Chapter 1
Simulation of standing and propagating sea waves with three-dimensional ARMA model

Ivan Gankevich, Alexander Degtyarev

1.1 Introduction

Studying behaviour of a ship at sea is often based on some model of external excitations — any disturbance that displaces the vessel from equilibrium — major component of which is wind waves. Currently, the most popular sea wave simulation models are based on the linear expansion of a stochastic moving surface as a system of independent random variables. Such models were studied by St. Denis & Pearson [1], Rosenblatt [2], Sveshnikov [3], and Longuet—Higgins [4]. The most popular model is that of Longuet—Higgins (LH), which approximates propagating sea waves as a superposition of elementary harmonic waves with random phases ε_n and random amplitudes c_n:

$$\zeta(x,y,t) = \sum_n c_n \cos(u_n x + v_n y - \omega_n t + \varepsilon_n), \tag{1.1}$$

where the wave number (u_n, v_n) is continuously distributed on the (u, v) plane, i.e. the unit area $du \times dv$ contains an infinite number of wave numbers. The frequency ω_n associated with wave numbers (u_n, v_n) is given by a dispersion relation

$$\omega_n = \omega(u_n, v_n).$$

The phase ε_n are jointly independent random variables uniformly distributed in the interval $[0, 2\pi]$.

Longuet—Higgins showed that under the above conditions, the function $\zeta(x,y,t)$ is a three-dimensional steady-state homogeneous ergodic Gaussian

Ivan Gankevich
Saint Petersburg State University, Universitetskii prospekt 35, Petergof, Saint Petersburg, Russia 198504, e-mail: i.gankevich@spbu.ru

Alexander Degtyarev
Saint Petersburg State University, Universitetskii prospekt 35, Petergof, Saint Petersburg, Russia 198504, e-mail: a.degtyarev@spbu.ru

field, defined by

$$2E_\zeta(u,v)dudv = \sum_n c_n^2,$$

where $E_\zeta(u,v)$ is two-dimensional spectral density of wave energy.

Formula (1.1) is derived from equation of continuity and equation of motion for incompressible inviscid fluid. For ocean waves incompressibility and isotropy of a fluid is assumed; since the motion of ocean waves is due to gravitational forces, irrotational motion of the fluid is assumed which let us introduce the velocity potential ϕ. Under these assumptions the equation of continuity reduces to Laplace equation:

$$\Delta\phi = \frac{\partial^2\phi_x}{\partial x^2} + \frac{\partial^2\phi_y}{\partial y^2} + \frac{\partial^2\phi_z}{\partial z^2} = 0.$$

The Laplace equation is linear and its solution can be found using Fourier transforms. Thus, for plane waves a well-known solution is given in the form of a definite integral [5]:

$$\phi(x,z,t) = \int\limits_0^\infty e^{kz}\left[A(k,t)\cos kx + B(k,t)\sin kx\right]dk.$$

A similar, but slightly more complicated solution is obtained for the three-dimensional case. The constants A and B are determined from the boundary conditions on the surface. In the linear formulation the equation of the wave profile (which is derived from linearised kinematic boundary condition and equation of motion, see sec. 1.4) is

$$\begin{aligned}\zeta(x,t) &= -\frac{1}{g}\frac{\partial\phi(x,0,t)}{\partial t} \\ &= \int\limits_0^\infty \left[\frac{\partial A(k,t)}{\partial t}\cos kx + \frac{\partial B(k,t)}{\partial t}\sin kx\right]dk \\ &= \int\limits_0^\infty C_t(k,t)\cos\left(kx + \varepsilon(k,t)\right).\end{aligned} \tag{1.2}$$

If we set $c_n = C_t(k_n,t)dk$, then wave model (1.1) may be associated with an approximation of integral (1.2).

Although, LH model is based on simple linear wave theory and has straightforward computational algorithm, it has some serious shortcomings.

- LH model is designed to represent a stationary Gaussian field. Normal distribution of the simulated process (1.1) is a consequence of the central limit theorem: its application to the analysis of storm or shallow water waves represents a significant challenge.

- LH model is periodic and need a large set of frequencies to perform long-term simulation.
- In the numerical implementation of the LH model, it appears that convergence rate of (1.1) is slow. This leads to a skewed simulated wave energy spectrum and skewed cumulative distribution functions of various wave parameters (heights, lengths, etc.). This problem becomes especially significant when simulating complex sea waves that have a wide spectrum with multiple peaks.

The latter point becomes particularly critical in long-term numerical simulation. In a time domain computation of the responses of a vessel in a random seaway, the repeated evaluation of the apparently simple eq. (1.1) at hundreds of points on the hull for thousands of time steps becomes a major factor determining the execution speed of the code [6]. So, finding a less computationally intensive method for modelling ocean waves has the potential to increase performance of long-term simulation.

1.2 Related work

1.2.1 Ocean wave modelling

Another approach to simulating sea waves involves representing stochastic moving surface as a linear transformation of white noise with memory, which allows to model stationary ergodic Gaussian random process with given correlation characteristics [7]. The first attempts to model two-dimensional disturbances were undertaken in [8], which resulted in the development of the resonance theory of wind waves, and the formal mathematical framework was developed in [9, 10] — the authors built a one-dimensional model of ocean waves based on autoregressive-moving average (ARMA) model.

One-dimensional ARMA model does not have some of the LH model deficiencies: it is both computationally efficient and requires less number of coefficients to converge. In [11] ARMA model is used to generate time series spectrum of which is compatible with Pierson—Moskowitz (PM) approximation of ocean wave spectrum. The authors carry out experiments for one-dimensional AR, MA and ARMA models. They mention excellent agreement between target and initial spectra and higher performance of ARMA model compared to models based on summing large number of harmonic components with random phases. They also mention that in order to reach agreement between target and initial spectrum MA model require lesser number of coefficients than AR model. In [12] the authors generalise ARMA model coefficients determination formulae for multi-variate case.

AR model was successfully applied to predict evolution of propagating wave profiles based on instantaneous wave recordings. In [13] AR model is used to predict swell waves to control wave-energy converters (WEC) in real-time. In order to make WEC more efficient its internal oscillator frequency should match the one of ocean waves. The authors treat wave elevation as time series and compare performance of AR model, neural networks and cyclical models in forecasting time series future values. AR model gives the most accurate prediction of low-frequency swell waves for up to two typical wave periods. It is an example of successful application of AR process to ocean wave modelling.

The feature that distinguishes present work with respect to afore-mentioned ones is the study of three-dimensional (2D in space and 1D in time) ARMA model, which is mostly a different problem.

1. Yule—Walker system of equations, which are used to determine AR coefficients, has complex block-block structure.
2. Optimal model order (in a sense that target spectrum agrees with initial) is determined manually.
3. Instead of PM spectrum, analytic formulae for standing and propagating waves ACF are used as the model input.
4. Three-dimensional wavy surface should be compatible with real ocean surface not only in terms of spectral characteristics, but also in the shape of wave profiles. So, model verification includes distributions of various parameters of generated waves (lengths, heights, periods etc.).

Multi-dimensionality of investigated model not only complexifies the task, but also allows to carry out visual validation of generated wavy surface. It is the opportunity to visualise output of the programme that allowed to ensure that generated surface is compatible with real ocean surface, and is not abstract multi-dimensional stochastic process that is real only statistically.

