Donelan frequency spectrum depends on the frequency itself, the position of that frequency in the spectrum (ω/ω_p) and on the intensity of wind input $\Omega_c = U_c/c_p$, where U_c is the component of the wind in the direction of travel of the waves at the spectral peak, and c_p is the phase velocity of the waves at the spectral peak. The directional spreading, on the other hand, is related only to ω/ω_p. According to Donelan et al., the wind component U_c is defined based on wind speed U_{10}:

$$U_c = U_{10}\cos(\theta_w - \theta_p) \qquad \left|\theta_w - \theta_p\right| \leq 15^\circ \tag{3.62}$$

where θ_w is the direction of wind speed U_{10}, and θ_p represents the direction of travel of the waves at the spectral peak. We may move on to the directional Donelan frequency spectrum, which is as follows:

$$\Theta(\omega, \theta) = \frac{1}{2}\Theta(\omega)\beta\,\mathrm{sech}^2\{\beta[\theta - \theta_p]\} \tag{3.63}$$

From the equation one may see that the directional spread is the following:

$$D(\omega, \theta) = \frac{1}{2}\beta\,\mathrm{sech}^2\{\beta[\theta - \theta_p]\} \tag{3.64}$$

where sech is the *hyperbolic secant function*. The width of the directional spread is determined by the parameter β, which is defined as:

$$\beta = \begin{cases} 2.61(\omega/\omega_p)^{1.3} & 0.56 < \omega/\omega_p < 0.95 \\ 2.28(\omega/\omega_p)^{-1.3} & 0.95 \leq \omega/\omega_p < 1.6 \\ 1.24 & \text{otherwise} \end{cases} \tag{3.65}$$

Given wind input U_c, and the inverse wave age Ω_c based on the component of the wind in the direction of the peak waves, then, assuming deep water, we may write peak angular frequency ω_p in step with Equation 3.50:

$$\omega_p = \Omega_c \frac{g}{U_c} \tag{3.66}$$

Donelan et al. defines inverse wave age Ω_c based on dimensionless fetch χ as follows:

$$\Omega_c = 11.6\chi^{-0.23} \qquad \chi = \frac{gF}{U_c^2} \tag{3.67}$$

where the inverse wave age is restricted to the interval $\Omega_c \in (0.83, 5)$, and F denotes the fetch in meters. The Donelan directional spreading function from Equation 3.64 does not only differ in form from the one proposed by Mitsuyasu et al. [1975] and Hasselmann et al. [1980], but also in sensitivity, or rather insensitivity, to wind speed, see Figure 3.10. That is because the Donelan formulation in Equation 3.65 employs the ratio ω/ω_p, whereas Mitsuyasu et al. and Hasselmann et al. use comparisons with absolute values.

With the directional spread covered, we move on to the one-dimensional frequency spectrum as described by Donelan et al. [1985]. We may write:

$$\Theta(\omega) = \alpha g^2 \omega^{-5} (\omega/\omega_p) \exp\left[-\left(\frac{\omega_p}{\omega}\right)^4\right] \gamma^r \tag{3.68}$$

with

$$\gamma = \begin{cases} 1.7 & 0.83 < \Omega_c < 1 \\ 1.7 + 6.0 \lg(\Omega_c) & 1 \leq \Omega_c < 5 \end{cases}$$

$$r = \exp\left[-\frac{\left(\omega - \omega_p\right)^2}{2\sigma^2\omega_p^2}\right]$$

$$\sigma = 0.08\left(1 + \frac{4}{\Omega_c^3}\right)$$

$$\alpha = 0.006\,\Omega_c^{0.55}$$

where, in lockstep with the JONSWAP spectrum, γ^r is called the peak enhancement factor, σ the peak width parameter, and α the equilibrium range parameter. In contrast to the JONSWAP spectrum introduced earlier, the one-dimensional frequency spectrum as given by Donelan et al. is able to transition to fully developed state. As one can see in Figure 3.11, said transition does work if inverse wave age Ω_c is given explicitely, instead of being estimated based on fetch. We may conclude that Equation 3.67 is not that good an approximation of inverse wave age Ω_c, especially because it does not ensure a lower limit of $\Omega_c = 0.83$ for large fetch.

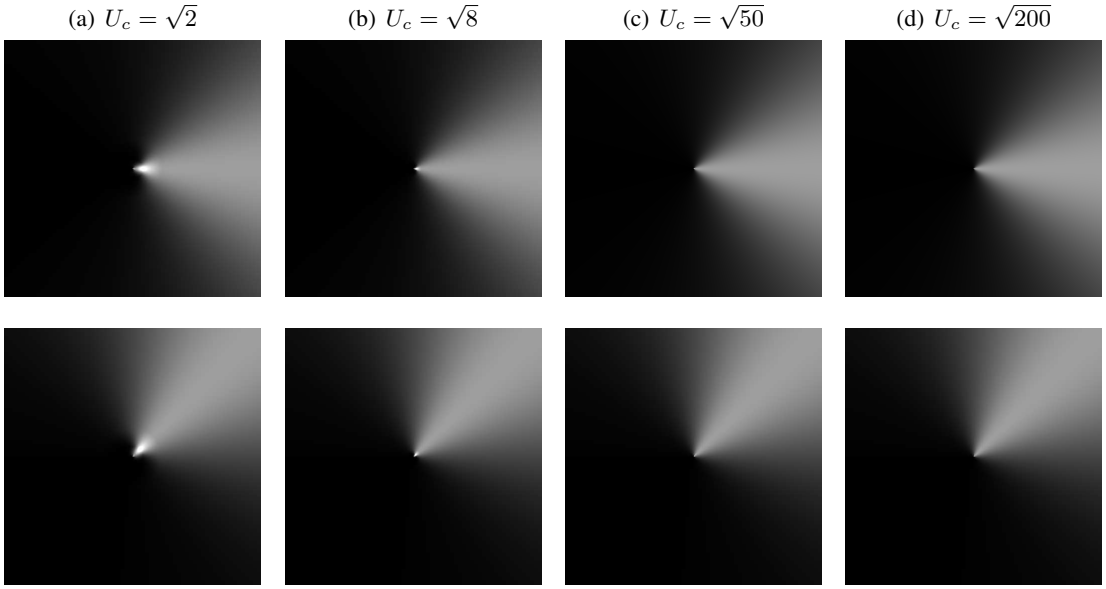


Figure 3.10: A set of instances of the directional spreading function introduced by Donelan et al. [1985]. Each column corresponds to a different wind speed, starting with the weakest one at the left. In the top row, the wind direction points to the right, in the bottom row to the upper right. One can see that the directional spread changes only slightly in intensity and form with increasing wind speed U_c.

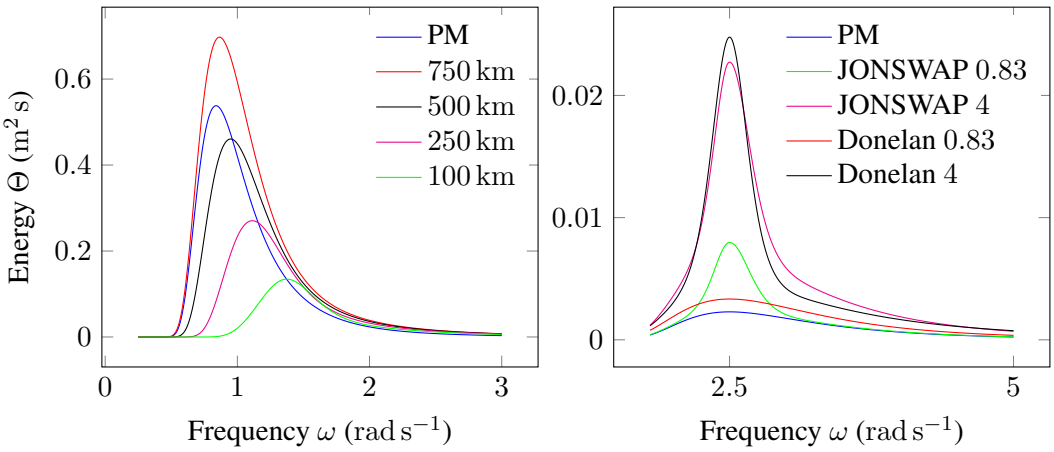


Figure 3.11: Left: Donelan frequency spectra with different fetch values F, Pierson Moskowitz spectrum for comparison ($U_{10} = 10\,\mathrm{m\,s^{-1}}$). The Donelan spectrum converges towards the Pierson Moskowitz spectrum with increasing fetch, but overshoots. Right: JONSWAP and Donelan spectra with different inverse wave age values Ω_c, Pierson Moskowitz spectrum for comparison ($\omega_p = 2.5$). Given specific values for Ω_c, the Donelan spectrum transitions well between a developing and a fully developed sea.

Unified Spectrum

The Unified spectrum is a directional wavenumber spectrum introduced by Elfouhaily et al. [1997]. It consists of a one-dimensional wavenumber spectrum combined with a directional spreading function. All of the spectral models which we introduced beforehand focus on the *long-wave regime*, which, according to Elfouhaily et al., is defined as the wavenumber range $k < 10 \times k_p$, where k_p is the wavenumber the spectrum has its peak at. The Unified spectrum, on the other hand, models the *entire* wavenumber range, including the *short-wave regime*, which is defined as $k >= 10 \times k_p$. Because the short-wave regime extends into the gravity-capillary wave domain, we need a dispersion relation which extends into said domain too. We may write:

$$\omega^2 = gk\left[1 + \left(\frac{k}{k_m}\right)^2\right] \qquad k_m = \sqrt{\frac{p_w g}{T}} = 370\,\mathrm{rad\,m^{-1}}$$

where k_m is the wavenumber at which gravity-capillary waves on an air-water interface have their minimum phase velocity [Lamb, 1945]. The constants p_w, g and T are water density, acceleration due to earth's gravity, and water surface tension, respectively.

The combined works of Hasselman et al. [1973], Mitsuyasu et al. [1975] and Donelan et al. [1985] serve as foundation for the Unified spectrum. Moreover, Elfouhaily et al. did expand their model to the short-wave regime by incorporating related theory [Kitaigorodskii, 1970, Phillips, 1985] and observations [Cox and Munk, 1954, Hara et al., 1994, Jähne and Riemer, 1990]. We may write the Unified spectrum as follows:

$$\Theta(k,\theta) = \frac{1}{2\pi}\Theta(k)\{1 + \Delta(k) + \cos[2(\theta - \theta_p)]\} \tag{3.69}$$

where θ_p is the direction of travel of the waves at the spectral peak. Since the Unified spectrum is given in terms of wavenumber, we are in need of a wavenumber-based formulation of its foundation, namely the Pierson Moskowitz spectrum [Pierson and Moskowitz, 1964], as well as the peak enhancement factor introduced by Hasselman et al. [1973]. Elfouhaily et al. give the wavenumber formulation of the Pierson Moskowitz spectrum as follows:

$$L_{PM}(k) = \exp\left[-\frac{5}{4}\left(\frac{k_p}{k}\right)^2\right]$$

where k_p denotes the wavenumber the spectrum has its peak at. According to Elfouhaily et al. it is defined as:

$$k_p = g\frac{\Omega_c^2}{U_{10}^2}$$

where U_{10} represents the wind speed in meters per second at a height of ten meters above the sea surface, and Ω_c denotes the inverse wave age based on the component of the wind in the direction of the peak waves. The latter term is given by Elfouhaily et al. as follows:

$$\Omega_c = 0.84\tanh\left[\left(\frac{\chi}{\chi_0}\right)^{0.4}\right]^{-0.75} \qquad \chi_0 = 2.2 \times 10^4 \tag{3.70}$$

where χ is the dimensionless fetch. For simplicity's sake we will replicate the definition of dimensionless fetch χ from before:

$$\chi = \frac{gF}{U_{10}^2}$$

with F representing the dimensional fetch in meters. According to Elfouhaily et al., the inverse wave age should be restricted to the range $\Omega_c \in [0.84, 5)$. From Equation 3.70 one may see that Ω_c is guaranteed a lower limit of 0.84 for large fetch, i.e., fully developed seas. It follows that the Unified spectrum is not prone to overdevelopment with increasing fetch, which is a notable improvement compared to the spectral models of Hasselman et al. [1973] and Donelan et al. [1985] (Figure 3.12).

Elfouhaily et al. give the wavenumber formulation of the peak enhancement factor as follows:

$$J_p = \gamma^{\Gamma}$$

where base γ depends only on inverse wave age Ω_c, and exponent Γ requires peak wavenumber k_p and the peak width parameter σ. We may write:

$$\gamma = \begin{cases} 1.7 & 0.84 \leq \Omega_c < 1 \\ 1.7 + 6.0 \lg(\Omega_c) & 1 \leq \Omega_c < 5 \end{cases}$$

$$\Gamma = \exp\left[-\frac{\left(\sqrt{\frac{k}{k_p}} - 1\right)^2}{2\sigma^2} \right]$$

$$\sigma = 0.08 \left(1 + \frac{4}{\Omega_c^3}\right)$$

With the basic terms covered, we are able to move on to the one-dimensional wavenumber spectrum as given by Elfouhaily et al. [1997]:

$$\Theta(k) = k^{-3}(B_l + B_h) \tag{3.71}$$

where B_l represents the low frequency regime, and B_h the high frequency regime respectively. The long wave regime input B_l is defined as:

$$B_l = \frac{1}{2}\alpha_p \frac{c_p}{c} F_p$$

where $c = \omega/k$ is the wave phase speed, and c_p is the phase speed at the spectral peak. The equilibrium range parameter for long waves, α_p, is given as follows:

$$\alpha_p = 0.006\sqrt{\Omega_c}$$

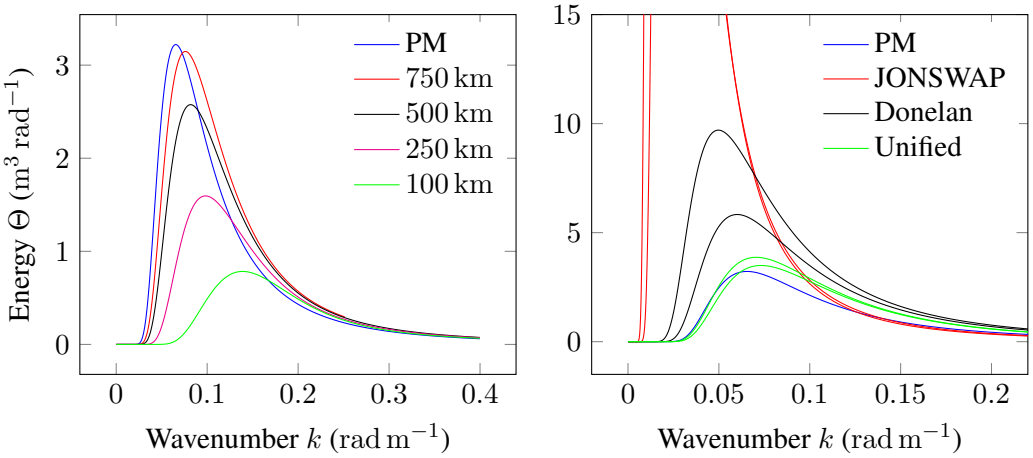


Figure 3.12: Left: Unified frequency spectra with different fetch values F, Pierson Moskowitz spectrum for comparison ($U_{10} = 10\,\mathrm{m\,s^{-1}}$). The Unified spectrum converges towards the Pierson Moskowitz spectrum with increasing fetch. Right: Fetch limited spectral models, each with a fetch of 1000 km and 1500 km, Pierson Moskowitz spectrum for comparison ($U_{10} = 10\,\mathrm{m\,s^{-1}}$). In contrast to the JONSWAP and Donelan spectra, the Unified spectrum is not prone to overdevelopment given large fetch values.

The *long wave side effect function* F_p, which restricts the energy contribution of the B_l term to the low frequency regime, is written as:

$$F_p = L_{PM} J_p \exp\left[-\frac{\Omega_c}{\sqrt{10}}\left(\sqrt{\frac{k}{k_p}} - 1\right)\right]$$

Similar in form to the long wave regime input, the short wave regime input B_h is defined as follows:

$$B_h = \frac{1}{2}\alpha_m \frac{c_m}{c} F_m$$

where $c_m = 0.23\,\mathrm{m\,s^{-1}}$ is the minimum phase speed at wavenumber $k_m = 370\,\mathrm{rad\,m^{-1}}$ [Lamb, 1945]. The *short wave side effect function* F_m accounts for the bandwidth of gravity-capillary waves as well as for a smooth cutoff of energy beyond the gravity-capillary frequency band. We may write:

$$F_m = L_{PM} J_p \exp\left[-\frac{1}{4}\left(\frac{k}{k_m} - 1\right)^2\right]$$

The equilibrium range parameter for short waves, α_m, is given as:

$$\alpha_m = 10^{-2} \begin{cases} 1 + \ln\left(\frac{u^*}{c_m}\right) & u^* < c_m \\ 1 + 3\ln\left(\frac{u^*}{c_m}\right) & u^* \geq c_m \end{cases}$$

where u^* denotes the friction velocity at the water surface. We compute friction velocity through application of the *law of the wall* [von Kármán, 1931]:

$$u^* = \kappa\, U_{10} \left[\ln\left(\frac{10}{z_0}\right)\right]^{-1} \qquad \kappa = 0.41$$

where κ is the *von Kármán constant* [von Kármán, 1931], and z_0 is called the *roughness parameter*. Elfouhaily et al. employ Donelan et al. [1993] to model the roughness parameter as follows:

$$z_0 = 3.7 \times 10^{-5}\, \frac{U_{10}^2}{g} \left(\frac{U_{10}}{c_p}\right)^{0.9}$$

Elfouhaily et al. point out that both, the low frequency and the high frequency wave regime, are modeled based on the air-sea interaction process of friction between wind and waves, u/c, i.e., the *generalized wave age*. Figure 3.13 depicts a comparison between the Unified spectrum and the Donelan spectrum. The scaling is logarithmic to better emphasise the differences in the high frequency regime.

As one may see from Equation 3.69, the Unified spectrum's directional spread is given as follows:

$$D(k,\theta) = \frac{1}{2\pi}\{1 + \Delta(k) + \cos[2(\theta - \theta_p)]\} \tag{3.72}$$

where θ_p is the direction of travel of the waves at the spectral peak. The remaining terms are:

$$\Delta(k) = \tanh\left[a_0 + a_p \left(\frac{c}{c_p}\right)^{2.5} + a_m \left(\frac{c_m}{c}\right)^{2.5}\right]$$

with

$$a_m = 0.13\frac{u^*}{c_m} \qquad a_0 = \frac{\log(2)}{4} \qquad a_p = 4$$

c is the phase speed, c_p is the phase speed at the spectral peak, and c_m is the minimum phase speed at k_m. Moreover, a_0 and a_p are constants, whereas a_m is a function for high-frequency waves dependent on the friction velocity u^* and minimum phase speed c_m. Figure 3.14 depicts example instances of the Unified spectrum's directional spread. One can see that the directional spread is *centrosymmetrical*, in short $D(k,\theta) = D(k,\theta \pm \pi)$. As a consequence, the two-dimensional wavenumber spectrum is centrosymmetrical, too. Therefore, the Unified spectrum satisfies the following conditions:

$$\Theta(k,\theta) = \Theta(k,\theta \pm \pi) \qquad \Theta(\mathbf{k}) = \Theta(-\mathbf{k})$$

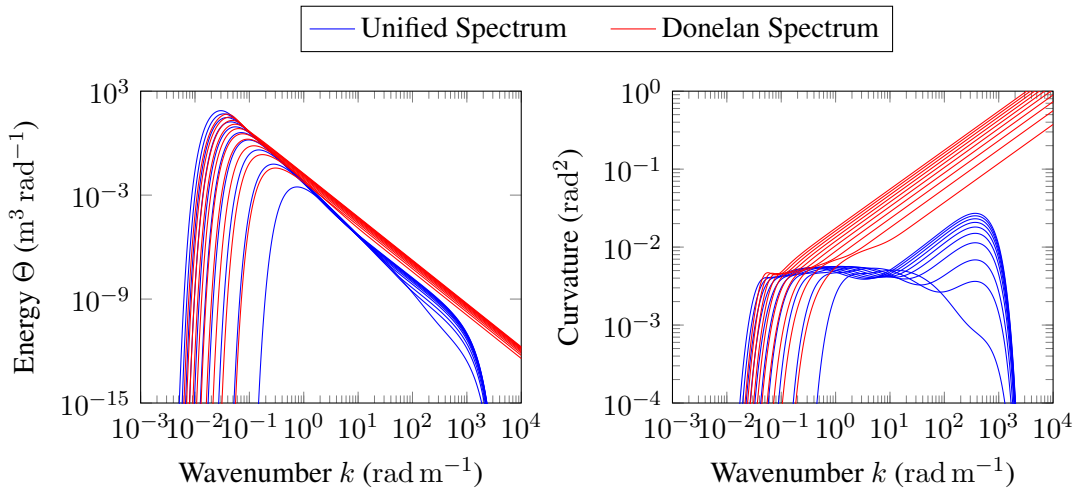


Figure 3.13: Unified spectra and Donelan spectra for the full wavenumber range and for wind speeds from $3\,\mathrm{m\,s^{-1}}$ up to $21\,\mathrm{m\,s^{-1}}$ with a $2\,\mathrm{m\,s^{-1}}$ step ($F = 500\,\mathrm{km}$). Left: One-dimensional wavenumber spectra $\Theta(k)$. The spectra match in the long-wave regime up to the spectral peak, and diverge afterwards. The Unified spectrum does not give energy beyond the gravity-capillary wave band. Right: One-dimensional curvature spectra $C(k) = k^3\Theta(k)$. The difference in shape of the spectra is even more pronounced beyond the primary spectral peak. The Unified spectrum properly models the secondary gravity-capillary peak at $k_m = 370\,\mathrm{rad\,m^{-1}}$.