1.2.2 Pressure field determination formulae

Small amplitude waves theory

In [14–16] the authors propose a solution for inverse problem of hydrodynamics of potential flow within the framework of small-amplitude wave theory (under assumption that wave length is much larger than height: $\lambda \gg h$). In that case inverse problem is linear and reduces to Laplace equation with mixed boundary conditions, and equation of motion is solely used to determine pressures for calculated velocity potential derivatives. The assumption of small amplitudes means the slow decay of wind wave coherence function, i.e. small change of local wave number in time and space compared to the wavy surface elevation (z coordinate). This assumption allows to calculate elevation z derivative as $\zeta_z = k\zeta$, where k is wave number. In two-dimensional case the solution is written explicitly as

$$\left.\frac{\partial \phi}{\partial x}\right|_{x,t} = -\frac{1}{\sqrt{1+\alpha^2}} e^{-I(x)} \int\limits_0^x \frac{\partial \dot{\zeta}/\partial z + \alpha \dot{\alpha}}{\sqrt{1+\alpha^2}} e^{I(x)} dx, \qquad (1.3)$$

$$I(x) = \int\limits_0^x \frac{\partial \alpha / \partial z}{1+\alpha^2} dx,$$

where α is wave slope. In three-dimensional case solution is written in the form of elliptic partial differential equation (PDE):

$$\begin{gathered}\frac{\partial^2 \phi}{\partial x^2}\left(1+\alpha_x^2\right) + \frac{\partial^2 \phi}{\partial y^2}\left(1+\alpha_y^2\right) + 2\alpha_x\alpha_y \frac{\partial^2 \phi}{\partial x \partial y} + \\ \left(\frac{\partial \alpha_x}{\partial z} + \alpha_x \frac{\partial \alpha_x}{\partial x} + \alpha_y \frac{\partial \alpha_x}{\partial y}\right)\frac{\partial \phi}{\partial x} + \\ \left(\frac{\partial \alpha_y}{\partial z} + \alpha_x \frac{\partial \alpha_y}{\partial x} + \alpha_y \frac{\partial \alpha_y}{\partial y}\right)\frac{\partial \phi}{\partial y} + \\ \frac{\partial \dot{\zeta}}{\partial z} + \alpha_x \dot{\alpha}_x + \alpha_y \dot{\alpha}_y = 0.\end{gathered}$$

The authors suggest transforming this equation to finite differences and solve it numerically.

As will be shown in sec. 1.4.3 that (1.3) diverges when attempted to calculate velocity field for large amplitude waves, and this is the reason that it can not be used together with ARMA model, that generates arbitrary amplitude waves.

Linearisation of boundary condition

LH model allows to derive an explicit formula for velocity field by linearising kinematic boundary condition. Velocity potential formula is written as

$$\phi(x,y,z,t) = \sum_n \frac{c_n g}{\omega_n} e^{\sqrt{u_n^2+v_n^2} z} \sin(u_n x + v_n y - \omega_n t + \varepsilon_n).$$

This formula is differentiated to obtain velocity potential derivatives, which are plugged to dynamic boundary condition to obtain pressures.

1.3 Three-dimensional ARMA process as a sea wave simulation model

ARMA ocean simulation model defines wavy surface as three-dimensional (two dimensions in space and one in time) autoregressive moving average process: every surface point is represented as a weighted sum of previous in time and space points plus weighted sum of previous in time and space normally distributed random impulses. The governing equation for 3-D ARMA process is

$$\zeta_{\mathbf{i}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{N}} \Phi_{\mathbf{j}} \zeta_{\mathbf{i}-\mathbf{j}} + \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{M}} \Theta_{\mathbf{j}} \varepsilon_{\mathbf{i}-\mathbf{j}}, \tag{1.4}$$

where ζ — wave elevation, Φ — AR process coefficients, Θ — MA process coefficients, ε — white noise with Gaussian distribution, $\mathbf{N}$ — AR process order, $\mathbf{M}$ — MA process order, and $\Phi_{\mathbf{0}} \equiv 0$, $\Theta_{\mathbf{0}} \equiv 0$. Here arrows denote multi-component indices with a component for each dimension. In general, any scalar quantity can be a component (temperature, salinity, concentration of some substance in water etc.). Equation parameters are AR and MA process coefficients and order.

Any ARMA process can be uniquely represented as either MA or AR process of infinite order [10], and the parameters of the spectral representation are defined by the rule of division of power series (in a rational factorized form [9]):

$$S(\omega) = \frac{\Delta\sigma^2}{\pi} \frac{\prod\limits_m (1 - z_m e^{-im\omega\Delta})(1 - z_m e^{im\omega\Delta})}{\prod\limits_n (1 - p_n e^{-in\omega\Delta})(1 - p_n e^{in\omega\Delta})},$$

where z_m and p_n are the zeros of numerator (MA), and denominator (AR), respectively, which form a pair of mutually conjugate numbers. If some of the zeros are located near the unit circle, then the spectral density will have pronounced dips.

1.3.1 Autoregressive (AR) process

AR process is ARMA process with only one random impulse instead of their weighted sum:

$$\zeta_{\mathbf{i}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{N}} \Phi_{\mathbf{j}} \zeta_{\mathbf{i}-\mathbf{j}} + \varepsilon_{i,j,k}. \tag{1.5}$$

The coefficients Φ are calculated from auto-covariate function (ACF) via three-dimensional Yule—Walker (YW) equations, which are obtained after multiplying both parts of the previous equation by $\zeta_{\mathbf{i}-\mathbf{k}}$ and computing the expected value. Generic form of YW equations is

$$\gamma_{\mathbf{k}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{N}} \Phi_{\mathbf{j}} \ \gamma_{\mathbf{k}-\mathbf{j}} + \sigma_\varepsilon^2 \delta_{\mathbf{k}}, \qquad \delta_{\mathbf{k}} = \begin{cases} 1, & \text{if } \mathbf{k} = 0 \\ 0, & \text{if } \mathbf{k} \neq 0, \end{cases} \tag{1.6}$$

where γ — ACF of process ζ, σ_ε^2 — white noise variance. Matrix form of three-dimensional YW equations, which is used in the present work, is

$$\Gamma \begin{bmatrix} \Phi_{\mathbf{0}} \\ \Phi_{0,0,1} \\ \vdots \\ \Phi_{\mathbf{N}} \end{bmatrix} = \begin{bmatrix} \gamma_{0,0,0} - \sigma_\varepsilon^2 \\ \gamma_{0,0,1} \\ \vdots \\ \gamma_{\mathbf{N}} \end{bmatrix}, \qquad \Gamma = \begin{bmatrix} \Gamma_0 & \Gamma_1 & \cdots & \Gamma_{N_1} \\ \Gamma_1 & \Gamma_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \Gamma_1 \\ \Gamma_{N_1} & \cdots & \Gamma_1 & \Gamma_0 \end{bmatrix},$$

where $\mathbf{N} = (p_1, p_2, p_3)$ and

$$\Gamma_i = \begin{bmatrix} \Gamma_i^0 & \Gamma_i^1 & \cdots & \Gamma_i^{N_2} \\ \Gamma_i^1 & \Gamma_i^0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \Gamma_i^1 \\ \Gamma_i^{N_2} & \cdots & \Gamma_i^1 & \Gamma_i^0 \end{bmatrix} \qquad \Gamma_i^j = \begin{bmatrix} \gamma_{i,j,0} & \gamma_{i,j,1} & \cdots & \gamma_{i,j,N_3} \\ \gamma_{i,j,1} & \gamma_{i,j,0} & \ddots & x & \vdots \\ \vdots & \ddots & \ddots & \gamma_{i,j,1} \\ \gamma_{i,j,N_3} & \cdots & \gamma_{i,j,1} & \gamma_{i,j,0} \end{bmatrix},$$

Since $\Phi_{\mathbf{0}} \equiv 0$, the first row and column of Γ can be eliminated. Matrix Γ is block-toeplitz, positive definite and symmetric, hence the system is efficiently solved by Cholesky decomposition, which is particularly suitable for these types of matrices.