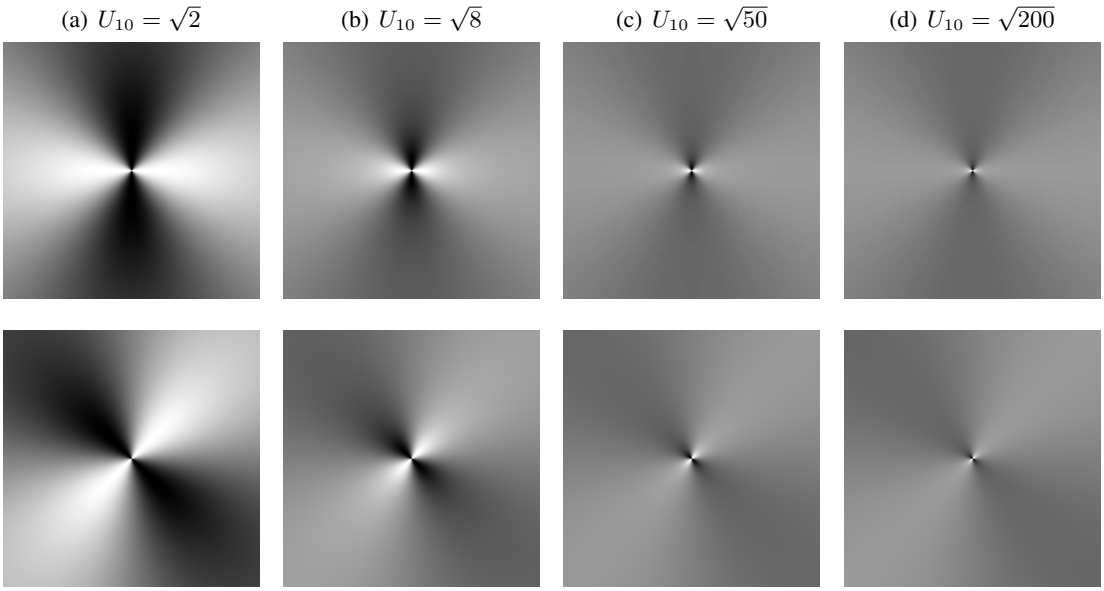


Figure 3.14: A set of instances of the directional spreading function by Elfouhaily et al. [1997]. Each column corresponds to a different wind speed, starting with the weakest one at the left. In the top row, the wind direction points to the right, in the bottom row to the upper right.

Phillips Spectrum

The Phillips Spectrum, as introduced by Tessendorf [1999], is an ad-hoc formulation of a two-dimensional wavenumber spectrum $\Theta(\mathbf{k})$. It has been developed with computer graphics purposes in mind and is loosely based on Phillips [1958, 1985]. We may write Tessendorf's formulation of the Phillips Spectrum as follows:

$$\Theta(\mathbf{k}) = A\frac{\exp\left(-(kL)^{-2} - (kl)^2\right)}{k^4}\left|\mathbf{k}_n^{\mathsf{T}}\mathbf{w}_n\right|^2 \tag{3.73}$$

with

$$\mathbf{k}_n = \frac{\mathbf{k}}{\|\mathbf{k}\|} = \frac{\mathbf{k}}{k} \qquad\qquad \mathbf{w}_n = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

where A and l are numeric constants, and $\mathbf{w}$ encodes wind direction and wind speed $\|\mathbf{w}\|$. The numeric constant A represents a scaling factor which we will discuss at a later point. According to Tessendorf, L defines the largest possible wave arising from a continuous wind of speed $\|\mathbf{w}\|$. The parameter l, on the other hand, acts as a frequency filter, it controls the suppression of energy contributed by high wavenumbers k. In order to accomplish that, it is mandatory for l to be several orders of magnitude smaller than L. The author chose the following as a default value for l:

$$l = 10^{-3}L$$

Moreover, it follows from Equation 3.73 that setting $l = 0$ disables any wavenumber-based suppression of energy. Additionally, one may see that Tessendorf employs a one-dimensional wavenumber spectrum, which is as follows:

$$\Theta(k) = A\frac{\exp\left(-(kL)^{-2} - (kl)^2\right)}{k^4} \tag{3.74}$$

A set of instances of the one-dimensional wavenumber spectrum term are depicted in Figure 3.15. One may see that the shape of the spectrum is similar to the Pierson Moskowitz one in Figure 3.5, with the frequency range of the spectrum peak being located at even lower frequencies than the Pierson Moskowitz and JONSWAP ones. The magnitude of the energy, on the other hand, exceeds the one given by Pierson Moskowitz and JONWAP by at least a factor of 10^3. Recall that both the Pierson Moskowitz and the JONSWAP model are based on real-world data aquired on the ocean, then we may reason that those models give us a value range for energy precise enough to generate believable results. Therefore we need to rescale the energy obtained from the Phillips spectrum into that range. It seemed reasonable to us that this is the purpose of the numeric constant A in Equations 3.73 and 3.74. Since Tessendorf does not give any information about A, choosing a value that scales energy appropriately for all reasonable input parametrisations has proven to be difficult. The constant $\alpha = 8.1 \times 10^{-3}$ as given by Pierson and Moskowitz [1964] (Equation 3.36) seemed to be promising initially, in the end the author settled for the following more adaptive solution:

$$A = \frac{0.81}{LK}B \qquad\qquad B \in [0, 1]$$

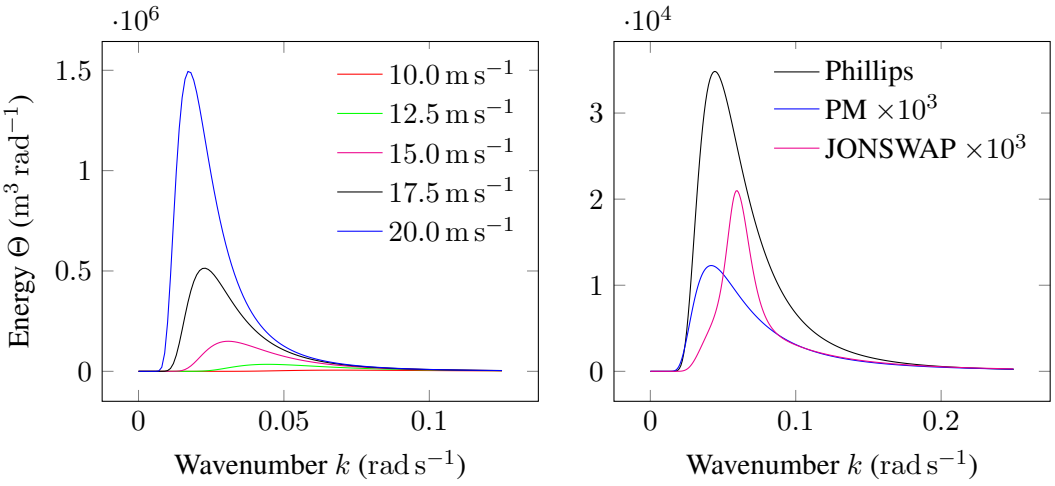
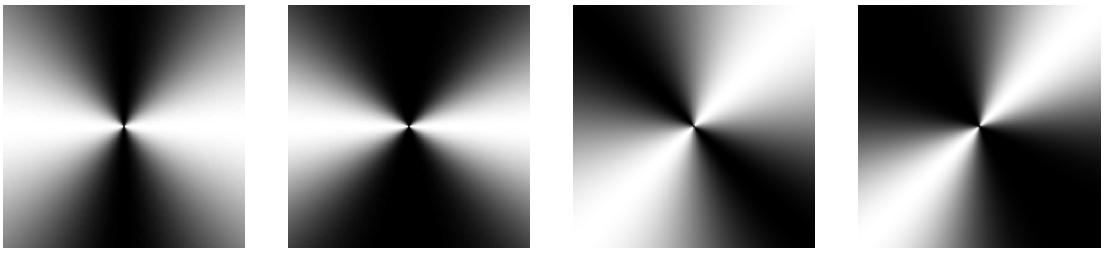


Figure 3.15: Left: The one- dimensional wavenumber spectrum term of the Phillips frequency spectrum with different wind speed values $\|\mathbf{w}\|$. Right: The difference in scale between the Phillips spectrum and both, the Pierson Moskowitz spectrum and the JONSWAP spectrum, amounts to at least a factor of 10^3 ($U_{10} = \|\mathbf{w}\| = 12.5\,\mathrm{m\,s^{-1}}$, $F = 200\,\mathrm{km}$).



(a) $\cos^2$ (b) $\cos^4$ (c) $\cos^2$ (d) $\cos^4$

Figure 3.16: Example instances of the Phillips directional spreading function with exponents 2 and 4. For (a) and (b) the wind direction poins to the right, for (c) and (d) it points to the upper right. The directional spread with an exponent of 4 is distinctly more narrow than the one with an exponent of 2. Moreover, all instances are centrosymmetrical.

where L and K represent side lengths in meters of the rectangular area we want to synthesize.

Now that we have discussed the one-dimensional wavenumber spectrum and the scale factor A, we move on to the directional term from Equation 3.73, which is as follows:

$$D(\mathbf{k}) \;=\; \left|\mathbf{k}_n^{\mathsf{T}}\mathbf{w}_n\right|^2 \;=\; \left|\cos(\theta)\right|^2$$

where θ is the angle between normalized wavevector $\mathbf{k}_n$ and normalized wind direction $\mathbf{w}_n$. We may increase the exponent in order to make the resulting directional spread more narrow. Moreover, as only the absolute value of the cosinus is taken into account, the result is centrosymmetrical. Figure 3.16 gives an illustration of the directional spread with different wind directions and different exponents.

Recall that the directional spread distributes the energy represented by $\Theta(k)$ among all directions, without changing the amount of energy at wavenumber k. Energy is conserved, therefore we may write:

$$\int_{\theta=-\pi}^{\pi} D(k,\theta)\,\mathrm{d}\theta = 1 \qquad\qquad D(k,\theta) \geq 0$$

We insert two variants of the Phillips directional spread into the integral:

$$\int_{\theta=-\pi}^{\pi} \left|\cos(\theta)\right|^2 \,\mathrm{d}\theta = \pi$$
$$\int_{\theta=-\pi}^{\pi} \left|\cos(\theta)\right|^4 \,\mathrm{d}\theta = \frac{3\pi}{4}$$

As one can see, the results are *not* equal one, therefore energy is *not* conserved. In combination with the unusual magnitude of energy output by the one-dimensional Phillips wavenumber spectrum, it is really difficult to find an appropriate scale factor A in order to generate plausible ocean surfaces.

CHAPTER 4

Implementation

The previous chapter discussed the wave spectrum concept, a theoretical framework developed by oceanographic research to describe the distribution of wave energy among ocean surface waves which have been exposed to identical conditions, such as wind speed and distance to shore [Neumann and Pierson Jr, 1966]. Based on wave spectra we are able to synthesize ocean surfaces of great variety: from a perfectly calm sea, over one slightly agitated by the wind, to one with massive waves caused by a storm. But to be able to visualize such diverse seas in a convincing manner we are dependent on more data than surface elevation alone. Therefore, as described by Tessendorf [1999], we may capitalize on the spectral representation of ocean waves to obtain three distinct additional datasets. One, high-quality normal vectors by deriving the ocean surface's slope vectors in frequency space. Two, displacements to transform the gentle form of the sea into a more agitated one, where the waves are more similar to Gerstner waves than to sinusoids. Three, the first-order partial derivatives of aforementioned displacements, as they allow us to deduce the locations on the ocean surface where foam and spray may arise. Each of the three datasets requires us to construct additional spectra, all derived directly from the ocean surface elevation spectrum.

Alongside data synthesis, it is the rendering of a water body as large as the ocean that poses a challenge. The variety of possible camera views may range from closeups of the water surface to vistas where the sea spans all the way to the horizon. In the former case we would like to be able to observe small-scale detail on the water surface, such as ripples caused by gravity-capillary waves. The latter case is not difficult to handle in principle, because the ocean surface data we synthesize is seamlessly tileable. Still, we would prefer that the viewer may not be able to notice that the ocean surface repeats itself in all directions. To tackle both issues at once, we adopt the level-of-detail approach taken by Eric Bruneton [2010] and Dupuy and Bruneton [2012]. We compute a set of ocean surface tiles of differing size, where each tile samples a different part of the wave energy spectrum. By combining the tiles we may extend the sum of waves as far as into the gravity-capillary wave domain. As for the tiling artifacts, even though we are unable to entirely remove periodicity by combing multiple tiles, we are at least able to increase the period to the least common multiple of the different tile sizes.

As one may already guess, we are burdened with a significant number of Fourier Transforms, based on both the number of spectra per tile (surface elevation, slopes, displacements, first order derivatives of the displacements), and the number of different tiles. Thus, we need to make sure to keep the computational workload on an acceptable level for real-time display. We chose to tackle the issue at hand in a twofold manner. First, key properties of the ocean surface spectrum allow us to accelerate the computation of its Inverse Fourier Transform considerably. Second, we reduce the amount of spectral data to transform. We achieve the latter by improving upon the level-of-detail approach by Eric Bruneton [2010] and Dupuy and Bruneton [2012] with a multi-resolution scheme, where specific datasets are generated at a lower resolution than the other ones.

The remainder of this chapter is organized as follows: Section 4.1 gives a compact summary of ocean surface synthesis by means of a wave energy spectrum. Section 4.2 describes characteristic properties of wave spectra which enable us to accelerate the computation of the Inverse Fourier Transform. Section 4.3 introduces the additional wave spectrum based datasets which we employ for rendering, whereas Section 4.4 elaborates on our wave spectrum specific level of detail approach. Last, Section 4.5 gives an overview of our demo application and the algorithms it incorporates.

4.1 Spectrum Synthesis

Recall Equation 3.35 in Chapter 3.3, where we defined surface elevation as follows:

$$\eta(\mathbf{x}, t) = \sum_{\mathbf{k}} \frac{1}{\sqrt{2}}(\xi_r + \mathrm{i}\xi_i)\sqrt{2\Theta(\mathbf{k})\Delta k_x \Delta k_z}\ \mathrm{e}^{-\mathrm{i}\omega(\mathbf{k})t}\ \mathrm{e}^{\mathrm{i}\mathbf{k}^{\mathsf{T}}\mathbf{x}} \tag{4.1}$$

where ξ_r and ξ_i are random scalars drawn from the Standard Normal Distribution. Both the wavevector domain $\mathbf{k}$ and the spatial domain $\mathbf{x}$ are of resolution $N \times N$, where N is a natural number, and a power of two. The latter requirement is a concession to the Fast Fourier Transform algorithm, which works fastest at such resolutions [Cooley and Tukey, 1965]. Moreover, the spatial domain $\mathbf{x}$ has an extent of $L \times L$, thus the grid point spacing of the wavevector domain is as follows:

$$\Delta k_x = \Delta k_z = \Delta k = \frac{2\pi}{L}$$

We may subsume part of Equation 4.1 into the following:

$$h_0(\mathbf{k}) = \frac{1}{\sqrt{2}}(\xi_r + \mathrm{i}\xi_i)\sqrt{2\Theta(\mathbf{k})\Delta k_x \Delta k_z} \tag{4.2}$$

where $h_0(\mathbf{k})$ represents a random generated spectrum based on wave energy spectrum $\Theta(\mathbf{k})$. As long as the wavevector domain and the wave energy spectrum do not change, h_0 does not change either. Hence, it is necessary to generate h_0 only once for a specific set of parameters such as area, resolution, wind and fetch. As h_0 by itself has no notion of time, we still need to take care of animation. Based on Equation 4.1 and 4.2 we may write:

$$h(\mathbf{k}, t) = h_0(\mathbf{k})\ \mathrm{e}^{-\mathrm{i}\omega(\mathbf{k})t} \tag{4.3}$$

where t denotes the time. Thus, h as defined by Equation 4.3 has to be computed whenever one requires a new keyframe for the animated ocean surface. Equation 4.2 on the other hand is only to be evaluated anew in case parameters change, easing the computational workload by a considerable margin.

4.2 Discrete Fourier Transform

We compute surface elevation in the form of an Inverse Discrete Fourier Transform as follows:

$$\eta(\mathbf{x}, t) = \sum_{\mathbf{k}} h(\mathbf{k}, t)\, \mathrm{e}^{\mathrm{i}\mathbf{k}^{\mathsf{T}}\mathbf{x}} \tag{4.4}$$

with

$$\mathbf{k} = (x, y) \in \{(\alpha\Delta k, \beta\Delta k) | -\frac{N}{2} \leq \alpha < \frac{N}{2}, -\frac{N}{2} < \beta \leq \frac{N}{2}\}$$

The wavevector $\mathbf{k} = (0, 0)$ gives the location of the zero frequency component, which encodes the average value of the signal in the spatial domain. In our specific case that would be the mean surface elevation $\mathrm{E}[\eta] = 0$. As one may read off the above definition of the wavevector domain, the zero frequency component is located near the center of the wavevector domain, where $\alpha = 0$ and $\beta = 0$. Actual implementations of the Inverse Fourier Transform, such as FFTW [Frigo and Johnson, 2005], expect the zero frequency component as the first element, i.e., at the upper left of the two-dimensional input array. The wavevector domain of such implementations is defined as follows:

$$\mathbf{k} = (x, y) \in \{(\alpha\Delta k, \beta\Delta k) | 0 \leq \alpha < N, 0 \leq \beta < N\}$$

We know that the spectrum represents a periodic signal, therefore the spectrum repeats itself at infinity in both directions. Hence, independent of where the zero frequency is located inside the wavevector domain, we may write:

$$h(\mathbf{k}(\alpha\Delta k, \beta\Delta k), t) = h(\mathbf{k}((\alpha + lN)\Delta k, (\beta + mN)\Delta k), t) \qquad l, m \in \mathbb{Z}$$

Figure 4.1 shows how we may use the periodical nature of the spectrum to convert between the underlying layouts of the two different wavevector domains of interest.

Hermitian Spectrum

The Fourier Transform of real-valued input is *Hermitian* – the real part of the resulting spectrum is an *even* function and the imaginary part is *odd* [Bracewell, 2000]. A function $f(n)$ is said to be even if $f(-n) = f(n)$. On the other hand, a function $f(n)$ is said to be odd if $f(-n) = -f(n)$, with $f(0) = 0$. Let $\mathcal{F}(k)$ be the Fourier Transform of a real-valued function $f(n)$, then the following is true:

$$\mathcal{F}(-k) = \mathcal{F}(k)^* \tag{4.5}$$

where $*$ denotes the complex conjugate operator. The relation is similar to a centrosymmetric one, $\mathcal{F}(-k) = \mathcal{F}(k)$, with $\mathcal{F}(0)$ as a fixpoint at the center. The only difference is the complex conjugation. Fourier Transform implementations such as FFTW [Frigo and Johnson, 2005] take

(a)

$+3.6+0.0\text{i}$ $-2.0-1.5\text{i}$	$+0.5-0.9\text{i}$ $-0.2-0.7\text{i}$	$-8.3+0.0\text{i}$ $+2.7+2.6\text{i}$	$+0.5+0.9\text{i}$ $+0.1+0.3\text{i}$
$-0.5+0.0\text{i}$ $-2.0+1.5\text{i}$	$+0.5-0.2\text{i}$ $+0.1-0.3\text{i}$	$+0$ $+2.7-2.6\text{i}$	$+0.5+0.2\text{i}$ $-0.2+0.7\text{i}$

(b)

Q2	Q1
Q3	Q4

(c)

Q2	Q1	Q2	Q1
Q3	Q4	Q3	Q4
Q2	Q1	Q2	Q1
Q3	Q4	Q3	Q4

(d)

Q4	Q3
Q1	Q2

(e)

$+0$ $+2.7-2.6\text{i}$	$+0.5+0.2\text{i}$ $-0.2+0.7\text{i}$	$-0.5+0.0\text{i}$ $-2.0+1.5\text{i}$	$+0.5-0.2\text{i}$ $+0.1-0.3\text{i}$
$-8.3+0.0\text{i}$ $+2.7+2.6\text{i}$	$+0.5+0.9\text{i}$ $+0.1+0.3\text{i}$	$+3.6+0.0\text{i}$ $-2.0-1.5\text{i}$	$+0.5-0.9\text{i}$ $-0.2-0.7\text{i}$

Figure 4.1: (a) The spectrum of a real-valued signal ($N = 4$), the zero frequency component is highlighted in red. (b) Quadrant layout of the spectrum, the quadrant containig the zero frequency component is highlighted in red. (c) The signal is periodic, thus the spectrum is too. We may replicate the spectrum's quadrants along both dimensions, giving us alternative quadrant layouts to choose from. (d) The quadrant layout we seek, where the zero frequency component is held by the quadrant at the origin. Compared to the original layout, diagonally opposite quadrants have been swapped. (e) The spectrum is rearranged according to the modified quadrant layout, placing the zero frequency component at the origin.

advantage of Equation 4.5 to be able to provide optimized transform functionality for real-valued data. For example, for a forward Fourier Transform of real-valued data it is necessary to compute only half of the spectrum, since the other half is implicitely given as the complex conjugate of the first half. Likewise, for the Inverse Fourier Transform only half of the spectrum is necessary, the other half is implicitely given. Such optimized transforms provide gain in performance memory-wise, as the spectrum's size is halved, as well as computation-wise, as only half the data is actually processed.

Surface elevation η is a real-valued function, therefore its spectrum must be Hermitian, too. Let $\mathcal{F}(\mathbf{k})$ be the Fourier Transform of surface elevation η, then we may write the Hermitian property as follows:

$$\mathcal{F}(-\mathbf{k}) = \mathcal{F}(\mathbf{k})^* \tag{4.6}$$

where finding $\mathcal{F}(\mathbf{k})^*$ for its counterpart $\mathcal{F}(-\mathbf{k})$ has proven to be non-trivial for even-sized

(a)

$$\begin{pmatrix} a & b & c \\ d & e & d^* \\ c^* & b^* & a^* \end{pmatrix}$$

(b)

$$\begin{pmatrix} -0.2 - 0.7\mathrm{i} & +2.7 + 2.6\mathrm{i} & +0.1 + 0.3\mathrm{i} \\ +0.5 - 0.2\mathrm{i} & +0 & +0.5 + 0.2\mathrm{i} \\ +0.1 - 0.3\mathrm{i} & +2.7 - 2.6\mathrm{i} & -0.2 + 0.7\mathrm{i} \end{pmatrix}$$

(c)

$$\begin{pmatrix} a & b & c & d & e \\ f & g & h & i & j \\ k & l & m & l^* & k^* \\ j^* & i^* & h^* & g^* & f^* \\ e^* & d^* & c^* & b^* & a^* \end{pmatrix}$$

(d)

$$\left(\begin{array}{cccc|c} +3.6 + 0.0\mathrm{i} & +0.5 - 0.9\mathrm{i} & -8.3 + 0.0\mathrm{i} & +0.5 + 0.9\mathrm{i} & +3.6 + 0.0\mathrm{i} \\ -2.0 - 1.5\mathrm{i} & -0.2 - 0.7\mathrm{i} & +2.7 + 2.6\mathrm{i} & +0.1 + 0.3\mathrm{i} & -2.0 - 1.5\mathrm{i} \\ -0.5 + 0.0\mathrm{i} & +0.5 - 0.2\mathrm{i} & +0 & +0.5 + 0.2\mathrm{i} & -0.5 + 0.0\mathrm{i} \\ -2.0 + 1.5\mathrm{i} & +0.1 - 0.3\mathrm{i} & +2.7 - 2.6\mathrm{i} & -0.2 + 0.7\mathrm{i} & -2.0 + 1.5\mathrm{i} \\ \hline +3.6 + 0.0\mathrm{i} & +0.5 - 0.9\mathrm{i} & -8.3 + 0.0\mathrm{i} & +0.5 + 0.9\mathrm{i} & +3.6 + 0.0\mathrm{i} \end{array}\right)$$

Figure 4.2: The center element and the zero frequency component are highlighted in red respectively. (a) Hermitian matrix ($N = 3$). (b) Example spectrum matching the Hermitian matrix layout. (c) Hermitian matrix ($N = 5$). (d) An even-sized spectrum ($N = 4$), the elements of the first row and column lack their complex conjugate counterparts. Only after we replicate the spectrum until both its dimensions have an odd-numbered size ($N = 5$), we may find that for all elements of the spectrum there is a pair which satisfies the Hermitian condition $\mathcal{F}(-\mathbf{k}) = \mathcal{F}(\mathbf{k})^*$.

spectra. Figure 4.2 shows that it is the periodic nature of the spectrum that allows us to find matching complex conjugate pairs for all elements of an even-sized spectrum.

Hermitian Wave Spectrum

The spectrum $h(\mathbf{k}, t)$ from Equation 4.3 is *not* Hermitian for two reasons. First, the real-valued part of the spectrum, namely the wave energy spectrum $\Theta(\mathbf{k})$, would need to be an even function, which is only the case if the directional spread employed by the wave energy spectrum is centrosymmetric. Only two of the directional spreading functions presented in this work are centrosymmetric: the Unified spectrum's and the Phillips spectrum's. Second, each element of $h_0(\mathbf{k})$ is generated in combination with a pair of random numbers, which makes it close to impossible to end up with matching pairs $h_0(-\mathbf{k}) = h_0(\mathbf{k})^*$. As we would like to profit from the performance gains an optimized Inverse Fourier Transform for real-valued functions offers, we are in need of a Hermitian spectrum. Thus, we may rewrite Equation 4.3 as follows:

$$h(\mathbf{k}, t) = \frac{1}{2} h_0(\mathbf{k}) \mathrm{e}^{-\mathrm{i}\omega(\mathbf{k})t} + \frac{1}{2} h_0(-\mathbf{k})^* \mathrm{e}^{\mathrm{i}\omega(\mathbf{k})t} \tag{4.7}$$

which gives us a Hermitian spectrum, independent of both the generated random numbers and whether the wave energy spectrum is an even function or not.

Note: Surface elevation η as computed by Equation 4.4 gives the same results for Equation 4.3 and Equation 4.7. Still, Equation 4.3 requires a standard Inverse Fourier Transform with a complete complex-valued source spectrum and a complex-valued result array of the same size. The real-valued part of the result array represents the surface elevation, whereas the imaginary-valued part is a by-product which holds no relevant information and is thefore to be discarded.

Equation 4.7, on the other hand, allows to apply aforementioned Inverse Fourier Transform optimized for Hermitian spectra, improving computation speed and memory usage roughly by a factor of two [Frigo and Johnson, 2017].

Combined Inverse Fourier Transform

Given one Hermitian spectrum, one may apply a specialized Inverse Fourier Transform which requires only half of the actual spectrum. Given *two* Hermitian spectra, one may combine them into a single spectrum and apply a standard Inverse Fourier Transform to retrieve both real-valued signals at once [Smith and Smith, 1997]. Let a and b be real-valued signals with identical underlying spatial domains. Let us denote the Fourier Transform with $\mathcal{F}$, and the Inverse Fourier Transform with $\mathcal{F}^{-1}$, then we may write:

$$c = \mathcal{F}^{-1}(\mathcal{F}(a) + \mathrm{i}\mathcal{F}(b)) \tag{4.8}$$

where c is complex-valued. We are able to retrieve the original real-valued signals from the real and imaginary components of c respectively:

$$a = \mathcal{R}(c) \qquad\qquad b = \mathcal{I}(c)$$

We will make heavy use of this kind of optimization later on, because most of our additional data required for rendering comes in pairs of Hermitian spectra.

Complex Conjugate Indices

For implementation purposes, let us view the two-dimensional Hermitian spectrum h as a two-dimensional array with size $N \times N$. Each element of h is identified by a pair of indices (i, j), with $0 \leq i, j < N$. Then, for an element at index (i, j), we may compute the index (m, n) of the corresponding complex conjugate element as follows:

$$m = (N - i) \bmod N \qquad\qquad n = (N - j) \bmod N \tag{4.9}$$

We may write the Hermitian property in terms of indices:

$$h(i, j) = h(m, n)^*$$

Moreover, let us view h_0 as a two-dimensional array with size $N \times N$, with $h_0(\mathbf{k}) = h_0(i, j)$. Then, we may employ Equation 4.9 to retrieve $h_0(-\mathbf{k})^* = h_0(m, n)^*$ as required by Equation 4.7.

4.3 Slopes and Displacements

At this point we are able to compute surface elevation by application of the Inverse Discrete Fourier Transform on a spectrum we generate. For lighting purposes we also need to find the surface's slope vectors to be able to compute the surface's normal vectors. The most simple way to compute the slope is through finite differences in the spatial domain of the surface. Such an

approach is efficient memory- and computation-wise, but may lack quality, because it can be a poor approximation to the slope of waves with small wavelengths [Tessendorf, 1999]. We are able to obtain more precise slope vectors by employing *spectral differentation*. Spectral differentiation allows us to find the derivatives of a function via the Fourier Transform [Trefethen, 2000].

First we will look at the more simple, one-dimensional case of spectral differentiation. We reduce the spatial domain as well as the wavevector domain from beforehand to one dimension. Let N be the resolution of the spatial and wavenumber domains, and L the size of the spatial domain. Moreover, let $\alpha \in \mathbb{N}$ with $-\frac{N}{2} \leq \alpha < \frac{N}{2}$, let the spatial domain be $x \in \alpha\frac{L}{N}$, and let the wavenumber domain be $k \in \alpha\frac{2\pi}{L}$. Let $g(k)$ be the Discrete Fourier Transform of $f(x)$, then we may write the Inverse Discrete Fourier Transform as follows:

$$f(x) = \sum_k g(k)\ \mathrm{e}^{\mathrm{i}kx}$$

If we follow the lead of the continuous Fourier Transform [Trefethen, 2000], then, in the discrete case, we may compute the nth derivative of $f(x)$ as follows:

$$\frac{\mathrm{d}^n f(x)}{\mathrm{d}x^n} = \sum_k (\mathrm{i}k)^n\ g(k)\ \mathrm{e}^{\mathrm{i}kx} \tag{4.10}$$

Johnson [2011] shows that Equation 4.10 is not correct for odd n in combination with even N. For example, if $f(x)$ is a real function, then $g(k)$ is Hermitian. But $\mathrm{i}k\ g(k)$ is *not* Hermitian for even N, therefore the first order derivative of $f(x)$ would end up being a complex-valued function instead of a real-valued one. To get correct results for all n and N, we rewrite the above equation as follows:

$$\frac{\mathrm{d}^n f(x)}{\mathrm{d}x^n} = \sum_k d(k,n)\ g(k)\ \mathrm{e}^{\mathrm{i}kx} \tag{4.11}$$

$$d(k,n) = \begin{cases} 0 & \text{if } n \text{ odd, and } N \text{ even, and} |k| = \frac{N}{2}\frac{2\pi}{L}, \\ (\mathrm{i}k)^n & \text{else.} \end{cases} \tag{4.12}$$

Now we may go back to the two-dimensional case, with our original two-dimensional domains $\mathbf{x}$ and $\mathbf{k}$. The resolution of said domains is $N \times N$, the size of the spatial domain is $L \times L$. Let $g(\mathbf{k})$ be the Discrete Fourier Transform of $f(\mathbf{x})$, then we may write the Inverse Discrete Fourier Transform in two dimensions as follows:

$$f(\mathbf{x}) = \sum_{\mathbf{k}} g(\mathbf{k})\ \mathrm{e}^{\mathrm{i}\mathbf{k}^\mathsf{T}\mathbf{x}}$$

Based on Equation 4.11, and by reusing Equation 4.12 for both dimensions (N and L are equal for both dimensions), we find the derivatives of $f(\mathbf{x})$:

$$\frac{\partial^{n+m} f(\mathbf{x})}{\partial x^n \partial z^m} = \sum_{\mathbf{k}} d(k_x,n) d(k_z,m)\ g(\mathbf{k})\ \mathrm{e}^{\mathrm{i}\mathbf{k}^\mathsf{T}\mathbf{x}} \tag{4.13}$$

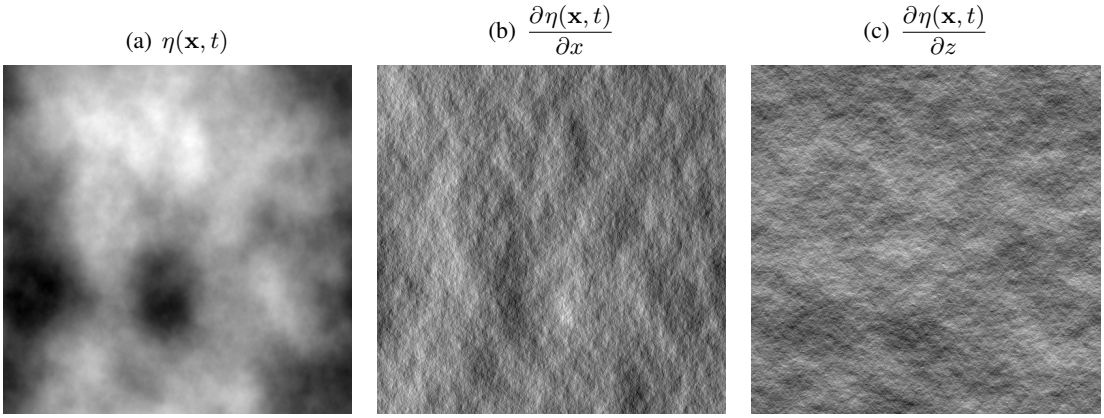


Figure 4.3: Brighter means larger values, darker means lower values. (a) An example surface wave height realization, displayed in greyscale (Unified spectrum, $U_{10} = 30m \cdot s^{-1}$, $F = 500km$). (b) The first component of the slope vector. (c) The second component of the slope vector.

where x and z denote the two components of vector $\mathbf{x}$, and k_x and k_z the two components of wavevector $\mathbf{k}$ respectively. As we need to find the surface slope vector, we need to compute the surface elevation's first order partial derivatives. Given surface elevation $\eta(\mathbf{x}, t)$, we obtain the two-dimensional surface slope vector $\mathbf{s}$ as follows:

$$\mathbf{s}(\mathbf{x}, t) = \left[\frac{\partial \eta(\mathbf{x}, t)}{\partial x}, \frac{\partial \eta(\mathbf{x}, t)}{\partial z} \right] \tag{4.14}$$

We leave it as an exercise for the reader to substitute the terms $\eta(\mathbf{x}, t)$ and $h(\mathbf{k}, t)$ into Equation 4.13 to be able to compute the surface slope vector as given in Equation 4.14. Figure 4.3 depicts an example instance of surface elevation η as well as its associated slopes in greyscale.

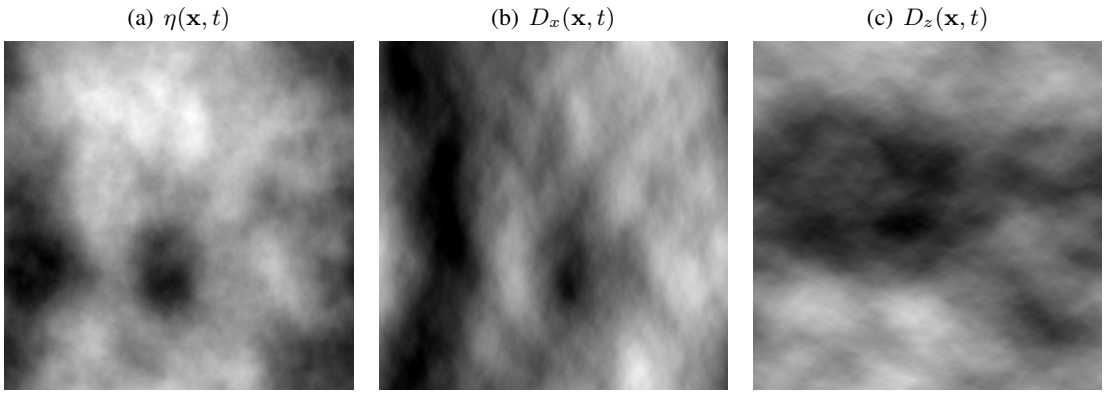
The two spectra which constitute the surface slope vector in the wavevector domain are Hermitian. Therefore we are able to apply the optimisation from Equation 4.8, which allows us to combine both spectra into one, obtaining both components of the surface slope vector with just one Inverse Fourier Transform.

Normal Vectors

In this work we employ a right-handed Carthesian coordinate system where the positive Y-axis points in the opposite direction of earth's gravity. The two-dimensional surface slope vector lies in the XZ-plane of the world space coordinate system. Based on the surface slope vector we may find the unit length surface normal vector $\mathbf{n}$ as follows:

$$\mathbf{n} = \frac{(-s_x, 1, -s_z)}{\left\| (-s_x, 1, -s_z) \right\|}$$

where s_x and s_z denote the two components of slope vector $\mathbf{s}$ as obtained by Equation 4.14.



(a) $\eta(\mathbf{x}, t)$ (b) $D_x(\mathbf{x}, t)$ (c) $D_z(\mathbf{x}, t)$

Figure 4.4: Brighter means larger values, darker means lower values. (a) An example surface wave height realization, displayed in greyscale (Unified spectrum, $U_{10} = 30m \cdot s^{-1}$, $F = 500km$). (b) The first component of the displacement vector. (c) The second component of the displacement vector.

Displacements

The waves generated with the methods presented up to now tend to have rounded peaks and troughs, which is typical for fair weather conditions. But we also would like to be able to synthesize waves matching poor weather conditions, such as strong winds or even storms. During such weather, ocean waves are sharply peaked at their tops and flattened at the bottoms. Tessendorf [1999] describes a method based on the Fourier Transform to produce such choppy waves. The concept is simple: displace the grid points of the spatial domain in the XZ-plane, with the displacement varying locally with the waves. Note that the displacement is a two-dimensional vector, therefore we end up computing a vector field. Similar to spectral differentiation, we compute an Inverse Fourier Transform of the spectrum h, where the latter is modified by an additional term. We may write the displacement vector field as follows:

$$\mathbf{D}(\mathbf{x}, t) = \big[D_x(\mathbf{x}, t), D_z(\mathbf{x}, t)\big] \tag{4.15}$$

with

$$D_x(\mathbf{x}, t) = \sum_{\mathbf{k}} d(k_x, k)\, h(\mathbf{k}, t)\, \mathrm{e}^{\mathrm{i}\mathbf{k}^\mathsf{T}\mathbf{x}} \tag{4.16}$$

$$D_z(\mathbf{x}, t) = \sum_{\mathbf{k}} d(k_z, k)\, h(\mathbf{k}, t)\, \mathrm{e}^{\mathrm{i}\mathbf{k}^\mathsf{T}\mathbf{x}} \tag{4.17}$$

$$d(l, k) = \begin{cases} 0 & \text{if } N \text{ even, and } k = 0 \text{ or} |l| = \frac{N}{2}\frac{2\pi}{L}, \\ -\mathrm{i}\frac{l}{k} & \text{else.} \end{cases} \tag{4.18}$$

where the term $d(l, k)$, in addition to make sure the resulting spectrum is Hermitian, also avoids a division by zero at $k = 0$. Figure 4.4 shows an example instance of surface elevation η as well as its associated displacements in greyscale. Because the spectra on the righthand side of

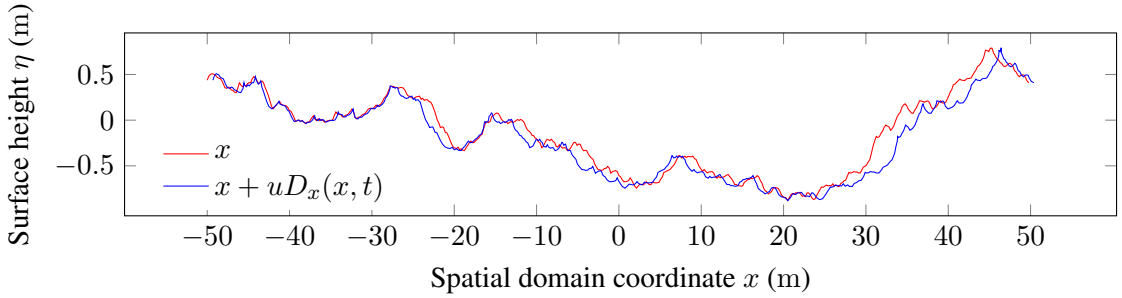


Figure 4.5: Red: the original wave profile. Blue: the displaced wave profile, note the steep tops and the flattened valleys ($u = -3$).