After solving this system of equations white noise variance is estimated from (1.6) by plugging $\mathbf{k} = \mathbf{0}$:

$$\sigma_\varepsilon^2 = \sigma_\zeta^2 - \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{N}} \Phi_{\mathbf{j}} \; \gamma_{\mathbf{j}}.$$

1.3.2 Moving average (MA) process

MA process is ARMA process with $\Phi \equiv 0$:

$$\zeta_{\mathbf{i}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{M}} \Theta_{\mathbf{j}} \varepsilon_{\mathbf{i}-\mathbf{j}}. \tag{1.7}$$

MA coefficients Θ are defined implicitly via the following non-linear system of equations:

$$\gamma_{\mathbf{i}} = \left[\sum_{\mathbf{j}=\mathbf{i}}^{\mathbf{M}} \Theta_{\mathbf{j}} \Theta_{\mathbf{j}-\mathbf{i}} \right] \sigma_\varepsilon^2.$$

The system is solved numerically by fixed-point iteration method via the following formulae

$$\Theta_{\mathbf{i}} = -\frac{\gamma_{\mathbf{0}}}{\sigma_\varepsilon^2} + \sum_{\mathbf{j}=\mathbf{i}}^{\mathbf{M}} \Theta_{\mathbf{j}} \Theta_{\mathbf{j}-\mathbf{i}}.$$

Here coefficients $\boldsymbol{\Theta}$ are calculated from back to front: from $\mathbf{i} = \mathbf{M}$ to $\mathbf{i} = \mathbf{0}$. White noise variance is estimated by

$$\sigma_\varepsilon^2 = \frac{\gamma_\mathbf{0}}{1 + \sum\limits_{\mathbf{j}=\mathbf{0}}^{\mathbf{M}} \Theta_\mathbf{j}^2}.$$

Authors of [7] suggest using Newton—Raphson method to solve this equation with higher precision, however, this method does not work in three dimensions. Using slower method does not have dramatic effect on the overall programme performance, because the number of coefficients is small and most of the time is spent generating wavy surface.

1.3.3 Mixed autoregressive moving average (ARMA) process

Generally speaking, ARMA process is obtained by plugging MA generated wavy surface as random impulse to AR process, however, in order to get the process with desired ACF one should re-compute AR coefficients before plugging. There are several approaches to "mix" AR and MA processes.

- The approach proposed in [7] which involves dividing ACF into MA and AR part along each dimension is not applicable here, because in three dimensions such division is not possible: there always be parts of the ACF that are not taken into account by AR and MA process.
- The alternative approach is to use the same (undivided) ACF for both AR and MA processes but use different process order, however, then realisation characteristics (mean, variance etc.) become skewed: these are characteristics of the two overlapped processes.

For the first approach there is a formula to re-compute ACF for AR process, but there is such formula for the second approach. So, the best solution for now is to simply use AR and MA process exclusively for different types of waves.

1.3.4 Process selection criteria for different wave profiles

One problem of ARMA model application to ocean wave generation is that for different types of wave profiles different processes must be used: standing waves are modelled by AR process, and propagating waves by MA process. This statement comes from practice: if one tries to use the processes the other way round, the resulting realisation either diverges or does not correspond to real ocean waves. So, the best way to apply ARMA model to ocean wave generation is to use AR process for standing waves and MA process for progressive waves.

The other problem is inability to automatically determine optimal number of coefficients for three-dimensional AR and MA processes. For one-dimensional processes this can be achieved via iterative methods [7], but they diverge in three-dimensional case.

The final problem, which is discussed in 1.3.3, is inability to "mix" AR and MA process in three dimensions.

In practice some statements made for AR and MA processes in [7] should be flipped for three-dimensional case. For example, the authors say that ACF of MA process cuts at q and ACF of AR process decays to nought infinitely, but in practice making ACF of 3-dimensional MA process not decay results in it being non-invertible and producing realisation that does not look like real ocean waves, whereas doing the same for ACF of AR process results in stationary process and adequate realisation. Also, the authors say that one should allocate the first q points of ACF to MA process (as it often needed to describe the peaks in ACF) and leave the rest points to AR process, but in practice in case of ACF of a propagating wave AR process is stationary only for the first time slice of the ACF, and the rest is left to MA process.

To summarise, the only established scenario of applying ARMA model to ocean wave generation is to use AR process for standing waves and MA process for propagating waves. With a new formulae for 3 dimensions a single mixed ARMA process might increase model precision, which is one of the objectives of the future research.

1.3.5 The shape of ACF for different types of waves

Analytic method of finding the ACF

The straightforward way to find ACF for a given ocean wave profile is to apply Wiener—Khinchin theorem. According to this theorem the autocorrelation K of a function ζ is given by the Fourier transform of the absolute square of the function:

$$K(t) = \mathscr{F}\left\{|\zeta(t)|^2\right\}. \tag{1.8}$$

When ζ is replaced with actual wave profile, this formula gives you analytic formula for the corresponding ACF.

For three-dimensional wave profile (2D in space and 1D in time) analytic formula is a polynomial of high order and is best obtained via symbolic computation programme. Then for practical usage it can be approximated by superposition of exponentially decaying cosines (which is how ACF of a stationary ARMA process looks like [7]).

Empirical method of finding the ACF

However, for three-dimensional case there exists simpler empirical method which does not require sophisticated software to determine shape of the ACF. It is known that ACF represented by exponentially decaying cosines satisfies first order Stokes' equations for gravity waves [17]. So, if the shape of the wave profile is the only concern in the simulation, then one can simply multiply it by a decaying exponent to get appropriate ACF. This ACF does not reflect other wave profile parameters, such as wave height and period, but opens possibility to simulate waves of a particular non-analytic shape by "drawing" their profile, then multiplying it by an exponent and using the resulting function as ACF. So, this empirical method is imprecise but offers simpler alternative to Wiener—Khinchin theorem approach; it is mainly useful to test ARMA model.

Standing wave ACF

For three-dimensional plain standing wave the profile is given by

$$\zeta(t,x,y) = A\sin(k_x x + k_y y)\sin(\sigma t). \tag{1.9}$$

Find ACF via analytic method. Multiplying the formula by a decaying exponent (because Fourier transform is defined for a function f that $f \xrightarrow[x\to\pm\infty]{} 0$) yields

$$\zeta(t,x,y) = A\exp\left[-\alpha(|t|+|x|+|y|)\right]\sin(k_x x + k_y y)\sin(\sigma t). \tag{1.10}$$

Then, apply 3D Fourier transform to the both sides of the equation via symbolic computation programme, fit the resulting polynomial to the following approximation:

$$K(t,x,y) = \gamma\exp\left[-\alpha(|t|+|x|+|y|)\right]\cos\beta t\cos\left[\beta x + \beta y\right]. \tag{1.11}$$

So, after applying Wiener—Khinchin theorem we get initial formula but with cosines instead of sines. This difference is important because the value of ACF at $(0,0,0)$ equals to the ARMA process variance, and if one used sines the value would be wrong.

If one tries to replicate the same formula via empirical method, the usual way is to adapt (1.10) to match (1.11). This can be done either by changing the phase of the sine, or by substituting sine with cosine to move the maximum of the function to the origin of coordinates.

Propagating wave ACF

Three-dimensional profile of plain propagating wave is given by

$$\zeta(t,x,y) = A\cos(\sigma t + k_x x + k_y y). \tag{1.12}$$

For the analytic method repeating steps from the previous two paragraphs yields

$$K(t,x,y) = \gamma \exp\left[-\alpha(|t|+|x|+|y|)\right] \cos\left[\beta(t+x+y)\right]. \tag{1.13}$$

For the empirical method the wave profile is simply multiplied by a decaying exponent without need to adapt the maximum value of ACF (as it is required for standing wave).

Comparison of studied methods

To summarise, the analytic method of finding ocean wave's ACF reduces to the following steps.

- Make wave profile decay when approaching $\pm\infty$ by multiplying it by a decaying exponent.
- Apply Fourier transform to the absolute square of the resulting equation using symbolic computation programme.
- Fit the resulting polynomial to the appropriate ACF approximation.

Two examples in this section showed that in case of standing and propagating waves their decaying profiles resemble the corresponding ACFs with the exception that the ACF's maximum should be moved to the origin to preserve simulated process variance. Empirical method of finding ACF reduces to the following steps.