Equations 4.16 and 4.17 are Hermitian, we are again able to apply Equation 4.8 and obtain both components of the displacement vector field with just one Inverse Fourier Transform.

Given grid point $\mathbf{x} = (x, z)$, time t and surface elevation $\eta(\mathbf{x}, t)$, we may write the corresponding three-dimensional, vertically displaced vertex coordinate as follows:

$$\mathbf{v}(\mathbf{x}, t) = \begin{bmatrix} x \\ \eta(\mathbf{x}, t) \\ z \end{bmatrix} \tag{4.19}$$

With displacement vector $\mathbf{D}(\mathbf{x}, t)$ at hand, according to Tessendorf [1999] we may compute the three-dimensional, vertically and horizontally displaced vertex coordinate as follows:

$$\mathbf{w}(\mathbf{x}, t) = \begin{bmatrix} x + uD_x(\mathbf{x}, t) \\ \eta(\mathbf{x}, t) \\ z + uD_z(\mathbf{x}, t) \end{bmatrix} \tag{4.20}$$

where $u \in \mathbb{R}$ is a user-controlled parameter to scale the importance of the displacement vector. We have found that the above formula is correct as long as $u \leq 0$, otherwise the effect is the opposite of what we want to achieve. Looking at Equation 4.20, one can see that it is not the wave heights that are altered, but instead the grid points on the XZ-plane are transformed based on the spatial structure of the height field. This particular transformation accomplishes the effects we sought: the waves' peaks are sharpened and the waves' valleys are broadened. Figure 4.5 shows two profiles of the wave height along one direction, where one profile uses displaced coordinates, while the other does not.

Slopes after Displacement

Based on Figure 4.5 one may see that the application of displacements does not only alter the underlying grid points, but as a direct consequence the surface's slopes, too. Hence, we need to adjust our computation of the slope vector to the new circumstances. Given surface elevation $\eta(\mathbf{x}, t)$, displacement vector $\mathbf{D}(\mathbf{x}, t)$, and grid point transformation $\mathbf{x} + u\mathbf{D}(\mathbf{x}, t)$, according

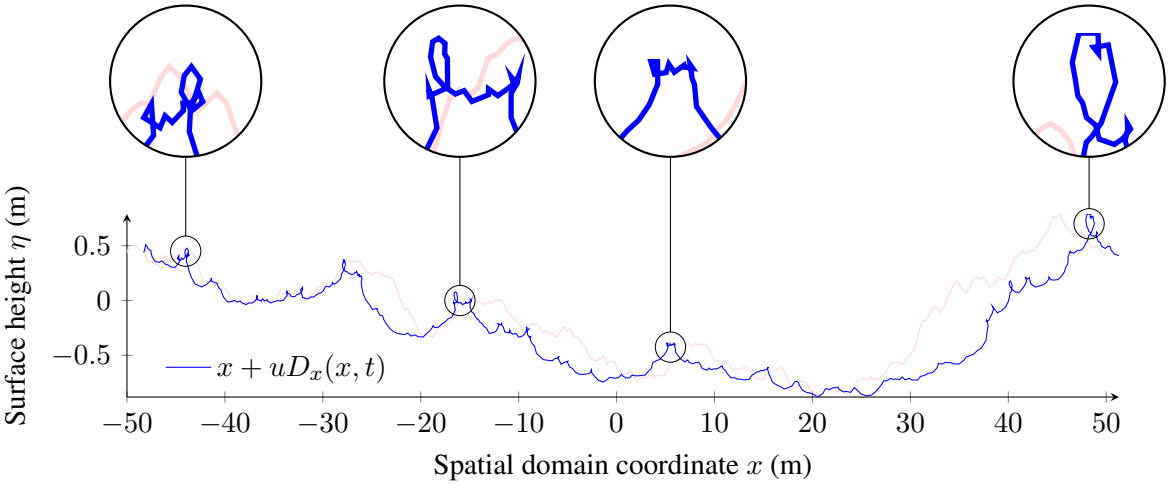


Figure 4.6: Background: the original wave profile. Foreground: the displaced wave profile, with some of the self-intersections highlighted. We chose a large displacement scaling parameter ($u = -7.5$) to better emphasize the effect of self-intersection.

to Dupuy and Bruneton [2012, private communication] we obtain the two-dimensional surface slope $\mathbf{s}$ as follows:

$$\mathbf{s}(\mathbf{x},t) = \left[\frac{\dfrac{\partial \eta(\mathbf{x},t)}{\partial x}}{1 + u\dfrac{\partial D_x(\mathbf{x},t)}{\partial x}}, \frac{\dfrac{\partial \eta(\mathbf{x},t)}{\partial z}}{1 + u\dfrac{\partial D_z(\mathbf{x},t)}{\partial z}} \right] \tag{4.21}$$

where u is the displacement scale parameter. Now, in addition to surface height η, displacement vector $\mathbf{D}$, and the slopes of η, we also need to compute a partial derivative of D_x and D_z each. Again, we may compute the latter two by means of spectral differentiation, see Equation 4.13. Morever, we are again able to apply the optimisation from Equation 4.8, which allows us to transform the spectra of the two partial derivatives as one.

Self-Intersections

The method by Tessendorf [1999] we use to generate choppy waves is not devoid of drawbacks. As shown in Figure 4.6, some of the displacement vectors we compute may be large enough to cause the displaced geometry to self-intersect. Near the top of some waves the surface actually passes through itself, and as a consequence inverts, with the surface normal pointing inwards instead of outwards. We may get rid of this undesired side-effect in a rather simple way: reduce the magnitude of displacement scaling parameter u. On the other hand, Tessendorf suggests that we use said self-intersections to our advantage, because they may be able to signal the breaking of waves, as well as the production of foam and spray. Hence, we need to be able to test for such self-intersections in an efficient manner. Tessendorf recommends to compute the determinant of the *Jacobian matrix* of the transformation from grid point $\mathbf{x}$ to displaced grid

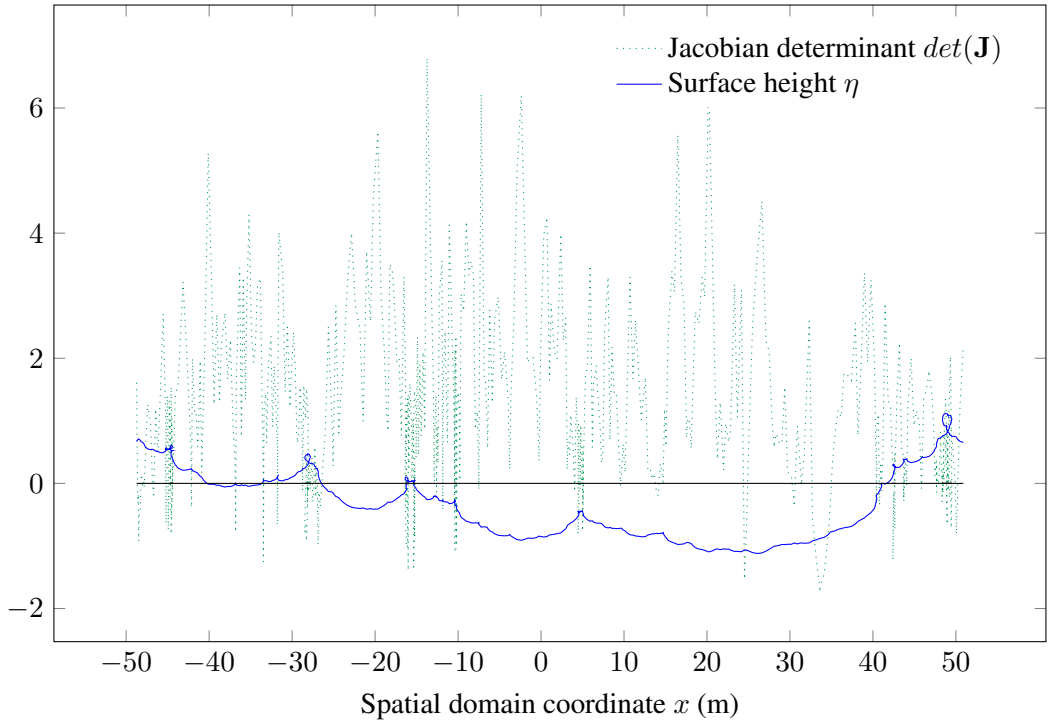


Figure 4.7: Blue: the displaced wave profile ($u = -7.5$). Green: the Jacobian determinant. Note, that each instance of surface self-intersection is accompanied by negative Jacobian determinants.

point $\mathbf{x} + u\mathbf{D}(\mathbf{x}, t)$. Based on the determinant we may decide if self-intersection is taking place. First, we need the Jacobian matrix of the grid point transformation $\mathbf{g}(\mathbf{x}, t) = \mathbf{x} + u\mathbf{D}(\mathbf{x}, t)$. The Jacobian matrix is the matrix of all first order partial derivatives of a vector valued function. As grid point transformation $\mathbf{g}$ maps from $\mathbb{R}^2$ to $\mathbb{R}^2$, the Jacobian matrix is 2×2 in size. Let $\mathbf{J}$ be the Jacobian matrix of transformation $\mathbf{g}$, then we may write:

$$\mathbf{J}(\mathbf{x}, t) = \begin{bmatrix} J_{xx} & J_{xz} \\ J_{zx} & J_{zz} \end{bmatrix} = \begin{bmatrix} 1 + u\dfrac{\partial D_x(\mathbf{x}, t)}{\partial x} & u\dfrac{\partial D_x(\mathbf{x}, t)}{\partial z} \\ u\dfrac{\partial D_z(\mathbf{x}, t)}{\partial x} & 1 + u\dfrac{\partial D_z(\mathbf{x}, t)}{\partial z} \end{bmatrix}$$

Next, we compute the determinant of the Jacobian matrix:

$$det(\mathbf{J}(\mathbf{x}, t)) = J_{xx}J_{zz} - J_{zx}J_{xz}$$

If the Jacobian determinant at $\mathbf{x}$ is positive, then $\mathbf{g}$ preserves orientation near $\mathbf{x}$. If it is negative, $\mathbf{g}$ reverses orientation. We are interested in the latter case, because wherever the surface self-intersects, it actually folds back on itself via a loop, and a loop requires a reversal of orientation. Figure 4.7 depicts a displaced surface, as well as the corresponding Jacobian determinants. One can see that wherever the surface self-intersects, the Jacobian determinant is negative.

4.4 Level of Detail

The core of our ocean model are the spectra discussed in Section 3.4. We sample the spectra in frequency space to obtain vertical and horizontal displacements produced by ocean waves. Because the sampling process is done by means of a Discrete Fourier Transform, the resulting data can be seamlessly tiled on the ocean surface. However, tiling a single perturbation pattern either leads to repetitive artifacts or to lack of detail, depending on whether the tile's footprint in world space is small or large. We chose to tackle these issues like Eric Bruneton [2010] and Dupuy and Bruneton [2012], which compute not only one perturbation pattern, but a set of perturbation patterns of different size, which are then superimposed over each other. Although this approach is not able to entirely remove the tiling artifacts from the wave field, it is still able to reduce periodicity to the least common multiple of the pattern periods. Moreover, as the wavenumber ranges of the different patterns complement each other, we are able to extend the sum of waves to include shorter wavelengths, even as far as into the gravity-capillary wave domain. Thus, given a sufficient number of patterns, we may be able to consistently reproduce small-scale detail for closeups of the water surface. In practice, up to four different perturbation patterns may suffice to obtain satisfactory results [Dupuy and Bruneton, 2012].

Since we position this work in the context of real-time rendering, we need to make sure that the accumulated cost of pattern data synthesis does not prevent us to keep real-time framerates. Thus, we may extend the level-of-detail approach by Eric Bruneton [2010] and Dupuy and Bruneton [2012] with a multi-resolution scheme, where specific pattern datasets, such as surface elevation and displacements, are generated at a lower resolution than the other datasets.

In the remainder of this section we upgrade our equations to handle more than one perturbation pattern, we explain how to parametrize the different pattern periods to be able to reduce tiling artifacts as well as to reproduce as much detail as possible, and last we discuss our multi-resolution approach.

Equations

At this point, one perturbation pattern consists of surface elevation η, displacement vector $\mathbf{D}$, the slopes of η, and the first order partial derivatives of $\mathbf{D}$. Because we compute the surface displacements as a sum of waves, it is straightforward to extend our computations to incorporate not just one perturbation pattern, but a set of patterns. Let $l \in \mathbb{N}^+$ represent the number of different patterns, then we may compute the combined surface elevation as follows:

$$y(\mathbf{x}, t) = \sum_{i=0}^{l-1} \eta_i(\mathbf{x}, t) \tag{4.22}$$

In lockstep with the above we compute the slope of the combined surface elevation:

$$\mathbf{s}(\mathbf{x}, t) = \left[\frac{\partial y(\mathbf{x}, t)}{\partial x}, \frac{\partial y(\mathbf{x}, t)}{\partial z} \right] = \left[\sum_i \frac{\partial \eta_i(\mathbf{x}, t)}{\partial x}, \sum_i \frac{\partial \eta_i(\mathbf{x}, t)}{\partial z} \right] \tag{4.23}$$

Furthermore, we may extend the horizontal grid point transformation required by the choppy wave algorithm to the following:

$$\mathbf{g}(\mathbf{x}, t) = \mathbf{x} + \sum_{i=0}^{l-1} u_i \mathbf{D}_i(\mathbf{x}, t) \tag{4.24}$$

Thus, we may write the slope of the combined displaced surface elevation as follows:

$$\mathbf{s}(\mathbf{x}, t) = \left[\frac{\sum_i \frac{\partial \eta_i(\mathbf{x}, t)}{\partial x}}{1 + \sum_i u_i \frac{\partial D_{xi}(\mathbf{x}, t)}{\partial x}}, \frac{\sum_i \frac{\partial \eta_i(\mathbf{x}, t)}{\partial z}}{1 + \sum_i u_i \frac{\partial D_{zi}(\mathbf{x}, t)}{\partial z}} \right] \tag{4.25}$$

Given the new grid point transformation, we are in need of the corresponding Jacobian matrix to be able to locate surface self-intersections. We may write:

$$\mathbf{J}(\mathbf{x}, t) = \begin{bmatrix} J_{xx} & J_{xz} \\ J_{zx} & J_{zz} \end{bmatrix} = \begin{bmatrix} 1 + \sum_i u_i \frac{\partial D_{xi}(\mathbf{x}, t)}{\partial x} & \sum_i u_i \frac{\partial D_{xi}(\mathbf{x}, t)}{\partial z} \\ \sum_i u_i \frac{\partial D_{zi}(\mathbf{x}, t)}{\partial x} & 1 + \sum_i u_i \frac{\partial D_{zi}(\mathbf{x}, t)}{\partial z} \end{bmatrix} \tag{4.26}$$

One can see that in case the number of patterns equals one, the new equations simply fall back to the original ones.

Tiling

As mentioned before, we aim to reduce tiling artifacts as much as possible by superimposing a set of patterns of different size [Eric Bruneton, 2010]. To be able to do so we require a minimum of two patterns: a large one which defines the general shape of the ocean surface, and a small one for closeup detail. Bridson [2015] argues that if one takes such an approach to reduce periodicity, then it would be best for the different pattern sizes to be related to each other by an irrational number, because only then one may be able to generate an aperiodic signal. In short, given pattern size L, then we may obtain the next tile size as αL, where α is an irrational number. Bridson goes even further, and states that it would be best to choose scaling factor α based on the *golden ratio* $\varphi = (1 + \sqrt{5})/2$. Thus, we devised a simple automatism where the user specifies the size of the largest pattern, and then each subsequent pattern is automatically shrunk in size by factor f, with $f \in \{\varphi^{-1}, 1 - \varphi^{-1}\}$. The two choices for f represent the lengths of the line segments one obtains if a line of length one is split in two based on the golden ratio. Let L be the size of the largest pattern, then we may compute sizes L_i of the successive patterns as follows:

$$L_i = f^i L \qquad\qquad 0 < i < l \tag{4.27}$$

where l is the number of patterns.

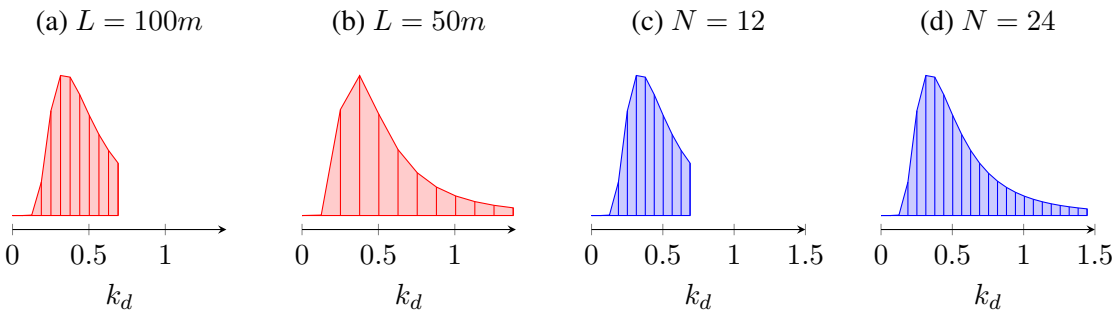


Figure 4.8: Red: Equal resolution, $N = 12$, different size L. (a) covers a smaller wavenumber range than (b) because the former employs a smaller distance $\Delta k = 2\pi/L$ between consecutive sample points. Blue: Equal size, $L = 100\,\mathrm{m}$, different resolution N. The distance between consecutive sample points is equivalent in (c) and (d). (d) covers a larger wavenumber range because it contains more samples than (c).

Although the above gives us an aperiodic signal (not accounting for numerical accuracy), we are not able to completely eliminate periodicity. In some cases the human observer is still able to recognize that the underlying dominant structure, as represented by the largest pattern, appears multiple times. Still, we have found Equation 4.27 to be a surprisingly simple as well as adequate solution to the issue at hand. Should one deem it necessary, he or she may always choose pattern sizes via a different mechanism, based on other irrational numbers, or even by hand as Dupuy and Bruneton did.

Wavenumber Range

We employ multiple patterns of different size to reduce tiling artifacts. A convenient side-effect of such an approach is that the combination of multiple patterns includes more distinct wavenumbers into the sum of waves than a single pattern does. However, before we elaborate on the effect of multiple patterns on wavenumber sampling, we may concern ourselves with the wavenumbers sampled by a single pattern. Recall that both, the spatial domain and the wavevector domain, are defined entirely by size L and resolution N. The minimal, non-zero wavenumber representable by the wavevector domain is $k = 2\pi/L$, which depends only on size. Moreover, the distance between subsequent sample points in both dimensions is defined as $\Delta k = 2\pi/L$. Hence, also Δk depends only on size. By choosing size L, we do not only define the minimal wavenumber we are able to sample, but also how densely we are able to sample the spectrum. The largest wavenumber contained in the wavevector domain is $k = \sqrt{2}\pi N/L$. So, given size L, resolution N defines how large the wavenumbers are allowed to grow, giving us an upper limit of the wavenumbers involved in the sum of waves. In short, with size L and resolution N specified for a pattern, we are able to compute the pattern's wavenumber range and sampling density.

Let k_d be a set of distinct wavenumbers based on parameters L and N. With the wavenumber set k_d at hand we may sample a one-dimensional wavenumber spectrum. Figure 4.8 depicts a

set of reconstructions of the same one-dimensional wavenumber spectrum, where one can see the effects of size L and resolution N on the wavenumber set k_d, and thus on the reconstructed spectrum.

Wavenumber Sampling

We have seen in Figure 4.8 that if we generate only one pattern, based on size and resolution, we either sparsely sample inside a large wavenumber range, or we densely sample inside a small wavenumber range. If, on the other hand, we generate multiple patterns of differing size, then the patterns' wavenumbers ranges complement each other. Thus, we are able to sample a significantly larger wavenumber range at a varying, but reasonable density. Varying, because the sampling density changes from pattern to pattern based on their size. The enlarged wavenumber range may contain short wavelengths up to, and including the gravity-capillary wave domain, which represents waves with a length from one to several centimeters. With such short waves involved in the sum of waves we have the opportunity to reproduce sufficient detail for close-ups of the water surface. Still, we need to make sure to set a lower limit of $\lambda_{min} = 1.73\,\mathrm{cm}$ for the wavelengths included by any of the patterns, because for waves shorter than λ_{min} gravity is replaced by surface tension as the major restoring force, making them capillary waves. The latter are not modeled correctly by any of the wave spectra presented in Section 3.4, because those wave spectra concern themselves exclusively with gravity waves.

Figure 4.9 shows that the wavenumber ranges of the different patterns are partially overlapping, therefore we need to make sure to sample each wavenumber only once. Thus, if the spectrum of one pattern already covers a specific wavenumber range, a second spectrum covering parts or the entirety of that same range, has to be zeroed in the region of overlap. Moreover, it is the spectrum of the pattern with larger size that should have precedence, as it has higher sampling density. Figure 4.10 shows the assembly of the overlapping spectra from Figure 4.9 into one coherent spectrum. We get the best sampling density, if the transition from data from one spectrum to data from the next spectrum takes place at the maximum wavenumber of the spectrum with the smaller wavenumber range.

Note: The typical profile of a one-dimensional wavenumber spectrum suggests that it would be beneficial to employ an adaptive sampling scheme which allows for tight sampling in the low wavenumber range, especially near the peak wavenumber, and sparse sampling in the high wavenumber range [Fréchot, 2007]. Because we employ a Discrete Fourier Transform to compute the sum of waves, we are subject to its constraints. One of those constraints is that the spacing between successive coordinates is constant, namely Δk. Hence, in combination with a Discrete Fourier Transform, we are not allowed to employ a sampling scheme where Δk changes dependent on which part of the wavenumber profile is being sampled.

Multiple Resolutions

Recall, that one perturbation pattern consists of surface elevation η, displacement vector $\mathbf{D}$, the slopes of η, and the first order partial derivatives of $\mathbf{D}$. Moreover, we have a set of such patterns. The computational load to synthesize all the cumulated pattern data has proven to be substantial. To improve upon the status quo, we chose to implement an approach common in

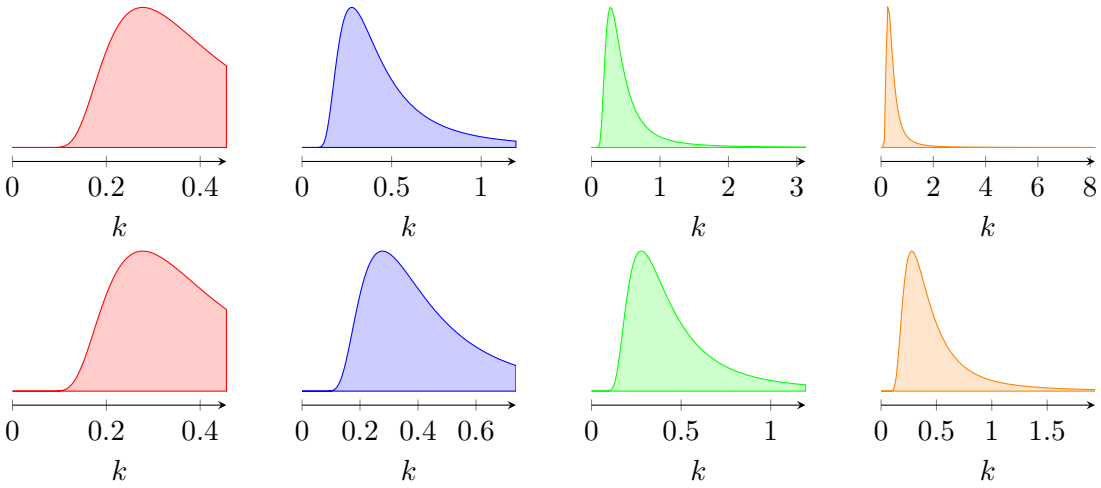


Figure 4.9: A one-dimensional wavenumber spectrum reconstructed at resolution $N = 128$, with initial pattern size $L = 1750\,\mathrm{m}$, and subsequent, reduced pattern sizes obtained by Equation 4.27. Top row: pattern sizes are reduced by factor $f = 1 - \varphi^{-1}$. Bottom row: pattern sizes are reduced by factor $f = \varphi^{-1}$. Note the difference in wavenumber range between top and bottom row.

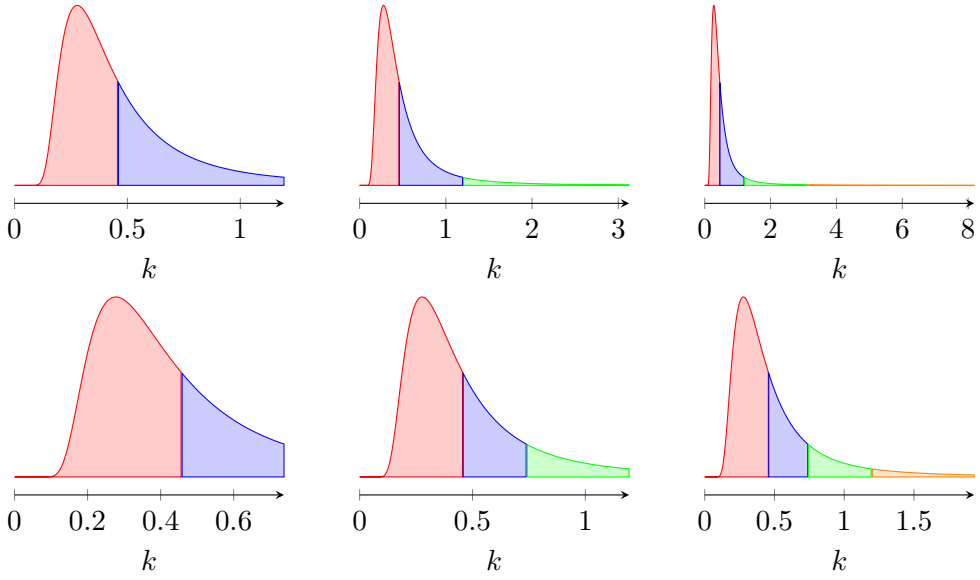


Figure 4.10: Gradual assembly of the reconstructed spectra from Figure 4.9 into one coherent spectrum. The transition from data from one spectrum to data from the next spectrum takes place at the maximum wavenumber of the spectrum with the smaller wavenumber range.

computer graphics: augment low-resolution geometry with high-resolution surface information to generate detailed lighting. For our specific case, that means we may synthesize vertical and horizontal displacements at a lower resolution than the associated derivatives. Given pattern size L, grid point spacing $\Delta k = 2\pi/L$, pattern resolutions $N_h, N_l \in \mathbb{N}$ with $N_l < N_h$, where N_h and N_l are powers of two, then we may define the two following wavevector domains:

$$\mathbf{h} = (k_x, k_z) \in \{(\alpha\Delta k, \beta\Delta k) | -\frac{N_h}{2} \leq \alpha < \frac{N_h}{2}, -\frac{N_h}{2} < \beta \leq \frac{N_h}{2}\} \tag{4.28}$$

$$\mathbf{l} = (k_x, k_z) \in \{(\alpha\Delta k, \beta\Delta k) | -\frac{N_l}{2} \leq \alpha < \frac{N_l}{2}, -\frac{N_l}{2} < \beta \leq \frac{N_l}{2}\} \tag{4.29}$$

where wavevector domain $\mathbf{h}$ is based on the higher resolution N_h and therefore contains more pairs (k_x, k_z) than wavevector domain $\mathbf{l}$. One may notice that $\mathbf{l}$ is a subset of $\mathbf{h}$ as long as both wavevector domains share the same grid point spacing, and thus the same underlying pattern size. Because all pairs (k_x, k_z) in $\mathbf{l}$ are also to be found in $\mathbf{h}$, it follows that wave energy $\Theta(\mathbf{l})$ is a subset of $\Theta(\mathbf{h})$. Hence, there is no need to generate two separate instances of the wave energy spectrum with differing resolutions. The low-resolution variant of the wave energy spectrum is already embedded inside the high-resolution variant, see Figure 4.11. The same applies to the surface elevation spectrum h as defined by Equation 4.3, as well as to the spectra of all derived datasets. If the high-resolution spectrum $ds(\mathbf{h})$ of any dataset has been computed and stored, then we may generate a low-resolution spectrum of said dataset simply by extracting the respective subset $ds(\mathbf{l})$ from $ds(\mathbf{h})$. However, as stated before, we are content with a low-resolution variant of surface elevation η and displacement vectors $\mathbf{D}$ respectively.

Because of our multi-resolution approach, we now have two sets of reconstructions of the wave energy spectrum, based on two different dataset resolutions. It follows that the two sets of reconstructions are assembled into two different coherent spectra. As before, the transition from one reconstruction to the next takes place at the maximum wavenumber of the reconstruction with the smaller wavenumber range. In addition, as to be seen in Figure 4.12, we must take into consideration that the two sets of spectral data have different wavenumber ranges, therefore the transitions from one reconstruction to the next may take place at different wavenumbers for each set.

The goal of our multi-resolution approach is to reduce the computational workload of pattern data synthesis. One can see that the improvement we are able to achieve is directly dependent upon the reduction in resolution of the horizontal and vertical displacements. Thus, we seek to make do with a minimum of data for surface elevation η and displacement vectors $\mathbf{D}$, without compromising the visual end result. Chapter 5 will elaborate on the equilibrium we found between the dataset resolutions and the visual fidelity of the resulting ocean surface.

4.5 Demo Application

At this point we have all building blocks of our work in place, therefore we may proceed with an overview of our showcasing application which puts all the constituent parts into a coherent whole. We implemented a compact demo application on Linux, it is written in Objective-C and

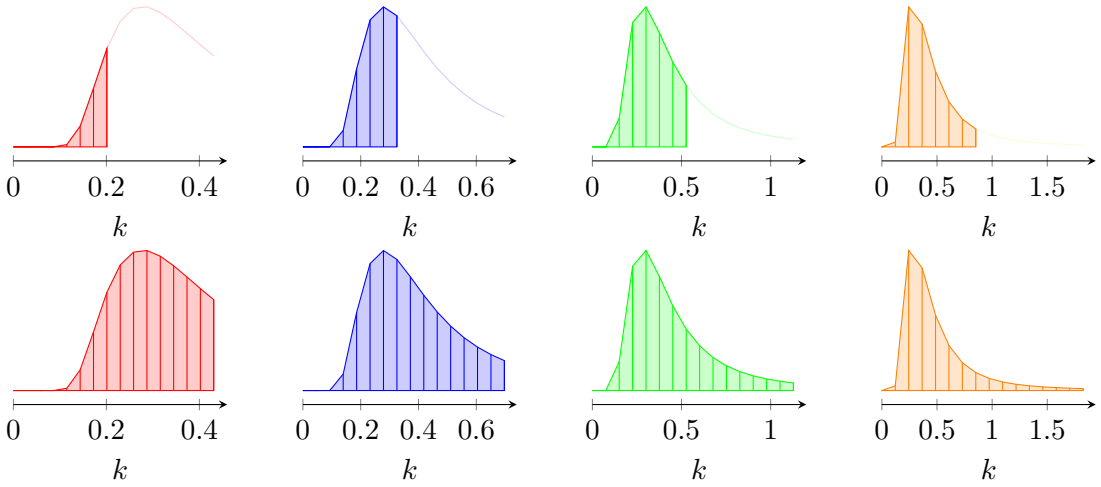


Figure 4.11: A one-dimensional wavenumber spectrum as reconstructed by multiple patterns of different size. Pattern size L shrinks with each consecutive column, it follows that both rows have matching distances between successive samples. Top row uses resolution $N = 8$, whereas bottom row uses $N = 16$. One can see that each reconstruction in top row is a subset of the reconstruction in bottom row.

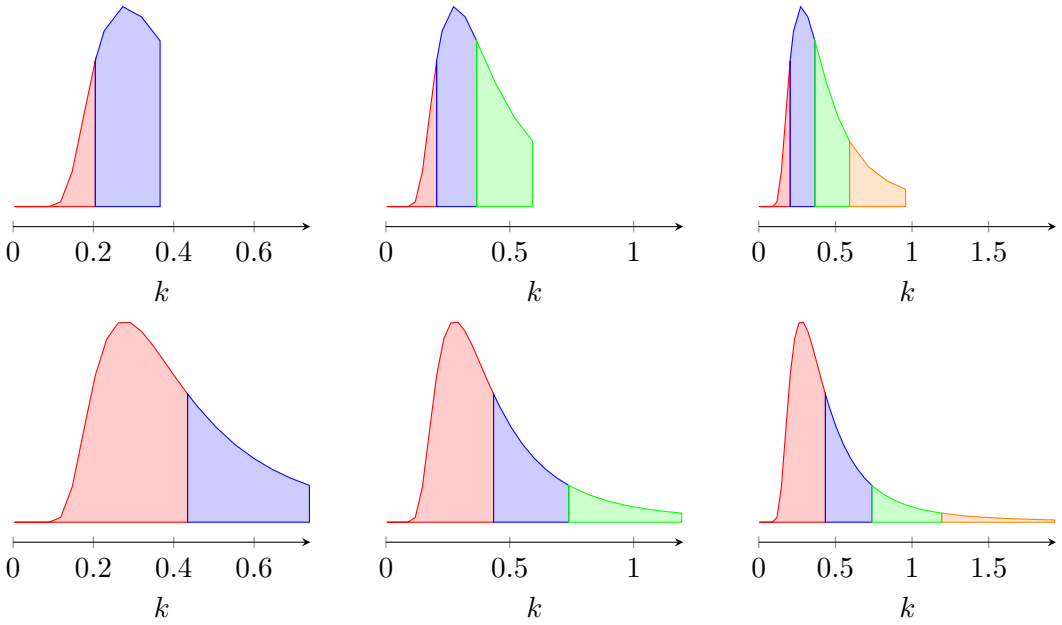


Figure 4.12: Gradual assembly of the reconstructed spectra from Figure 4.11 into one coherent spectrum. Due to the difference in resolution it follows that top row and bottom row transition between two successive reconstructions at different wavenumbers. One can see that the low-resolution variant of the spectrum in top row is still able to adequately sample the low wavenumbers which contain most of the energy of the entire spectrum.

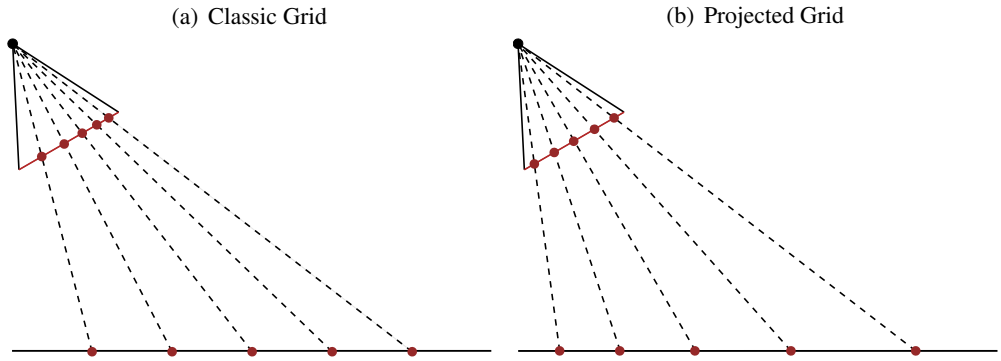


Figure 4.13: (a) A uniform grid in world space, its projection onto the image plane is not uniformly spaced though. (b) A uniform grid on the image plane, its associated world space positions are not uniformly spaced.

employs the GNUstep Base Library, where the latter provides container classes as well as basic algorithms. For rendering we use the compatibility profile of OpenGL 3.3 [OpenGL, a,b], where the GLFW library [GLFW] handles the creation of a window with the desired OpenGL context. Moreover, it is GLFW which gives us the means to handle mouse and keyboard input.

All of the ocean energy spectra presented in this work were, in a first step, implemented in MATLAB. This allowed for easy analysis, debugging and visualization. Only in a second step did we integrate the wave energy spectra into the demo application, making sure the results matched the ones of the MATLAB version. In addition, we chose to employ the same library for Fast Fourier Transforms as MATLAB, namely FFTW [a]. FFTW does not only provide a standard complex-to-complex DFT, but also a more specialized complex-to-real DFT, which has proven to be beneficial for our use case.

The user is given a wide range of control over frequency spectrum generation: he or she may configure the number of patterns, the resolution of the vertical and horizontal displacements, the resolution of the first-order partial derivatives, wind velocity U_{10}, fetch F, size L of the largest pattern, and scaling factor f for the automatic pattern size reduction. Moreover, the user may select the wave energy spectrum model, where all of the spectra presented in Section 3.4 are available options.

The application consists of three running threads: the first thread continuously generates frequency spectra for the desired number of patterns, then hands them to a second thread which applies the IDFT on said spectra. The transformed data, which represents the animated ocean surface, is then given to the main thread for rendering.

Surface Mesh

With all the ocean data synthesized, we may proceed to actually visualize it. First and foremost we need to take care of the mesh representation of the ocean surface. We have already seen that the ocean surface requires a meshing scheme which is able to automatically adapt to highly distinct viewing situations, such as closeups of the water surface as well as views with the

ocean spanning to the horizon. Based on these criteria, we chose to employ the projected grid [Hinsinger et al., 2002, Johanson, 2004]. More specifically, we chose the variant by Johanson [2004], because, in contrast to the variant by Hinsinger et al. [2002], it does not require to restrict the camera's freedom of movement for proper operation. The projected grid is based on a simple concept: to achieve a uniform distribution of detail on the image plane, a uniformly spaced grid is created in post-perspective space, transformed to world space and projected onto the $y = 0$ plane, see Figure 4.13. Thus, we have constant memory consumption, as the resolution of the grid in screen space is constant, and we have adaptive level of detail, because all screen space grid points are projected to points inside the view frustum, independent of the actual viewing situation. After the vertices have been projected onto the $y = 0$ plane, they are displaced vertically and horizontally by surface elevation $y(\mathbf{x}, t)$ and displacement $\mathbf{D}(\mathbf{x}, t)$, respectively. Last, all vertices are projected normally on the screen via the viewing pipeline.

Ocean Lighting

Now that we have dealt with the ocean surface mesh, we may move on to the next aspect of ocean surface rendering, namely the illumination of the water surface. We chose to implement the illumination model introduced by Bruneton et al. [2010] for three reasons. First, it gives visually pleasing as well as convincing results. Second, in step with our work, it is tailored to the deep water of the open ocean. Third, it has been adapted by Eric Bruneton [2010] in such a way that the wave energy spectrum has become an integral part of the lighting model itself.

The lighting term for the water surface, as presented by Bruneton et al., consists of three parts: reflected light from the sun, reflected light from the sky dome, and refracted light from the water. The latter term describes the fraction of light from the sun and sky which has entered the water due to refraction, and leaves it again due to multiple scattering. To model the sky dome and the sun, Bruneton et al. employ the work of Bruneton and Neyret [2008], where the latter compute the interaction of light and the atmosphere in its entirety. We, on the other hand, chose not to extend the scope of this thesis any further with such a complex technique, and therefore opted for the well-established work of Preetham et al. [1999], which is easy to implement and requires no preprocessing steps. The Preetham skylight model, as presented in Preetham et al. [1999, Section 3.1], takes only two parameters: one, the position of the sun on the hemisphere, and two, the haziness of the atmosphere, i.e., turbidity. Figure 4.14 depicts the effect of the two parameters on the resulting sky dome. With the skylight taken care of, we may turn towards the sunlight. We assume the sun to be a directional light source, which is uniform in color and intensity for the entire scene. Preetham et al. [1999, Section 3.2] gives us intensity and color of the sunlight based on the position of the sun, where the result is consistent with the corresponding Preetham skylight.