- Make wave profile decay when approaching $\pm\infty$ by multiplying it by a decaying exponent.
- Move maximum value of the resulting function to the origin by using trigonometric identities to shift the phase.

1.3.6 Evaluation and discussion

In [18–20] for AR model the following items were verified experimentally:

- probability distributions of different wave characteristics (wave heights, lengths, crests, periods, slopes, three-dimensionality),
- dispersion relation,
- retention of integral characteristics for mixed wave sea state.

In this work we repeat probability distribution tests for three-dimensional AR and MA model.

In [9] the authors show that several ocean wave characteristics (listed in table 1.1) have Weibull distribution, and wavy surface elevation has Gaussian distribution. In order to verify that distributions corresponding to generated realisation are correct, quantile-quantile plots are used (plots where analytic quantile values are used for *OX* axis and estimated quantile values for *OY* axis). If the estimated distribution matches analytic then the graph has the form of the straight line. Tails of the graph may diverge from the straight line, because they can not be reliably estimated from the finite-size realisation. Different methods of extracting waves from realisation produce variations in quantile function tails, it is probably impractical to extract every possible wave from realisation since they may (and often) overlap.

Characteristic	Weibull shape (k)
Wave height	2
Wave length	2.3
Crest length	2.3
Wave period	3
Wave slope	2.5
Three-dimensionality	2.5

Table 1.1 Values of Weibull shape parameter for different wave characteristics.

Verification was performed for standing and propagating waves. The corresponding ACFs and quantile-quantile plots of wave characteristics distributions are shown in fig. , , .

Graph tails in fig. deviate from original distribution for individual wave characteristics, because every wave have to be extracted from the resulting wavy surface to measure its length, period and height. There is no algorithm that guarantees correct extraction of all waves, because they may overlap each other. Weibull distribution right tail represents infrequently occurring waves, so it deviates more than left tail.

Degree of correspondence for standing waves (fig.) is lower for height and length, is roughly the same for surface elevation and is higher for wave period distribution tails. Lower correspondence degree for length and height may be attributed to the fact that Weibull distributions were obtained empirically for ocean waves which are typically propagating, and distributions may be different for standings waves. Higher correspondence degree for wave periods is attributed to the fact that wave periods of standing waves are extracted more precisely as the waves do not move outside simulated wavy surface region. The same correspondence degree for wave elevation is obtained, because this is the characteristic of the wavy surface (and corresponding AR or MA process) and is not affected by the type of waves.

ARMA model, owing to its non-physical nature, does not have the notion of ocean wave; it simulates wavy surface as a whole instead. Motions of individual waves and their shape are often rough, and the total number of waves can not be determined precisely. However, integral characteristics of wavy surface match the ones of real ocean waves.

Theoretically, ocean waves themselves can be chosen as ACFs, the only pre-processing step is to make them decay exponentially. This may allow to generate waves of arbitrary profiles, and is one of the directions of future work.

1.4 Determining wave pressures for discretely given wavy surface

Analytic solutions to boundary problems in classical equations are often used to study different properties of the solution, and for that purpose general solution formula is too difficult to study, as it contains integrals of unknown functions. Fourier method is one of the methods to find analytic solutions to a PDE. It is based on application of Fourier transform to each part of PDE, which reduces the equation to algebraic, and the solution is written as inverse Fourier transform of some function (which may contain Fourier transforms of other functions). Since, it is not possible to write analytic forms of these Fourier transforms in all cases, unique solutions are found and their behaviour is studied in different domains instead. At the same time, computing discrete Fourier transforms on the computer is possible for any discretely defined function and efficient when using FFT algorithms. These algorithms use symmetry of complex exponentials to decrease asymptotic complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n\log_2 n)$. So, even if general solution contains Fourier transforms of unknown functions, they still can be computed numerically, and FFT family of algorithms makes this approach efficient.

Alternative approach to solve a PDE is to reduce it to difference equations, which are solved by constructing various numerical schemes. This approach leads to approximate solution, and asymptotic complexity of corresponding algorithms is comparable to that of FFT. For example, stationary elliptic PDE transforms to implicit numerical scheme which is solved by iterative method on each step of which a tridiagonal or five-diagonal system of algebraic equations is solved via Thomas algorithm. Asymptotic complexity of this approach is $\mathcal{O}(nm)$, where n — number of wavy surface grid points, m — number of iterations. Despite their wide spread, iterative algorithms are inefficient on parallel computer architectures; in particular, their mapping to co-processors may involve copying data in and out of the co-processor in each iteration, which negatively affects their performance. At the same time, high number of Fourier transforms in the solution is an advantage, rather than a disadvantage. First, solutions obtained by Fourier method are explicit, hence their implementations scales with the large number of parallel computer cores. Second, there are implementations of FFT optimised for different processor architectures as well as co-processors (GPU, MIC) which makes it easy to get high performance on any computing platform. These advantages substantiate the choice of Fourier method to obtain explicit analytic solution to the problem of determining pressures under wavy ocean surface.

The problem of finding pressure field under wavy sea surface represents inverse problem of hydrodynamics for incompressible inviscid fluid. System of equations for it in general case is written as [5]

$$\begin{aligned}
&\nabla^2\phi = 0, && \\
&\phi_t + \frac{1}{2}|\upsilon|^2 + g\zeta = -\frac{p}{\rho}, && \text{at } z=\zeta(x,y,t), \\
&D\zeta = \nabla\phi\cdot\mathbf{n}, && \text{at } z=\zeta(x,y,t),
\end{aligned} \tag{1.14}$$

where ϕ — velocity potential, ζ — elevation (z coordinate) of wavy surface, p — wave pressure, ρ — fluid density, $\upsilon = (\phi_x, \phi_y, \phi_z)$ — velocity vector, g — acceleration of gravity, and D — substantial (Lagrange) derivative. The first equation is called continuity (Laplace) equation, the second one is the conservation of momentum law (the so called dynamic boundary condition); the third one is kinematic boundary condition for free wavy surface, which states that rate of change of wavy surface elevation ($D\zeta$) equals to the change of velocity potential derivative along the wavy surface normal ($\nabla\phi\cdot\mathbf{n}$).

Inverse problem of hydrodynamics consists in solving this system of equations for ϕ. In this formulation dynamic boundary condition becomes an explicit formula to determine pressure field using velocity potential derivatives obtained from the remaining equations. So, from mathematical point of view inverse problem of hydrodynamics reduces to Laplace equation with mixed boundary condition — Robin problem.

1.4.1 Two-dimensional case

Formula for infinite depth fluid

Two-dimensional Laplace equation with Robin boundary condition is written as

$$\begin{aligned}
&\phi_{xx} + \phi_{zz} = 0, && \\
&\zeta_t + \zeta_x\phi_x = \frac{\zeta_x}{\sqrt{1+\zeta_x^2}}\phi_x - \phi_z, && \text{at } z=\zeta(x,t).
\end{aligned} \tag{1.15}$$

Use Fourier method to solve this problem. Applying Fourier transform to both sides of the equation yields

$$-4\pi^2\left(u^2+v^2\right)\mathscr{F}_{u,v}\{\phi(x,z)\}=0,$$

hence $v=\pm iu$. Hereinafter we use the following symmetric form of Fourier transform:

$$\mathscr{F}_{u,v}\{f(x,y)\}=\iint\limits_{-\infty}^{\infty}f(x,y)e^{-2\pi i(xu+yv)}dxdy.$$

We seek solution in the form of inverse Fourier transform $\phi(x,z)=\mathscr{F}^{-1}_{x,z}\{E(u,v)\}$. Plugging[1] $v=iu$ into the formula yields

$$\phi(x,z)=\mathscr{F}^{-1}_{x}\left\{e^{2\pi uz}E(u)\right\}. \tag{1.16}$$

In order to make substitution $z=\zeta(x,t)$ not interfere with Fourier transforms, we rewrite (1.16) as a convolution:

$$\phi(x,z)=\mathscr{D}_1(x,z)\ast\mathscr{F}^{-1}_{x}\{E(u)\},$$

where $\mathscr{D}_1(x,z)$ — a function, form of which is defined in sec. 1.5.3 and which satisfies equation $\mathscr{F}_u\{\mathscr{D}_1(x,z)\}=e^{2\pi uz}$. Plugging formula ϕ into the boundary condition yields

$$\zeta_t=(if(x)-1)\left[\mathscr{D}_1(x,z)\ast\mathscr{F}^{-1}_{x}\{2\pi uE(u)\}\right],$$

where $f(x)=\zeta_x/\sqrt{1+\zeta_x^2}-\zeta_x$. Applying Fourier transform to both sides of this equation yields formula for coefficients E:

$$E(u)=\frac{1}{2\pi u}\frac{\mathscr{F}_u\{\zeta_t/(if(x)-1)\}}{\mathscr{F}_u\{\mathscr{D}_1(x,z)\}}$$

Finally, substituting z for $\zeta(x,t)$ and plugging resulting equation into (1.16) yields formula for $\phi(x,z)$:

$$\boxed{\phi(x,z)=\mathscr{F}^{-1}_{x}\left\{\frac{e^{2\pi uz}}{2\pi u}\frac{\mathscr{F}_u\{\zeta_t/(if(x)-1)\}}{\mathscr{F}_u\{\mathscr{D}_1(x,\zeta(x,t))\}}\right\}.} \tag{1.17}$$

Multiplier $e^{2\pi uz}/(2\pi u)$ makes a graph of a function to which Fourier transform is applied asymmetric with respect to OY axis. This makes it difficult to apply FFT which expects periodic function with nought on both ends of the interval. Using numerical integration instead of FFT is not faster than solving the initial system of equations with numerical schemes. This problem is alleviated by using formula (1.19) for finite depth fluid with wittingly large depth h. This formula is derived in the following section.

[1] $v=-iu$ is not applicable because velocity potential must go to nought when depth goes to infinity.

Formula for finite depth fluid

On the sea bottom vertical fluid velocity component equals nought: $\phi_z = 0$ on $z = -h$, where h — water depth. In this case equation $v = -iu$, which came from Laplace equation, can not be neglected, hence the solution is sought in the following form:

$$\phi(x,z) = \mathscr{F}_x^{-1}\left\{ \left(C_1 e^{2\pi u z} + C_2 e^{-2\pi u z} \right) E(u) \right\}. \tag{1.18}$$

Plugging ϕ into the boundary condition on the sea bottom yields

$$C_1 e^{-2\pi u h} - C_2 e^{2\pi u h} = 0,$$

hence $C_1 = \frac{1}{2} C e^{2\pi u h}$ and $C_2 = -\frac{1}{2} C e^{-2\pi u h}$. Constant C may take arbitrary value here, because after plugging it becomes part of unknown coefficients $E(u)$. Plugging formulae for C_1 and C_2 into (1.18) yields

$$\phi(x,z) = \mathscr{F}_x^{-1}\left\{ \cosh\left(2\pi u (z+h)\right) E(u) \right\}.$$

Plugging ϕ into the boundary condition on the free surface yields

$$\zeta_t = f(x) \mathscr{F}_x^{-1}\left\{ 2\pi i u \cosh\left(2\pi u (z+h)\right) E(u) \right\} - \mathscr{F}_x^{-1}\left\{ 2\pi u \sinh\left(2\pi u (z+h)\right) E(u) \right\}.$$

Here sinh and cosh give similar results near free surface, and since this is the main area of interest in practical applications, we assume that $\cosh\left(2\pi u (z+h)\right) \approx \sinh\left(2\pi u (z+h)\right)$. Performing analogous to the previous section transformations yields final formula for $\phi(x,z)$:

$$\boxed{\phi(x,z,t) = \mathscr{F}_x^{-1}\left\{ \frac{\cosh\left(2\pi u (z+h)\right)}{2\pi u} \frac{\mathscr{F}_u\{\zeta_t / (i f(x) - 1)\}}{\mathscr{F}_u\{\mathscr{D}_2(x, \zeta(x,t))\}} \right\},} \tag{1.19}$$

where $\mathscr{D}_2(x,z)$ — a function, form of which is defined in sec. 1.5.3 and which satisfies equation $\mathscr{F}_u\{\mathscr{D}_2(x,z)\} = \cosh(2\pi u z)$.

Reducing to the formulae from linear wave theory

Check the validity of derived formulae by substituting $\zeta(x,t)$ with known analytic formula for plain waves. Symbolic computation of Fourier transforms in this section were performed in Mathematica [21]. In the framework of linear wave theory assume that waves have small amplitude compared to their lengths, which allows us to simplify initial system of equations (1.15) to

$$\begin{aligned} &\phi_{xx} + \phi_{zz} = 0, \\ &\zeta_t = -\phi_z && \text{at } z = \zeta(x,t), \end{aligned}$$

solution to which is written as

$$\phi(x,z,t) = -\mathscr{F}_x^{-1}\left\{ \frac{e^{2\pi u z}}{2\pi u} \mathscr{F}_u\{\zeta_t\} \right\}.$$

Propagating wave profile is defined as $\zeta(x,t) = A\cos(2\pi(kx-t))$. Plugging this formula into (1.17) yields $\phi(x,z,t) = -\frac{A}{k}\sin(2\pi(kx-t))\cosh(2\pi kz)$. In order to reduce it to the formula from linear wave theory, rewrite hyperbolic sine in exponential form, discard the term containing $e^{-2\pi kz}$ as contradicting condition $\phi \underset{z\rightarrow-\infty}{\longrightarrow} 0$. Taking real part of the resulting formula yields $\phi(x,z,t) = \frac{A}{k}e^{2\pi kz}\sin(2\pi(kx-t))$, which corresponds to the known formula from linear wave theory. Similarly, under small-amplitude waves assumption the formula for finite depth fluid (1.19) is reduced to

$$\phi(x,z,t) = -\mathscr{F}_x^{-1}\left\{ \frac{\cosh(2\pi u(z+h))}{2\pi u \cosh(2\pi u h)} \mathscr{F}_u\{\zeta_t\} \right\}.$$

Substituting $\zeta(x,t)$ with propagating plain wave profile formula yields

$$\phi(x,z,t) = \frac{A}{k}\frac{\cosh(2\pi k(z+h))}{\cosh(2\pi kh)}\sin(2\pi(kx-t)), \qquad (1.20)$$

which corresponds to the formula from linear wave theory for finite depth fluid.

Different forms of Laplace equation solutions, in which decaying exponent is written with either "+" or "-" signs, may cause incompatibilities between formulae from linear wave theory and formulae derived in this work, where $\sinh$ is used instead of $\cosh$. Equality $\frac{\cosh(2\pi k(z+h))}{\cosh(2\pi kh)} \approx \frac{\sinh(2\pi k(z+h))}{\sinh(2\pi kh)}$ becomes strict on the free surface, and difference between left-hand and right-hand sides increases when approaching sea bottom (for sufficiently large depth difference near free surface is negligible). So, for sufficiently large depth any function ($\cosh$ or $\sinh$) may be used for velocity potential computation near free surface.

Reducing (1.17) and (1.19) to the known formulae from linear wave theory shows, that formula for infinite depth (1.17) is not suitable to compute velocity potentials with Fourier method, because it does not have symmetry, which is required for Fourier transform. However, formula for finite depth can be used instead by setting h to some characteristic water depth. For standing wave reducing to linear wave theory formulae is made under the same assumptions.