Whitecaps

The ocean lighting scheme by Bruneton et al. [2010] does not account for the breaking of waves on the open ocean. Neither the geometric deformations, nor the changes in reflectance caused by foam and spray, so-called whitecaps, are incorporated in their model. Dupuy and Bruneton [2012], on the other hand, extend the work of Eric Bruneton [2010] to include whitecaps.

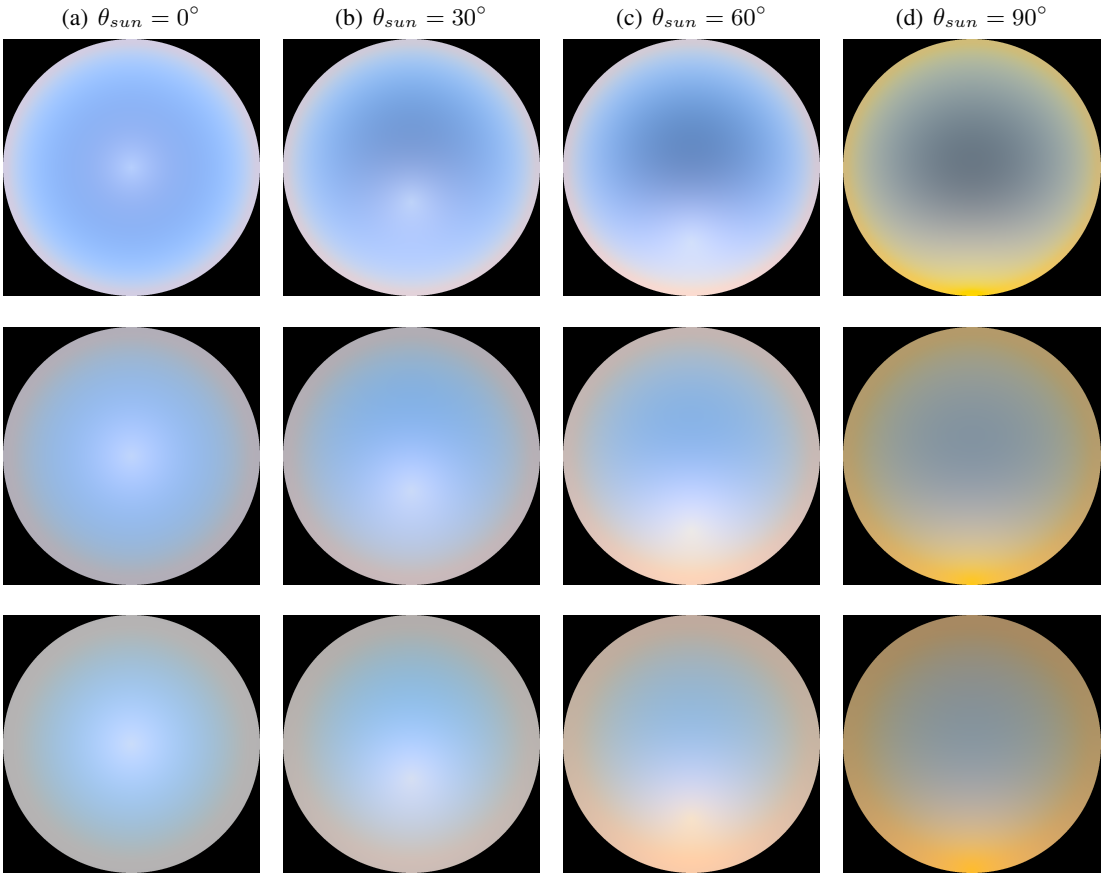


Figure 4.14: Example instances of the Preetham skylight model: the solar zenith angle θ_{sun} increases from left to right, turbidity T increases from top to bottom.

Their contribution is an optimized algorithm to compute for each pixel the coverage caused by whitecaps. Said coverage depends on the amount of self intersections taking place inside the world-space footprint of the pixel on the ocean surface. With the whitecap coverage at hand, Dupuy and Bruneton then proceed to modify the lighting term at each pixel accordingly. The aforementioned self intersections are located using the approach by Tessendorf [1999], whereas the computation of the whitecap coverage is based on fast linear prefiltering methods [Bruneton and Neyret, 2012], allowing it to leverage the hardware-accelerated mipmap generator of modern GPUs.

Tonemapping

Rendering the ocean illuminated by the Preetham sun- and skylight results in an image with a high dynamic range. For proper display of the image, we need to map it to the lower dynamic range of a standard monitor by means of a tonemapping operator. We chose to employ the global tonemapping operator by Reinhard et al. [2002, Equation 4] because it is straightforward

to implement and works well on the class of images our demo application renders. As the dynamic range of the image may change from frame to frame, we found it beneficial to emulate the behavior of the temporal luminance adaptation process of human vision. For simplicity's sake we chose to use the temporal luminance adaptation algorithm by Krawczyk et al. [2005, Equations 5, 6, 7, 12], which gives acceptable results and is easily integrated with the global tonemapping operator by Reinhard et al.

CHAPTER 5

Results

The previous chapter discussed in detail the approach we took to synthesize the animated ocean surface, including, one, a number of optimizations to reduce the computational workload, and, two, a level-of-detail mechanism which does not only give us fine-grained control over model detail, but also allows us to reduce potential tiling artifacts. Additionally, we gave an overview of our demo application, which integrates our approach to ocean surface synthesis with real-time rendering algorithms which have been developed specifically for the display of the open ocean. Thus, with our demo application at hand, we may move on to discuss the actual results we were able to achieve.

The remainder of this chapter is organized as follows: Section 5.1 gives a compact overview of the performance improvements which we were able to attain for the generation of the ocean surface spectra as well as for the computation of the sum of waves via the Inverse Fourier Transform. Section 5.2 delineates the pitfalls we encountered in the context of ocean surface synthesis with wave spectra, and what measures we took to either overcome or sidestep them. Last, Section 5.3 discusses the level of visual fidelity we were able to obtain, with focus on the different wave spectra, as well as our level-of-detail approach.

5.1 Performance

Note: All results of benchmarking shown throughout this section have been collected on an off-the-shelf laptop running an Intel®Core™i5-5300U Mobile Processor [Intel]. All our benchmarking code runs on a single CPU thread and gives the average computation time after repeating the desired task for a thousand iterations.

We have seen in Section 4.1 that the underlying spectrum of the animated ocean surface consists of two parts, a static part and a time-dependent part. As long as the wavevector domain and the wave energy spectrum's parameters do not change, the static part of the spectrum has to be computed only once for the entire lifetime of the animated ocean surface. Moreover, as it

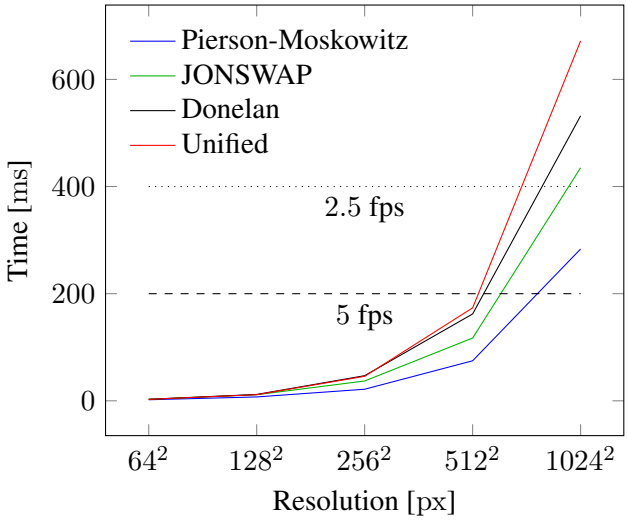


Figure 5.1: Computation times for the static part of the spectrum at various resolutions with different underlying wave energy spectra. One can see that at higher resolutions we would be unable to synthesize the static part of the spectrum more than a few times a second, much less so if we would employ more than one ocean surface pattern. Moreover, one may notice that the computation times of the different wave energy spectra diverge significantly. That is expected, considering the difference in complexity of the wave energy models in question.

is the static part of the spectrum that incorporates the wave energy spectrum, we are relieved of a significant computational burden. This is because evaluating the wave energy spectrum for a large set of wavevectors has proven to be extensive in terms of computational complexity. See Figure 5.1, which gives an overview of measurements we have taken for the computation time of the static part of the spectrum for a single ocean surface pattern.

With the static part of the spectrum on hand in a precomputed form, we are able to compute the spectra of all datasets within a fraction of time so short that we may synthesize at least one animated ocean surface pattern in real time on a single CPU thread. Figure 5.2 gives an overview of the measurements we have taken for the accumulated computation time of all spectral datasets for a single ocean surface pattern. Based on said measurements, one can see that, depending on pattern resolution, there may be computational resources left at our disposal. We may employ these resources to generate additional ocean surface patterns, i.e., levels of detail. As all of our ocean surface patterns are self-contained, we may assume that the computation time to generate l patterns of the same resolution roughly matches the computation time to generate one pattern, multiplied by l. Figure 5.3 corroborates said assumption. Thus, given the computation time of a single pattern, we may easily estimate the maximum number of patterns we are able to generate in real time on a single CPU thread. Still, we are content with a maximum of four patterns.

Due to performance considerations, we chose the IEEE 754 single-precision floating-point format [IEEE, 2008] as the underlying data type for the implementation of our spectrum generation code. Compared to the double-precision format, it simultaneously allowed us to cut our

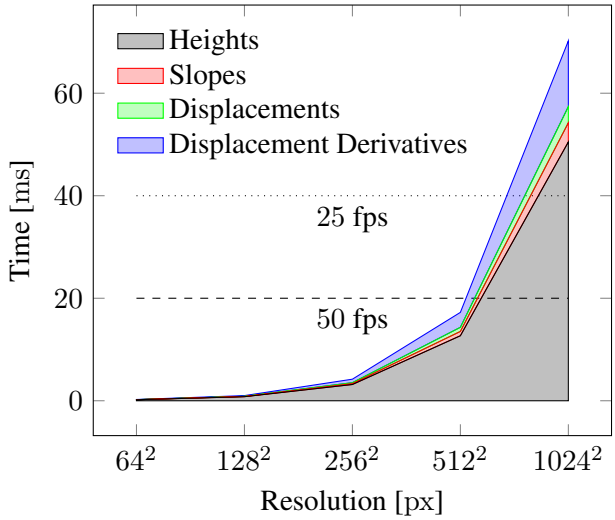


Figure 5.2: Accumulated computation times for the generation of spectral datasets at various resolutions for a single ocean surface pattern. We start at the bottom with the computation times we measured for the height spectrum, and step by step add the timings of the remaining datasets. One can see that the height spectrum alone takes the better part of the entire computation time. That is because all datasets share most of the arithmetic operations. The bulk of the CPU time spent on the supplemental datasets is caused by the additional stores to memory. Moreover, one can see that for all resolutions in question, except the highest one, we are able to generate all spectral datasets of the ocean surface pattern in real time.

memory requirements in half, as well as accelerate arithmetic operations by taking advantage of the C99 [Harbison and Steele, 2002] math library functions tailored to single-precision floating point numbers.

Fourier Transform

Recall, that a single ocean surface pattern consists of up to nine spectra: one for surface elevation η, two for displacement vector $\mathbf{D}$, two for the slopes of η, and four for the first order partial derivatives of $\mathbf{D}$. Moreover, we employ up to four distinct ocean surface patterns. It follows that in the worst case we have to compute the Inverse Discrete Fourier Transform for thirty-six spectra. Depending on pattern resolution, such a number of IDFTs may represent a massive computational burden, especially because all IDFTs have to be computed for each keyframe of the animated ocean surface. Fortunately, all our spectra are Hermitian, and as we have seen in Section 4.2, we may apply the Inverse Discrete Fourier Transform on two Hermitian spectra at once. Therefore the maximum number of required IDFTs for a single ocean surface pattern drops from nine to five: two for the displacement derivatives, one for the displacements, one for the slopes, and one for surface elevation. It follows that for four distinct ocean surface patterns we have to compute only twenty IDFTs instead of the full thirty-six, which represents a decrease of nearly forty-five percent.

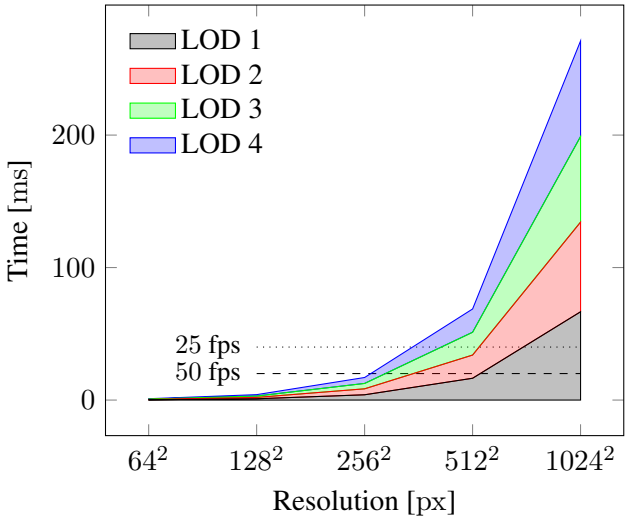


Figure 5.3: Accumulated computation times for the generation of up to four ocean surface patterns at various resolutions, where each pattern consists of all four datasets. For each resolution, the computation times among different LODs diverge only by a small margin. One can see that up to a resolution of 256×256 we are able to generate four ocean surface patterns at a rate of fifty times per second, whereas at a resolution of 512×512 we already have to reduce the number of patterns to two, and still only reach a rate of twenty-five times per second.

With the large number of Inverse Discrete Fourier Transforms taken care of, we have to focus on the IDFT itself to improve performance even further. As stated earlier in Section 4.5, we employ the FFTW library Frigo and Johnson [2005] to compute the Inverse Discrete Fourier Transform. FFTW has support for different floating point precisions, such as IEEE 754 [IEEE, 2008] single-precision, double-precision, as well as quad-precision. Initially we used the double-precision variant of FFTW, but later on we switched to the single-precision variant, FFTWF. The reasons are twofold: first, our spectra are already on hand in single-precision, and second, the FFTW documentation [FFTW, b] states that FFTWF outperforms the double-precision variant roughly by a factor of 1.5. Moreover, we found that for our specific use case, where all spectra have power-of-two resolutions, FFTWF consistently outperforms FFTW by an even larger factor of two. Thus, by switching to single-precision, we are not only able to double the computational performance of the IDFT, but also to keep the low memory requirements of single-precision floating point numbers throughout the entire process of ocean surface synthesis.

For a compact overview of the performance improvements we have been able to achieve for the IDFTs of a single ocean surface pattern, one may look at Figure 5.4. It is important to note that said improvements take effect equally for all the ocean surface patterns we generate, independent of their number. Furthermore, one may inspect Figure 5.5 for a performance comparison between the pattern generation stage and the Inverse Discrete Fourier Transform stage.

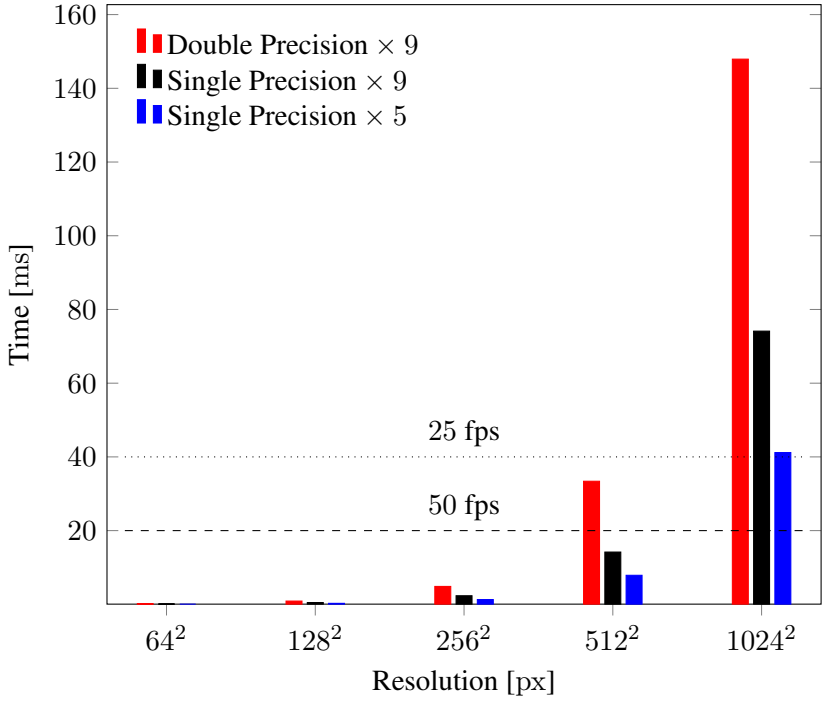


Figure 5.4: Comparison of computation times for the Inverse Discrete Fourier Transform for all nine spectra of a single ocean surface pattern at various resolutions. One can see that the single precision variant of the Inverse Discrete Fourier Transform already reduces the computational workload roughly by a factor of two compared to the double precision variant. Moreover, if we transform pairs of Hermitian spectra at once, we are able to cut the number of required IDFTs from nine to five, further reducing the computational workload by about forty-five percent.

Multi-Resolution

We have seen in Section 4.4 that in the context of our level-of-detail approach we may generate surface elevation η and displacement vector $\mathbf{D}$ at a lower resolution than their respective first order derivatives. We expected the resulting performance gains to be twofold: one, less data to generate, and two, less data to transform via the IDFT. The latter point takes effect as expected, but the former does not. That is because spectrum generation performance is dominated by a set of arithmetic operations which are shared by all datasets. As said arithmetic has to be computed for the datasets with the higher resolution anyway, the only gain in performance arises from a cutback in memory operations. We found that said cutback improves spectrum generation performance by at most ten percent, where the actual improvement is dependent on the difference between the two resolutions of choice. We must admit that ten percent are far from enough to allow us to generate the ocean surface patterns at a significantly higher rate than before, or at least to use the next-higher resolution for the derivative datasets. But still, it is a betterment, and the reduced resolution eases the strain on the subsequent transformation stage considerably. As

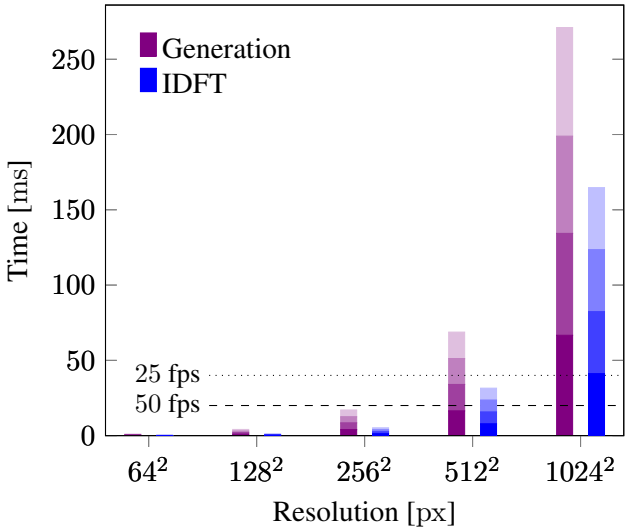


Figure 5.5: Accumulated computation times for up to four ocean surface patterns at various resolutions. For each pattern, all datasets are generated with single-precision and packed into five spectra. Said spectra are then transformed with the IDFT, again with single-precision. One can see that the generation of the spectra actually takes more time than their transformation via the IDFT. Still, up to a resolution of 256×256 we are able to keep real-time framerates for both the generation and the transformation of four ocean surface patterns. Whereas at the next higher resolution, 512×512, only the transformation stage is close to real-time framerates for four patterns.

one may surmise from Figure 5.4, at a resolution of 128×128 or less, the computation time of a single IDFT becomes basically negligible. Thus, in case we employ such a low resolution for surface elevation η and displacements $\mathbf{D}$, then we are left with the computation time required by three IDFTs instead of five, which represents an improvement of forty percent. Although such an improvement to the transformation stage does not change the fact that spectrum generation is the bottleneck of our implementation, it still frees up resources which may be employed elsewhere. Furthermore, with two datasets on hand at a lower resolution, we may obtain some gain in rendering performance because less data needs to be transfered to, and processed by the GPU.