1.4.2 Three-dimensional case

Three-dimensional version of (1.14) is written as

$$\begin{aligned}
&\phi_{xx}+\phi_{yy}+\phi_{zz}=0, \\
&\zeta_t+\zeta_x\phi_x+\zeta_y\phi_y = \frac{\zeta_x}{\sqrt{1+\zeta_x^2+\zeta_y^2}}\phi_x + \frac{\zeta_y}{\sqrt{1+\zeta_x^2+\zeta_y^2}}\phi_y - \phi_z, \quad \text{at } z=\zeta(x,y,t).
\end{aligned} \tag{1.21}$$

Again, use Fourier method to solve it. Applying Fourier transform to both sides of Laplace equation yields

$$-4\pi^2\left(u^2+v^2+w^2\right)\mathscr{F}_{u,v,w}\{\phi(x,y,z)\}=0,$$

hence $w=\pm i\sqrt{u^2+v^2}$. We seek solution in the form of inverse Fourier transform $\phi(x,y,z)=\mathscr{F}^{-1}_{x,y,z}\{E(u,v,w)\}$. Plugging $w=i\sqrt{u^2+v^2}=i|\mathbf{k}|$ into the formula yields

$$\phi(x,y,z)=\mathscr{F}^{-1}_{x,y}\left\{\left(C_1e^{2\pi|\mathbf{k}|z}-C_2e^{-2\pi|\mathbf{k}|z}\right)E(u,v)\right\}.$$

Plugging ϕ into the boundary condition on the sea bottom (analogous to two-dimensional case) yields

$$\phi(x,y,z)=\mathscr{F}^{-1}_{x,y}\{\cosh(2\pi|\mathbf{k}|(z+h))E(u,v)\}. \tag{1.22}$$

Plugging ϕ into the boundary condition on the free surface yields

$$\begin{aligned}
\zeta_t = \; & if_1(x,y)\mathscr{F}^{-1}_{x,y}\{2\pi u\cosh(2\pi|\mathbf{k}|(z+h))E(u,v)\} \\
&+ if_2(x,y)\mathscr{F}^{-1}_{x,y}\{2\pi v\cosh(2\pi|\mathbf{k}|(z+h))E(u,v)\} \\
&- \mathscr{F}^{-1}_{x,y}\{2\pi|\mathbf{k}|\sinh(2\pi|\mathbf{k}|(z+h))E(u,v)\}
\end{aligned}$$

where $f_1(x,y)=\zeta_x/\sqrt{1+\zeta_x^2+\zeta_y^2}-\zeta_x$ and $f_2(x,y)=\zeta_y/\sqrt{1+\zeta_x^2+\zeta_y^2}-\zeta_y$.

Like in sec. 1.4.1 we assume that $\cosh(2\pi u(z+h))\approx\sinh(2\pi u(z+h))$ near free surface, but in three-dimensional case this is not enough to solve the problem. In order to get analytic formula for coefficients E we need to assume, that all Fourier transforms in the equation have radially symmetric kernels, i.e. replace u and v with $|\mathbf{k}|$. There are two points supporting this assumption. First, in numerical implementation integration is done over positive wave numbers, so the sign of u and v does not affect the solution. Second, the rate growth of cosh term of the integral kernel is much higher than the one of u or $|\mathbf{k}|$, so the substitution has small effect on the magnitude of the solution. Despite these two points, a use of more mathematically rigorous approach would be preferable.

Making the replacement, applying Fourier transform to both sides of the equation and plugging the result into (1.22) yields formula for ϕ:

$$\phi(x,y,z,t) = \mathscr{F}^{-1}_{x,y}\left\{ \frac{\cosh\left(2\pi|\mathbf{k}|(z+h)\right)}{2\pi|\mathbf{k}|} \frac{\mathscr{F}_{u,v}\{\zeta_t / \left(i f_1(x,y) + i f_2(x,y) - 1 \right)\}}{\mathscr{F}_{u,v}\{\mathscr{D}_3\left(x,y,\zeta\left(x,y\right)\right)\}} \right\},$$

where $\mathscr{F}_{u,v}\{\mathscr{D}_3\left(x,y,z\right)\} = \cosh\left(2\pi|\mathbf{k}|z\right)$.

1.4.3 Evaluation and discussion

Comparing obtained generic formulae (1.17) and (1.19) to the known formulae from linear wave theory allows to see the difference between velocity fields for both large and small amplitude waves. In general analytic formula for velocity potential in not known, even for plain waves, so comparison is done numerically. Taking into account conclusions of 1.4.1, only finite depth formulae are compared.

The difference with linear wave theory formulae

In order to obtain velocity potential fields, ocean wavy surface was generated by AR model with varying wave amplitude. In numerical implementation wave numbers in Fourier transforms were chosen on the interval from 0 to the maximal wave number determined numerically from the obtained wavy surface. Experiments were conducted for waves of both small and large amplitudes.

The experiment showed that velocity potential fields produced by formula (1.19) for finite depth fluid and formula (1.20) from linear wave theory are qualitatively different (fig.). First, velocity potential contours have sinusoidal shape, which is different from oval shape described by linear wave theory. Second, velocity potential decays more rapidly than in linear wave theory as getting closer to the bottom, and the region where the majority of wave energy is concentrated is closer to the wave crest. Similar numerical experiment, in which all terms of (1.19) that are neglected in the framework of linear wave theory are eliminated, shows no difference (as much as machine precision allows) in resulting velocity potential fields.

The difference with small-amplitude wave theory

The experiment, in which velocity fields produced numerically by different formulae were compared, shows that velocity fields produced by formula (1.19) and (1.3) correspond to each other for small-amplitude waves. Two ocean wavy surface realisations were made by AR model: one containing small-amplitude waves, other containing large-amplitude waves. Integration in formula (1.19) was done over wave numbers range extracted from the generated wavy surface. For small-amplitude waves both formulae showed comparable results (the difference in the velocity is attributed to the stochastic nature of AR model), whereas for large-amplitude waves stable velocity field was produced only by formula (1.19) (fig.). So, generic formula (1.19) gives satisfactory results without restriction on wave amplitudes.

1.5 High-performance software implementation for heterogeneous platforms

1.5.1 White noise generation

In order to eliminate periodicity from generated wavy surface, it is imperative to use PRNG with sufficiently large period to generate white noise. Parallel Mersenne Twister [22] with a period of $2^{19937}-1$ is used as a generator in this work. It allows to produce aperiodic ocean wavy surface realisations in any practical usage scenarios.

There is no guarantee that multiple Mersenne Twisters executed in parallel threads with distinct initial states produce uncorrelated pseudo-random number sequences, however, algorithm of dynamic creation of Mersenne Twisters [23] may be used to provide such a guarantee. The essence of the algorithm is to find matrices of initial generator states, that give maximally uncorrelated pseudo-random number sequences when Mersenne Twisters are executed in parallel with these initial states. Since finding such initial states consumes considerable amount of processor time, vector of initial states is created preliminary with knowingly larger number of parallel threads and saved to a file, which is then read before starting white noise generation.

1.5.2 Wavy surface generation

In ARMA model value of wavy surface elevation at a particular point depends on previous in space and time points, as a result the so called ramp-up interval (see fig.), in which realisation does not correspond to specified ACF, forms at the beginning of the realisation. There are several solutions to this problem which depend on the simulation context.

If the realisation is used in the context of ship stability simulation without manoeuvring, ramp-up interval will not affect results of the simulation, because it is located on the border (too far away from the studied marine object). If ship stability with manoeuvring is studied, then the interval may be simply discarded from the realisation (the size of the interval approximately equals the number of AR coefficients in each dimension). However, this may lead to a loss of a very large number of points, because discarding is done for each dimension. Alternative approach is to generate ocean wavy surface on ramp-up interval with LH model and generate the rest of the realisation with ARMA model.