5.2 Ocean Surface Synthesis

Now that we have discussed the performance aspects of our approach to ocean surface synthesis, we may focus on the qualitative aspects. Recall, that we set out to generate and render a visually pleasing, as well as believable ocean surface. Performance aside, we had to tackle two major challenges to reach said goal: first, generate convincing wave geometry, and second, render the generated ocean with a lighting algorithm which gives plausible results. For each of the two tasks we encountered at least one major pitfall, which we will discuss subsequently.

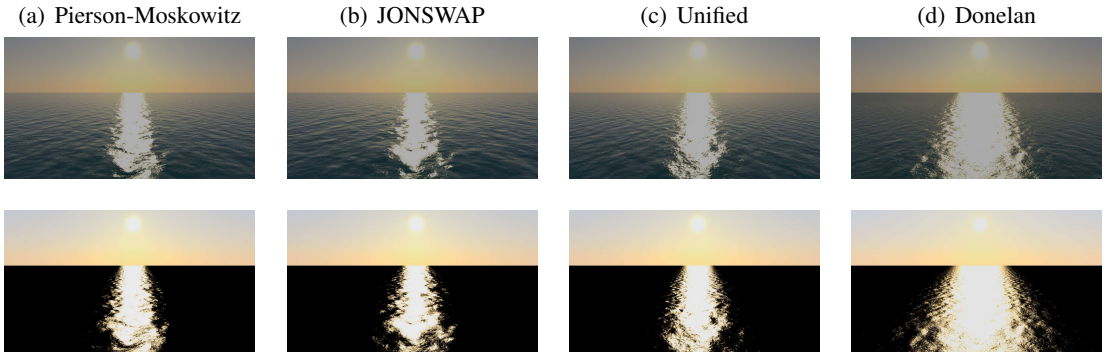


Figure 5.6: Example rendering of the same scene for each of the remaining four spectra ($U_{10} = 10\,\mathrm{m\,s^{-1}}$, $F = 250\,\mathrm{km}$). First row: the ocean surface rendered with the lighting algorithm by Bruneton et al. [2010]. Second row: the respective sun reflectance term. One can see that the sun reflectance of the Donelan spectrum significantly exceeds that of the other spectra.

Wave Geometry

Initially, after having implemented all the wave energy spectra presented in this work, we were dissatisfied with the results. There was always something *off* about the resulting ocean surface, more specifically, there seemed to be a mismatch in scale between the length of the waves and their respective amplitudes. Neither the geometry nor the animation of the ocean surface looked believable to the observer. It was even worse if we applied the choppy-wave algorithm to the ocean surface. Hence, in a first step we tried to mitigate the issue by introducing several fudge factors which allowed us, first, to rescale the ocean surface's geometry during rendering, and second, to directly influence the magnitude of the spectrum's energy output. By carefully tuning said fudge factors, we were at least able to obtain acceptable results. Still, each time we switched to a different wave energy spectrum or modified any of the parameters involved in the synthesis of the ocean surface, we had to adapt our fudge factors too. A most cumbersome and error-prone task for the user. It was only later on that we found the underlying cause of the issue: we were missing the proper integral domain conversions (Section 3.4) for all of our wave energy spectra. After implementing said conversions, the issue simply vanished. We finally got believable, comprehensible as well as consistent results across all wave energy spectra presented in this work, with the only exception being the Phillips spectrum. Since the latter does not model the ocean surfaces' underlying physical processes in an appropriate manner, it is neither consistent with the other spectra nor does it give acceptable results without significant tuning of the fudge factors. Thus, we decided to drop the Phillips spectrum from our implementation, and all of the now unused fudge factors with it.

Ocean Surface Lighting

Now that we have dropped the Phillips spectrum, we are left with four wave energy spectrum models. Although all of them give convincing results geometry-wise, it is only three of them

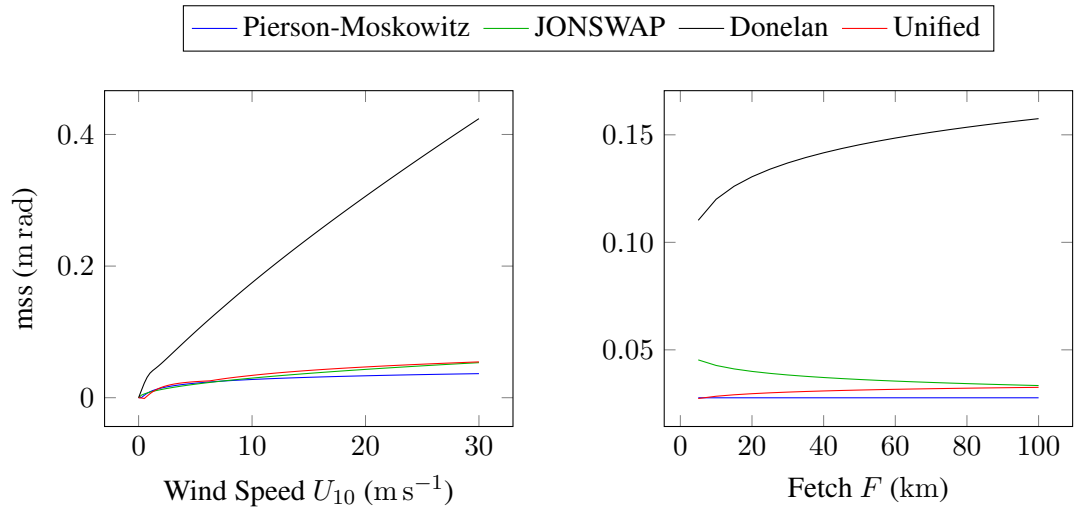


Figure 5.7: Left: the total mean square slope mss over wind speed U_{10} ($F = 250\,\mathrm{km}$). One can see that the Donelan spectrum severely overestimates the total mean square slope of the ocean surface. Right: the total mean square slope over fetch F ($U_{10} = 10\,\mathrm{m\,s^{-1}}$). One can see that under low-fetch conditions the JONSWAP spectrum overestimates the total mean square slope too, but not by such a large amount as the Donelan spectrum does.

that work well with the lighting algorithm by Bruneton et al. [2010]: the Pierson-Moskowitz spectrum, the JONSWAP spectrum, and the Unified spectrum. The fourth spectrum, namely the Donelan spectrum, is not eligible because in combination with Bruneton et al. [2010] it results in an ocean surface which reflects too much of the incoming sunlight. Figure 5.6 depicts an example of said shortcoming. We found that the underlying cause is that the Donelan spectrum does not give a good approximation of the ocean surface's actual *mean square slopes* (Appendix A.1). The mean square slopes of the ocean surface represent statistical moments which describe the distribution of the surface waves' slopes [Massel, 2011], and as such give a measure of the roughness of the ocean surface. Moreover, the mean square slopes are an essential component of the lighting model by Bruneton et al. [2010], where they are required for the computation of the sun reflectance term.

Given the importance of the mean square slopes with respect to the quality of the visual appearance of the ocean surface, we came to the conclusion that it would be best to employ only wave energy spectra which give an as good as possible approximation of the mean square slopes for the entire parameter space of wind speed U_{10} and fetch F. Figure 5.7 shows that the Donelan spectrum unconditionally fails to meet this specific requirement. The JONSWAP spectrum, on the other hand, tends to overestimate the mean square slopes only for low-fetch conditions, but not by an amount so large that it would noticeably compromise the plausibility of the end result. But still, that leaves only two of the five wave energy models presented in this work to be unconditionally able to measure up to the challenge, namely the Pierson-Moskowitz spectrum and the Unified spectrum.

5.3 Visual Fidelity

Now that we have taken care of the pitfalls which we encountered during our work on ocean surface generation, we may focus on the overall visual quality we were able to achieve. Let us begin with the most fundamental building block, namely the wave spectrum. Given any one of the wave energy spectra presented in this work that stem from oceanographic research, one is able to synthesize an ocean surface where the shape and the animation of the waves does not only look plausible, but convincing to the observer. Thus, the only remaining question left for the user is what kind of sea does he want to generate, and which spectrum would best fit his choice. We may give some direction on that matter. If one has a gentle sea in mind, then it would be best to use the Pierson-Moskowitz spectrum, whereas for highly turbulent seas one should prefer the JONSWAP spectrum. For all the kinds of sea that reside between those two extrema, the Unified spectrum is the wave energy model of choice. Even more so, because among the spectra presented in this work it is the only one to handle gravity-capillary waves in a sensible manner, and is therefore the one best suited for closeups where short waves are visible to the observer.

Now, given the wave spectrum of our choosing, we synthesize all datasets for the desired number of ocean surface patterns and use them to render the ocean. We have found that two complementary ocean surface patterns provide sufficient visual quality for most viewing conditions. For closeups, though, it has proven beneficial to include at least one additional pattern which adds short wave detail to the water surface. Also for wide spanning camera views, where the ocean is visible up to the horizon, we settled not for two, but for three to four patterns to keep tiling artifacts in check. As for pattern resolution, we chose 256×256 because it is the highest resolution which still allows us to keep real-time framerates with four active patterns.

Once we are satisfied with the visual appearance of the ocean for a specific number of patterns, we may employ our multi-resolution approach to lighten the computational load of CPU and GPU. By simple visual inspection we have found that for most scenes we are able to reduce the resolution of the horizontal and vertical displacements as far as to 32×32 without noticing a significant drop in visual quality, on condition that we keep a resolution of 256×256 for the slopes and the first-order derivatives. With surface heights and displacements synthesized at such a low resolution, the computational cost required by them shrinks to nearly zero, freeing up resources which we may employ elsewhere. Still, there are cases where the observer may notice a distinct drop in visual quality: one, in closeups, and two, in scenes with highly turbulent seas where the low resolution may deny the waves their distinctive narrow peaks.

Now that we have discussed the factors which have most impact on the visual fidelity of the result, we may give an overview of the variety of ocean surfaces which we are able to display by means of this work, see Figure 5.8. For a rough breakdown of our approach to ocean surface lighting, one may look at Figure 5.9, which shows the actual mesh representation of the ocean, all involved lighting terms, and the final result. Figure 5.10, on the other hand, depicts an example instance of the ocean surface tiles.

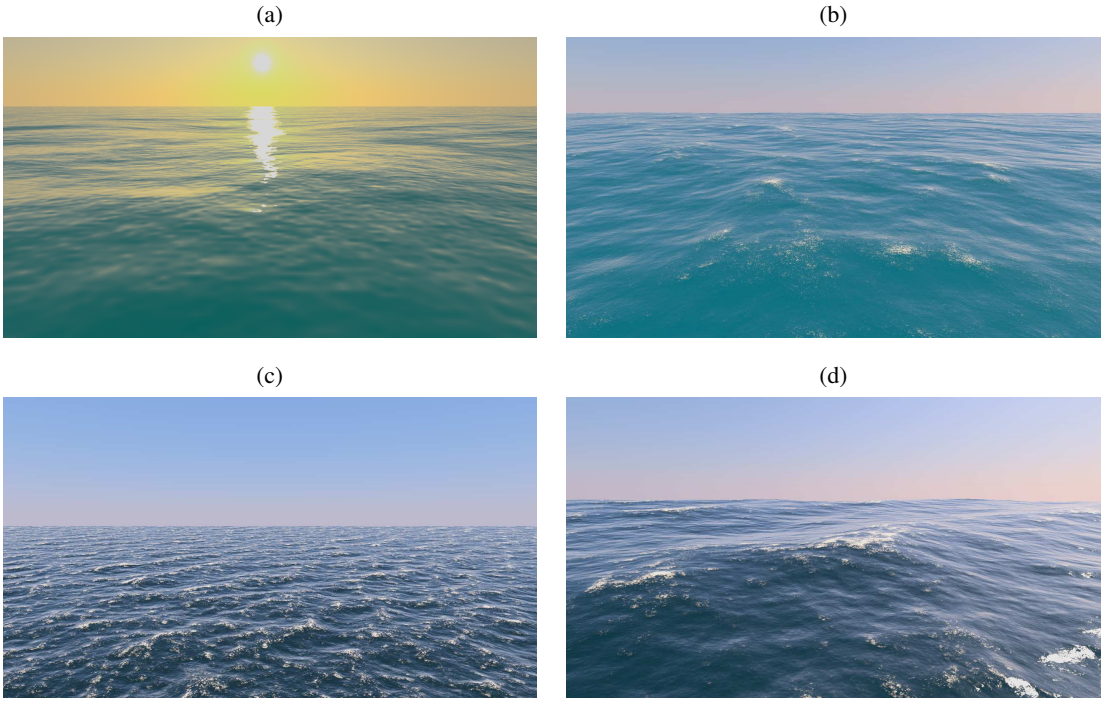
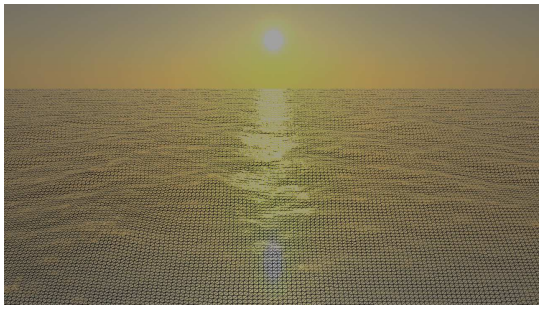
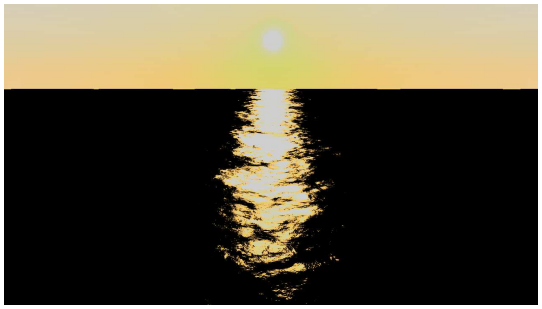


Figure 5.8: The ocean surface under a variety of conditions. (a) A calm sea with the camera looking into the sunset (Unified spectrum, four patterns). (b) A fully developed sea with medium sized waves. (Pierson-Moskowitz spectrum, two patterns). (c) A highly agitated developing sea with small waves (JONSWAP spectrum, three patterns). (d) A fully developed sea with large, sharp crested waves (Unified spectrum, three patterns).

(a) Projected Grid

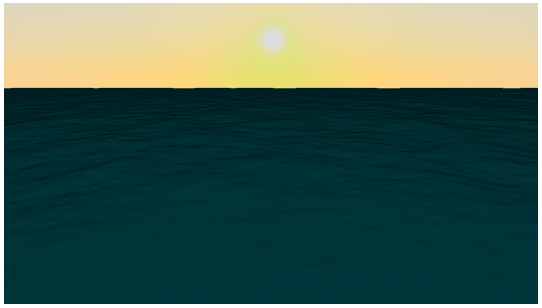


(b) Sun Contribution

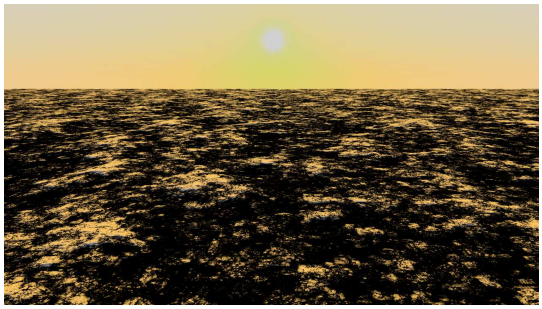


(c) Sky Contribution

(d) Sea Contribution



(e) Whitecaps Contribution



(f) Final Result

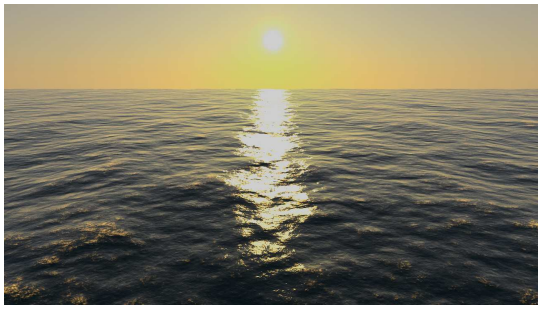


Figure 5.9: An overview of the meshing scheme and all ocean lighting terms involved in our implementation. (a) The ocean surface as represented by the projected grid. (b) Sun light reflected by the water surface. (c) Sky light reflected by the water surface. (d) Light refracted from the water body. (e) Whitecap foam. (f) The final result.

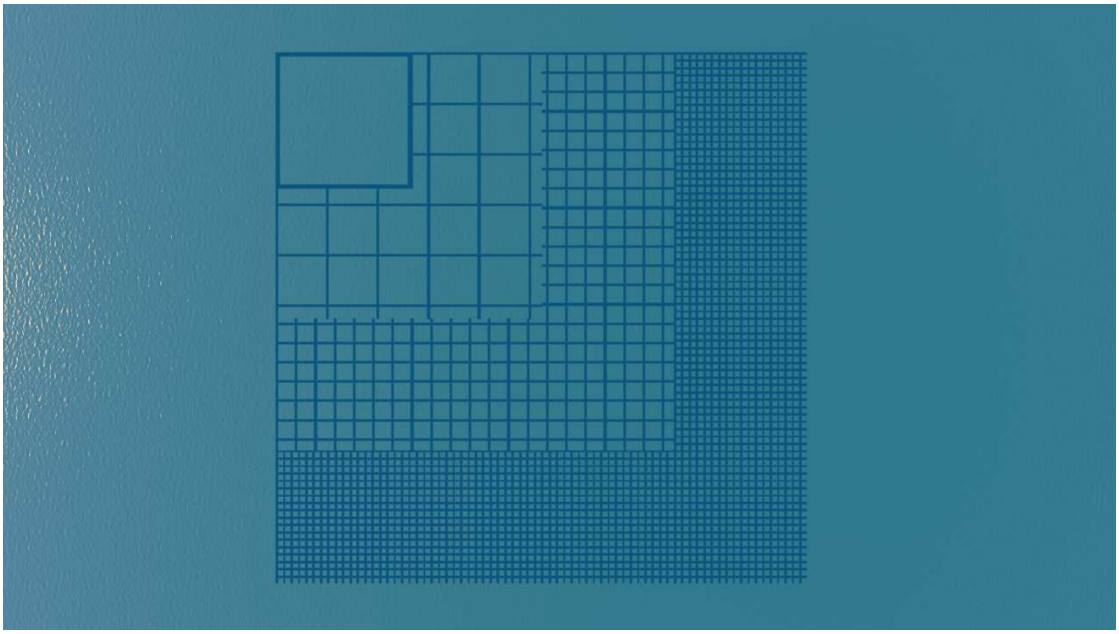


Figure 5.10: A top down view of the ocean overlayed with a simple depiction of its four pattern sizes.

CHAPTER 6

Summary

We stated in the introduction that in this work we seek to synthesize animated ocean surfaces which are both believable and computationally feasible. To achieve said objective, we picked the choppy wave algorithm by Tessendorf [1999], which adopts the wave spectrum concept from oceanographic research for computer graphics purposes. Tessendorf approximates the perturbations produced by the water surface by sampling the wave spectrum in frequency space via the Fast Fourier Transform algorithm. The FFT algorithm is highly efficient and generates a perturbation pattern which seamlessly tiles the ocean surface. An additional upside of the wave spectrum approach is that oceanographic research provides a wide range of wave spectra which are specifically tailored to the deep water of the open ocean. Said wave spectra are employed for oceanographic forecasts and simulations, therefore they give a high degree of realism. Thus, we chose to implement four such wave-energy spectra, all well established in oceanographic literature, plus the wave spectrum introduced by Tessendorf (Section 3.4). The latter spectrum was unable to produce results which meet the high standard given by the other spectra, and therefore we dropped it (Section 3.4 and 5.2). The remaining four spectra, on the other hand, allowed us to obtain animated wave geometry at the level of quality we sought after, but only after we had made sure that each wave spectrum had been converted to the correct integral domain (Section 3.4).