Algorithm of wavy surface generation is data-parallel: the realisation is divided into equal parts along the time axis each of which is generated independently, however, in the beginning of each realisation there is ramp-up interval. To eliminate it for MA process, overlap-add method [24–26] (a popular method in signal processing) is used. The essence of the method is to add another interval, size of which is equal to the ramp-up interval size, to the end of each part. Then wavy surface is generated in each point of each part (including points from the added interval), the interval at the end of part N is superimposed on the ramp-up interval at the beginning of the part $N+1$, and values in corresponding points are added. To eliminate the ramp-up interval for AR process, the realisation is divided into part along each dimension, and each part is computed only when all dependent parts are ready. For that purpose, an array of current part states is maintained in the programme, and all the parts are put into a queue. A parallel thread acquires a shared lock, finds the first part in the queue, for which all dependent parts have "completed" state, removes this part for the queue, frees the lock and generates the part. After that a thread updates the state of the part, and repeats the same steps until the queue becomes empty. This algorithm eliminates all ramp-up intervals except the one at the beginning of the realisation, and the size of the parts should be sufficiently small to balance the load on all processor cores.

1.5.3 Velocity potential computation

In solutions (1.17) and (1.19) to two-dimensional problem there are functions $\mathscr{D}_1(x,z) = \mathscr{F}_x^{-1}\left\{e^{2\pi u z}\right\}$ and $\mathscr{D}_2(x,z) = \mathscr{F}_x^{-1}\left\{\cosh(2\pi u z)\right\}$ which has multiple analytic representations and are difficult to compute. Each function is a Fourier transform of linear combination of exponents which reduces to poorly defined Dirac delta function of a complex argument (see table 1.2). The usual way of handling this type of functions is to write them as multiplication of Dirac delta functions of real and imaginary part, however, this approach does not work here, because applying inverse Fourier transform to this representation does not produce exponent, which severely warp resulting velocity field. In order to get unique analytic definition, normalisation factor $1/\cosh(2\pi u h)$ (which is also included in the formula for $E(u)$) may be used. Despite the fact that normalisation allows to obtain adequate velocity potential field, numerical experiments show that there is little difference between this field and the one produced by formulae, in which terms with ζ are omitted. As a result, we do not use normalisation factors in the formula.

Function	Without normalisation	Normalised
$\mathscr{D}_1(x,z)$	$\delta(x+iz)$	$\frac{1}{2h}\operatorname{sech}\left(\frac{\pi(x-i(h+z))}{2h}\right)$
$\mathscr{D}_2(x,z)$	$\frac{1}{2}\left[\delta(x-iz)+\delta(x+iz)\right]$	$\frac{1}{4h}\left[\operatorname{sech}\left(\frac{\pi(x-i(h+z))}{2h}\right)+\operatorname{sech}\left(\frac{\pi(x+i(h+z))}{2h}\right)\right]$

Table 1.2 Formulae for computing $\mathscr{D}_1(x,z)$ and $\mathscr{D}_2(x,z)$ from 1.4.1, that use normalisation to eliminate uncertainty from definition of Dirac delta function of complex argument.

1.5.4 Evaluation

ARMA model does not require highly optimised software implementation to be efficient, its performance is high even without use of co-processors; there are two main causes of that. First, ARMA model itself does not use transcendental functions (sines, cosines and exponents) as opposed to LH model. All calculations, except model coefficients, are done via polynomials, which can be efficiently computed on modern processors using a series of fused multiply-add (FMA) instructions. Second, pressure computation is done via explicit analytic formula using nested FFTs. Since two-dimensional FFT of the same size is repeatedly applied to every time slice, its coefficients (complex exponents) are pre-computed for all slices, and computations are performed with only a few transcendental functions. In case of MA model, performance is also increased by doing convolution with FFT. So, high performance of ARMA model is due to scarce use of transcendental functions and heavy use of FFT, not to mention that high convergence rate and non-existence of periodicity allows to use far fewer coefficients compared to LH model.

ARMA implementation uses several libraries of reusable mathematical functions and numerical algorithms (listed in table 1.3), and was implemented using OpenMP and OpenCL parallel programming technologies, that allow to use the most efficient implementation for a particular algorithm.

Library	What it is used for
DCMT [23]	parallel PRNG
Blitz [27, 28]	multidimensional arrays
GSL [29]	PDF, CDF, FFT computation
	checking process stationarity
LAPACK, GotoBLAS [30, 31]	finding AR coefficients
GL, GLUT [32]	three-dimensional visualisation
CGAL [33]	wave numbers triangulation

Table 1.3 A list of mathematical libraries used in ARMA model implementation.

For the purpose of evaluation we use simplified version of (1.4.2):

$$\begin{aligned}\phi(x,y,z,t) &= \mathscr{F}^{-1}_{x,y}\left\{ \frac{\cosh\left(2\pi|\mathbf{k}|(z+h)\right)}{2\pi|\mathbf{k}|\cosh\left(2\pi|\mathbf{k}|h\right)} \mathscr{F}_{u,v}\{\zeta_t\} \right\} \\ &= \mathscr{F}^{-1}_{x,y}\{g_1(u,v)\mathscr{F}_{u,v}\{g_2(x,y)\}\}\,. \end{aligned} \tag{1.23}$$

This formula is particularly suitable for computation on GPUs:

- it contains transcendental mathematical functions (hyperbolic cosines and complex exponents);
- it is computed over large four-dimensional (t, x, y, z) region;
- it is analytic with no information dependencies between individual data points in t and z dimensions.

Since standing sea wave generator does not allow efficient GPU implementation due to autoregressive dependencies between wavy surface points, only velocity potential solver was rewritten in OpenCL and its performance was compared to existing OpenMP implementation.

For each implementation the overall performance of the solver for a particular time instant was measured. Velocity field was computed for one t point, for 128 z points below wavy surface and for each x and y point of four-dimensional (t,x,y,z) grid. The only parameter that was varied between subsequent programme runs is the size of the grid along x dimension. A total of 10 runs were performed and an average time of each stage was computed.

A different FFT library was used for each version of the solver. For OpenMP version FFT routines from GNU Scientific Library (GSL) [29] were used, and for OpenCL version clFFT library [34] was used instead. There are two major differences in the routines from these libraries.

- The order of frequencies in Fourier transforms is different and clFFT library requires reordering the result of (1.23) whereas GSL does not.
- Discontinuity at $(x,y)=(0,0)$ of velocity potential field grid is handled automatically by clFFT library, whereas GSL library produce skewed values at this point.

For GSL library an additional interpolation from neighbouring points was used to smooth velocity potential field at these points. We have not spotted other differences in FFT implementations that have impact on the overall performance.

In the course of the numerical experiments we have measured how much time each solver's implementation spends in each computation stage to explain how efficient data copying between host and device is in OpenCL implementation, and how one implementation corresponds to the other in terms of performance.

1.5.5 Results

The experiments showed that GPU implementation outperforms CPU implementation by a factor of 10–15 (fig. 1.1), however, distribution of time between computation stages is different for each implementation (fig. 1.2). The major time consumer in CPU implementation is computation of g_1, whereas in GPU implementation its running time is comparable to computation of g_2. GPU computes g_1 much faster than CPU due to a large amount of modules for transcendental mathematical function computation. In both implementations g_2 is computed on CPU, but for GPU implementation the result is duplicated for each z grid point in order to perform multiplication of all XYZ planes along z dimension in single OpenCL kernel, and, subsequently copied to GPU memory which severely hinders overall stage performance. Copying the resulting velocity potential field between CPU and GPU consumes $\approx 20\%$ of velocity potential solver execution time.

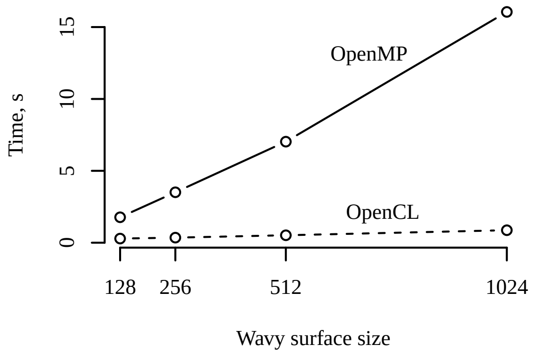

Figure 1.1 Performance comparison of CPU (OpenMP) and GPU (OpenCL) versions of velocity potential solver.

1.6 Conclusion

Three-dimensional ARMA ocean simulation model coupled with analytic formula for determining pressures under wavy sea surface is computationally efficient of performing long-term ship behaviour simulations on the computer. Possible applications of the approach include studying ship behaviour

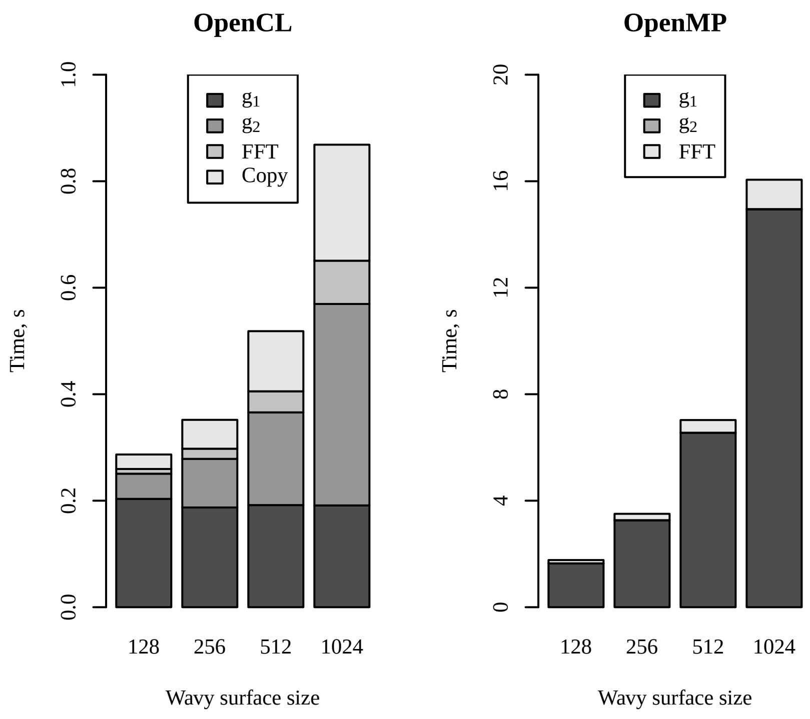

Figure 1.2 Performance breakdown for GPU (OpenCL) and CPU (OpenMP) versions of velocity potential solver.

in storm and shallow water waves. Its validity was visually and statistically verified in a number of experiments: distribution of characteristics of waves, produced by ARMA model, match the ones of real ocean waves, and velocity potential field, produced by the analytic formula correspond to the one produced by the formula for small-amplitude waves, and the formula itself reduces to the known one from linear wave theory.

Numerical experiments showed that wavy surface generation is efficient on CPU as it involves no transcendental mathematical functions, and velocity potential field computation is efficient on GPU due to heavy use of Fourier transforms. The use of dynamically generated Mersenne Twister PRNGs allows to produce uncorrelated sequences of pseudo-random numbers with no practical limitation on the realisation period, which in turn allows to perform long simulation sessions on parallel machines.

The future work is to make ARMA mathematical apparatus and its numerical implementation a base of virtual testbed for marine objects dynamics studies.

References

1. M. St. Denis, W.J. Pierson Jr., On the motions of ships in confused seas. Tech. rep., New York Univ. Bronx school of engineering and science (1953)
2. M. Rosenblatt, A random model of the sea surface generated by a hurricane. Tech. rep., DTIC Document (1956)
3. A.A. Sveshnikov, Math. Akad. Mechanics and Engineering 3, 32 (1959)
4. M.S. Longuet-Higgins, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 249(966), 321 (1957)
5. N. Kochin, I. Kibel, N. Roze, Theoretical hydrodynamics [in Russian] (FizMatLit, 1966)
6. R.F. Beck, A.M. Reed, P.D. Sclavounos, B.L. Hutchison, Transactions-Society of Naval Architects and Marine Engineers 109, 1 (2001)
7. G.E. Box, G.M. Jenkins, Time series analysis: forecasting and control, revised ed (Holden-Day, 1976)
8. M. Kostecki, Stochastic model of sea waves. Ph.D. thesis, CTO, Gdansk (1972)
9. V.A. Rozhkov, Y.A. Trapeznikov, Probabilistic models of oceanographic processes (Gidrometeoizdat, Leningrad, 1990)
10. A.T. Gurgenidze, Y.A. Trapeznikov, Probabilistic model of wind waves (Gidrometeoizdat, Leningrad, 1988), pp. 8–23
11. P.D. Spanos, ARMA Algorithms for Ocean Spectral Analysis (University of Texas at Austin. Engineering Mechanics Research Laboratory, 1982)
12. P.D. Spanos, B. Zeldin, Earthquake engineering & structural dynamics 25(5), 497 (1996)
13. F. Fusco, J.V. Ringwood, IEEE Transactions on sustainable energy 1(2), 99 (2010)
14. A. Degtyarev, I. Gankevich, in Proceedings of 11^{th} International Conference on Stability of Ships and Ocean Vehicles, Athens (2012), pp. 841–852
15. A.B. Degtyarev, , A.B. Podoliakin, in Proc. of II int. conf. on shipbuilding (ISC'98), vol. V (Saint-Petersburg, 1998), vol. V, pp. 416–423
16. A. Degtyarev, A. Boukhanovsky, Analysis of peculiarities of ship-environmental interaction. Tech. Rep. 09-97-1AB-1VA, Strathclyde University, Ship Stability Research Center, Glasgow (1997)
17. P. Boccotti, Meccanica 18(4), 205 (1983)
18. A.B. Degtyarev, A.M. Reed, Proceedings of the 12th International Ship Stability Work-shop (2011)
19. A.B. Degtyarev, A.M. Reed, International Shipbuilding Progress 60(1-4), 523 (2013)
20. A.V. Boukhanovsky, Probabilistic modeling of wind wave fields taking into account their heterogeneity and nonstationarity. Ph.D. thesis, Saint Petersburg State University (1997)
21. Wolfram Research, Inc., Mathematica. Champaign, Illinois (2016)
22. M. Matsumoto, T. Nishimura, ACM Transactions on Modeling and Computer Simulation (TOMACS) 8(1), 3 (1998)
23. M. Matsumoto, T. Nishimura, Monte Carlo and Quasi-Monte Carlo Methods 2000, 56 (1998)
24. A.V. Oppenheim, R.W. Schafer, J.R. Buck, et al., Discrete-time signal processing, vol. 2 (Prentice hall Englewood Cliffs, NJ, 1989)
25. D. Svoboda, in Image Analysis and Processing–ICIAP 2011 (Springer, 2011), pp. 453–462
26. K. Pavel, S. David, Algorithms for efficient computation of convolution (INTECH Open Access Publisher, 2013)
27. T.L. Veldhuizen, M.E. Jernigan, in International Conference on Computing in Object-Oriented Parallel Environments (Springer, 1997), pp. 49–56
28. T. Veldhuizen, Computer science technical report 542, 60 (2000)

29. M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, F. Rossi, R. Ulerich, GNU Scientific Library Reference Manual, 3rd edn. (Network Theory Ltd., 2009). Eds. Brian Gough
30. K. Goto, R. Van De Geijn, ACM Transactions on Mathematical Software (TOMS) 35(1), 4 (2008)
31. K. Goto, R.A. Geijn, ACM Transactions on Mathematical Software (TOMS) 34(3), 12 (2008)
32. M.J. Kilgard. The opengl utility toolkit (glut) programming interface (1996). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.8270
33. A. Fabri, S. Pion, in Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems (ACM, 2009), pp. 538–539
34. clFFT developers. clFFT: OpenCL Fast Fourier Transforms (FFTs). https://clmathlibraries.github.io/clFFT/