For the purpose of ocean surface lighting, we decided to implement the algorithm by Eric Bruneton [2010], as it gives believable results and it has specifically been tailored to the wave spectrum of the open ocean. After a successful implementation, it turned out that only two of our four remaining wave spectra were able to unconditionally match the algorithm's stringent requirements, with a third spectrum keeping up under most conditions (Section 5.2). Furthermore, we implemented the work by Dupuy and Bruneton [2012] to incorporate whitecaps into the ocean surface's lighting model.

With ocean surface synthesis and lighting taken care of, we found that as long as we employ only one perturbation pattern, we may not be able to reproduce a high-quality ocean surface for all possible viewing situations. Depending on the observer's viewpoint, we may be faced with tiling artifacts or lack of detail. Therefore, we adopted the approach by Eric Bruneton [2010],

which addresses both matters by superimposing multiple, complementary perturbation patterns of different size (Section 4.4). With multiple patterns at hand, where for each pattern we have to generate and Fourier-transform up to nine separate spectra (one for vertical perturbation, two for horizontal perturbation, two for gradients, four for whitecaps), pattern generation had become computationally too expensive to still allow for real-time operation. Therefore we took a number of measures to restore real-time behaviour. First, we split surface synthesis into two separate threads, where the first one generates the pattern's actual spectra, and the second one applies the Inverse Fourier Transform on said spectra (Section 4.5). Second, as the spectrum consists of a static part and a time-dependent part, we made sure to compute only the time-dependent part of the spectrum for each frame (Section 4.1). Third, as all of our spectra are Hermitian, we were able to compute the Inverse Fourier Transform of two spectra at once (Section 4.2). Fourth, we switched the Inverse Fourier Transform from double precision to single precision, accelerating its computation by a factor of two (Section 5.1).

With performance back to real-time framerates, we improved upon the work by Eric Bruneton [2010] with a multi-resolution variant, where the water surface's wave geometry is synthesized at a lower resolution than its respective normal vectors and whitecaps (Section 4.4). The lower resolution lead to a considerable speedup of the Fourier Transform stage, but not to a significant degradation in visual quality for most scenes (Section 5.3). Thus we decided to permanently keep the multi-resolution approach, as it allows for an even more fine-grained balance between model detail and performance.

6.1 Contributions

Now that we have given a compact summary of our work, we may highlight our contributions. First and foremost, we gave an in-depth discussion of five distinct wave spectrum models. For each wave spectrum we determined three key aspects: is it able to generate believable ocean geometry, for what kind of sea is it most adequate for, and is it suitable for the lighting scheme in Eric Bruneton [2010]. Second, instead of using the projected grid variant by Hinsinger et al. [2002], as Eric Bruneton [2010] did, we chose to employ the enhanced version by Johanson [2004], as it allows our camera full freedom of movement. Last, we improved upon Dupuy and Bruneton [2012]'s implementation of the choppy wave algorithm with a multi-resolution variant, allowing for an increase in performance without a significant drop in visual quality for most scenes. Moreover, although not a contribution in and by itself, we have given an in-depth discussion of the theoretical background involved in the ocean surface synthesis algorithm and its respective performance optimizations, including the aspects which have been touched only superficially by the works of Eric Bruneton [2010], Tessendorf [1999], and Dupuy and Bruneton [2012].

6.2 Future Work

Now that we have summarized our work and highlighted our contributions, we may point out a few potential improvements. One could take the next step in ocean surface synthesis and implement support for wave spectra which incorporate shallow water, such as the TMA spectrum

[Hughes, 1984]. In that case, one would be required to use a dispersion relation which models both shallow and deep water, including the necessary follow-up work to adapt the wave energy spectrum's integral domain conversions to the new dispersion relation [Horvath, 2015]. Wave spectra aside, one could significantly improve upon performance by following the lead of Eric Bruneton [2010] and generate all surface patterns on the GPU, as well as compute all Fourier Transforms on the GPU, while the static part of the underlying wave spectrum would still be synthesized on the CPU. Additionally, it would be beneficial to implement an alternative to the whitecap algorithm by Dupuy and Bruneton [2012], because the latter is computationally highly expensive, especially with regard to pattern synthesis.

6.3 Concluding Remarks

We have shown the integration of the wave spectrum concept into a coherent framework for the real-time display of believable ocean surfaces. Specific properties of the wave spectrum allow us, first, to synthesize the water surface with high efficiency, and second, to strike a well adjusted balance between model detail and performance. Still, one has to be careful when picking a specific wave spectrum model, as not all models are adequate for real-time rendering purposes.

APPENDIX A

Appendix

A.1 Mean Square Slopes

The ocean lighting algorithm by Eric Bruneton [2010] requires us to compute the mean square slopes of the wave energy spectrum $\Theta(\mathbf{k})$. Let $\mathbf{k} = (k_x, k_z)$ be the wavevector, then the mean square slope in the upwind direction, mss_x, and the mean square slope in the crosswind direction, mss_z, are defined as follows:

$$mss_x = \iint_{\mathbf{k}} k_x^2 \Theta(\mathbf{k}) \mathrm{d}\mathbf{k} \qquad\qquad mss_z = \iint_{\mathbf{k}} k_z^2 \Theta(\mathbf{k}) \mathrm{d}\mathbf{k} \tag{A.1}$$

The total mean square, on the other hand, is independent of direction and defined as follows:

$$mss = mss_x + mss_z = \iint_{\mathbf{k}} \|\mathbf{k}\|^2 \, \Theta(\mathbf{k}) \mathrm{d}\mathbf{k} = \int_0^\infty k^2 \Theta(k) \mathrm{d}k \tag{A.2}$$

List of Figures

List of Tables

Bibliography

G.B. Airy. *Tides and Waves: Extracted from the Encyclopaedia Metropolitana, Tom. V Pag. 241 - 396*. William Clowes and Sons, 1845.

Jose Henrique GM Alves, Michael L Banner, and Ian R Young. Revisiting the Pierson-Moskowitz asymptotic limits for fully developed wind waves. *Journal of physical oceanography*, 33(7):1301–1323, 2003.

G.K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library. Cambridge University Press, 2000.

F. Biesel. Study of wave propagation in water of gradually varying depth. In *Gravity Waves*, page 243, November 1952.

R.N. Bracewell. *The Fourier Transform and Its Applications*. Electrical engineering series. McGraw-Hill Higher Education, 2000.

C. L. Bretschneider. The generation and decay of wind waves in deep water. *Trans. Am. Geophys. Union*, 33(3):381–389, 1952.

R. Bridson. *Fluid Simulation for Computer Graphics, Second Edition*. Taylor & Francis, 2015.

Eric Bruneton and Fabrice Neyret. Precomputed atmospheric scattering. In *Proceedings of the Nineteenth Eurographics Conference on Rendering*, EGSR '08, pages 1079–1086. Eurographics Association, 2008.

Eric Bruneton and Fabrice Neyret. A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics*, 18 (2):242–260, February 2012.

Eric Bruneton, Fabrice Neyret, and Nicolas Holzschuch. Real-time realistic ocean lighting using seamless transitions from geometry to brdf. *Computer Graphics Forum*, 29(2):487–496, 2010.

Lining Chen, Yicheng Jin, and Yong Yin. Ocean wave rendering with whitecap in the visual system of a maritime simulator. *CIT*, 25(1):63–76, 2017.

U.S. Coastal Engineering Research Center. *Shore Protection Manual*. Shore Protection Manual. Department of Defense, Department of the Army, Coastal Engineering Research Center, 1977.

James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

Charles Cox and Walter Munk. Measurement of the Roughness of the Sea Surface from Photographs of the Sun's Glitter. *Journal of the Optical Society of America*, 44(11):838–850, November 1954.

Xie Cui, Jin Yi-cheng, and Liu Xiu-wen. Real-time ocean wave in multi-channel marine simulator. In *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 332–335. ACM, 2004.

M. A. Donelan, J. Hamilton, and W. H. Hui. Directional spectra of wind-generated waves. *Phil. Trans. Roy. Soc. London A*, 315:509–562, 1985.

Mark A. Donelan, Fred W. Dobson, Stuart D. Smith, and Robert J. Anderson. On the Dependence of Sea Surface Roughness on Wave Development. *Journal of Physical Oceanography*, 23:2143–2152, 1993.

Jonathan Dupuy and Eric Bruneton. Real-time animation and rendering of ocean whitecaps. In *SIGGRAPH Asia 2012 Technical Briefs*, SA '12, pages 15:1–15:3. ACM, 2012.

T. Elfouhaily, B. Chapron, K. Katsaros, and D. Vandemark. A unified directional spectrum for long and short wind-driven waves. *J. Geophys. Res.*, 102(C7):15781–15796, 1997.

Eric Bruneton. Real-time realistic ocean lighting using seamless transitions from geometry to brdf. http://www-evasion.imag.fr/people/Eric.Bruneton/OceanLightingFFT.zip, 2010. Accessed: 25.09.2015.

Leonhard Euler. *Introductio in analysin infinitorum*, volume 2. MM Bousquet, 1748.

FFTW. FFTW – Fastest Fourier Transform in the West. `http://www.fftw.org`, a. Last Accessed: 2017-08-21.

FFTW. Fftw. `http://www.fftw.org/speed/`, b. Last Accessed: 2017-08-21.

Mark Finch. Effective water simulation from physical models. In Randima Fernando, editor, *GPU Gems*, pages 5–29. Addison-Wesley, 2004.

Alain Fournier and William T. Reeves. A simple model of ocean waves. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 75–84. ACM, 1986.

Jocelyn Fréchot. Realistic simulation of ocean surface using wave spectra. *JVRB - Journal of Virtual Reality and Broadcasting*, 4(2007)(11), 2007.

Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

Matteo Frigo and Steven G. Johnson. *FFTW*. Massachusetts Institute of Technology, 2017. Last Accessed: 2017-08-21.

Franz Gerstner. Theorie der Wellen. *Annalen der Physik*, 32(8):412–445, 1809.

GLFW. Glfw. `http://www.glfw.org`.

GNUstep. Gnustep. `http://www.gnustep.org`.

Jean-Christophe Gonzato and Bertrand Le Saëc. *A phenomenological model of coastal scenes based on physical considerations*, pages 137–148. Springer Vienna, 1997.

Jean-Christophe Gonzato and Bertrand Le Saëc. On Modeling and Rendering Ocean Scenes. *Journal of Visualisation and Computer Animation*, pages 27–37, 2000.

G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Probability and Random Processes. OUP Oxford, 2001.

Tetsu Hara, Erik J. Bock, and David Lyzenga. In situ measurements of capillary-gravity wave spectra using a scanning laser slope gauge and microwave radars. *Journal of Geophysical Research: Oceans*, 99(C6):12593–12602, 1994.

Samuel P. Harbison and Guy L. Steele. *C: A Reference Manual*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 5th edition, 2002. ISBN 013089592X.

K. Hasselman, T. P. Barnett, E. Bouws, D. E. Carlson, and P. Hasselmann. Measurements of wind-wave growth and swell decay during the joint north sea wave project (jonswap). *Deutsche Hydrographische Zeitschrift*, 8(12), 1973.

D. E. Hasselmann, M. Dunckel, and J. A. Ewing. Directional wave spectra observed during JONSWAP 1973. *J. Phys. Oceanogr.*, 10:1264–1280, 1980.

Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. Interactive animation of ocean waves. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, pages 161–166. ACM, 2002.

Christopher J. Horvath. Empirical directional wave spectra for computer graphics. In *Proceedings of the 2015 Symposium on Digital Production*, DigiPro '15, pages 29–39. ACM, 2015.

Steven A. Hughes. *The TMA shallow-water spectrum description and applications [microform] / by Stevens A. Hughes*. Dept. of the Army, US Army Corps of Engineers ; Available from National Technical Information Service Washington, DC : [Springfield, Va, 1984.

C. Huygens. *Treatise on Light*. tredition, 2012.

U.S. Hydrographic Office. *Breakers and Surf: Principles in Forecasting*. H.O. pub. Hydrographic Office under the authority of the Secretary of the Navy, 1944.

U.S. Hydrographic Office. *Supplement to Breakers and Surf: Principles in Forecasting*. H.O. pub. Hydrographic Office under the authority of the Secretary of the Navy, 1945.

IEEE. IEEE standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008.

Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. SPH Fluids in Computer Graphics. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014.

Intel. Intel®core™i5-5300u processor. `https://ark.intel.com/en/products/85213/Intel-Core-i5-5300U-Processor-3M-Cache-up-to-2_90-GHz`. Accessed: 2017-08-21.

John Isidoro, Alex Vlachos, and Chris Brennan. Rendering ocean water. *Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks, Wordware*, 2002.

Stefan Jeschke and Chris Wojtan. Water wave animation via wavefront parameter interpolation. *ACM Trans. Graph.*, 34(3):27:1–27:14, May 2015.

B. Jähne and K. Riemer. Two-dimensional wave number spectra of small-scale water surface waves. *J. Geophys. Res.*, 95(C7):11531–11646, 1990.

Claes Johanson. Real-time water rendering - introducing the projected grid concept. Master's thesis, Department of Computer Science, Lund University, 2004.

Steven G. Johnson. Notes on fft-based differentiation. `http://math.mit.edu/~stevenj/fft-deriv.pdf`, 2011. Accessed: 27.02.2016.

Julie Karre. Teacher at Sea: Aboard NOAA Ship Oregon II. `https://teacheratsea.files.wordpress.com/2013/08/p1020149.jpg`, August 2013. [Digital Photograph; accessed February 1, 2017].

B. Kinsman. *Wind Waves: Their Generation and Propagation on the Ocean Surface*. Dover Phoenix Editions. Dover Publications, 2002.

S. A. Kitaigorodskii. Applications of the theory of similarity to the analysis of wind-generated wave motion as a stochastic process. *Izv. Geophys. Ser. Acad. Sci., USSR*, 1:105–117, 1962.

S. A. Kitaigorodskii. *The physics of air-sea interaction*. Israel Program for Scientific Translations, 1970.

G.J. Komen, L. Cavaleri, M. Donelan, K. Hasselmann, S. Hasselmann, and P.A.E.M. Janssen. *Dynamics and Modelling of Ocean Waves*. Cambridge University Press, 1996.

Grzegorz Krawczyk, Karol Myszkowski, and Hans-Peter Seidel. Perceptual effects in real-time tone mapping. In *Proceedings of the 21st Spring Conference on Computer Graphics*, SCCG '05, pages 195–202. ACM, 2005.

Yuri Kryachko. Using vertex texture displacement for realistic water rendering. *GPU Gems*, 2: 283–294, 2005.

H. Lamb. *Hydrodynamics*. Dover Books on Physics. Dover publications, 1945.

Nam-Kyung Lee, Nakhoon Baek, and Kwan Woo Ryu. Real-time simulation of surface gravity ocean waves based on the tma spectrum. In Yong Shi, G. Dick van Albada, Jack Dongarra, and Peter M. A. Sloot, editors, *International Conference on Computational Science (2)*, volume 4488 of *Lecture Notes in Computer Science*, pages 122–129. Springer, 2007.

M. J. Lighthill. *An Introduction to Fourier Analysis and Generalised Functions:*. Cambridge University Press, 1958.

Stanisław R. Massel. On the geometry of ocean surface waves. *Oceanologia*, 53(2):521 – 548, 2011.

G. A. Mastin, P. A. Watterberg, and J. F. Mareda. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications*, 7(3):16–23, March 1987.

MATLAB. Matlab. `http://www.mathworks.com/products/matlab/`. The MathWorks, Natick, MA, USA.

Nelson L. Max. Vectorized procedural models for natural terrain: Waves and islands in the sunset. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '81, pages 317–324. ACM, 1981.

C.C. Mei. *The Applied Dynamics of Ocean Surface Waves*. Advanced series on ocean engineering. World Scientific, 1989.

Jason L Mitchell. Real-time synthesis and rendering of ocean water. *ATI ResearchTechnical Report*, pages 121–126, 2005.

H. Mitsuyasu, F. Tasai, T. Suhara, S. Mizuno, M. Ohkusu, T. Honda, and K. Rikiishi. Observations of the Directional Spectrum of Ocean Waves Using a Cloverleaf Buoy. *Journal of Physical Oceanography*, 5:750, 1975.

Walter Munk. Origin and generation of waves. *Coastal Engineering Proceedings*, 1(1):1, 2010.

Gerhard Neumann and Willard J Pierson Jr. *Principles of physical oceanography*. Englewood Cliffs, NJ (USA) Prentice Hall, 1966.

NOAA. NOAA's National Weather Service (NWS) Collection. `http://www.photolib.noaa.gov/htmls/wea00816.htm`, Winter 1989. [Digital Photograph; accessed February 1, 2017].

NVIDIA. Ocean surface simulation. NVIDIA Graphics SDK 11 Direct3D, 2011.

M.K. Ochi. *Ocean Waves: The Stochastic Approach*. Cambridge Ocean Technology Series. Cambridge University Press, 2005.

OpenGL. Opengl. http://www.opengl.org, a.

OpenGL. Opengl. https://www.opengl.org/registry/doc/glspec33.compatibility.20100311.pdf, b.

Darwyn R. Peachey. Modeling waves and surf. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 65–74. ACM, 1986.

Ken Perlin. An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 287–296. ACM, 1985.

O. M. Phillips. The equilibrium range in the spectrum of wind-generated waves. *Journal of Fluid Mechanics*, 4:426–434, 8 1958.

O. M. Phillips. Spectral and statistical properties of the equilibrium range in wind-generated gravity waves. *J. Fluid Mech.*, 156:505–531, 1985.

Willard J. Pierson and Lionel Moskowitz. A proposed spectral form for fully developed wind seas based on the similarity theory of S. A. Kitaigorodskii. *J. Geophys. Res.*, 69(24), December 1964.

A. J. Preetham, Peter Shirley, and Brian Smits. A practical analytic model for daylight. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 91–100. ACM Press/Addison-Wesley Publishing Co., 1999.

Simon Premoze and Michael Ashikhmin. Rendering natural waters. In *8th Pacific Conference on Computer Graphics and Applications (PG 2000), 3-5 October 2000, Hong Kong, China*, pages 23–30, 2000.

W.H. Press. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

William John Macquorn Rankine. On the exact form of waves near the surface of deep water. *Philosophical Transactions of the Royal Society of London*, 153:127–138, 1863.

Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 267–276. ACM, 2002.

Vincent Ross, Denis Dion, and Guy Potvin. Detailed analytical approach to the gaussian surface bidirectional reflectance distribution function specular component applied to the sea surface. *JOSA A*, 22(11):2442–2453, 2005.

Björn Rydahl. A vfx ocean toolkit with real time preview. 2009.

Bruce Schachter. Long crested wave models. *Computer Graphics and Image Processing*, 12(2): 187 – 201, 1980.

Jens Schneider and Rüdiger Westermann. Towards real-time visual simulation of water surfaces. In *Proceedings of the Vision Modeling and Visualization Conference 2001*, VMV '01, pages 211–218. Aka GmbH, 2001.

Helen Smith. Ocean exploration 2020 photo contest. `http://oceanexplorer.noaa.gov/oceanexploration2020/photocontest/finalists/scenic/smith.html`, June 2012. [Digital Photograph; accessed February 1, 2017].

Winthrop W. Smith and Joanne M. Smith. *Handbook of Real-Time Fast Fourier Transforms: Algorithms to Product Testing*. Wiley-IEEE Press, 1st edition, 1997.

H. U. Sverdrup and W. H. Munk. Wind, sea, and swell: theory of relations for forecasting. Technical Report 601, U. S. Hydrographic Office, March 1947.

Jerry Tessendorf. Simulating ocean water. In *SIGGRAPH course notes*. ACM, 1999.

S. Thon and D. Ghazanfarpour. Ocean waves synthesis and animation using real world information. *Computers & Graphics*, 26(1):99–108, 2002.

S. Thon, J. M. Dischler, and D. Ghazanfarpour. Ocean waves synthesis using a spectrum-based turbulence function. In *Computer Graphics International, 2000. Proceedings*, pages 65–72, 2000.

Lloyd N. Trefethen. *Spectral Methods in MatLab*. Society for Industrial and Applied Mathematics, 2000.

Pauline Y. Ts'o and Brian A. Barsky. Modeling and rendering waves: Wave-tracing using beta-splines and reflective and refractive texture mapping. *ACM Trans. Graph.*, 6(3):191–214, July 1987.

Theodore von Kármán. Mechanical similitude and turbulence. *NACA TM 611*, 1931.

I.R. Young. *Wind Generated Ocean Waves*. Elsevier Ocean Engineering Series. Elsevier Science, 1999.