Nonlinear Investigation and Shape Optimization of Wave Energy Converter "Wave Carpet"

by

Ninh Hai Do

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Mechanical Engineering

and the Designated Emphasis

in

Computational and Data Science and Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor M. Reza Alam, Chair
Professor Ömer Savaş
Professor Per-Olof Persson

Summer 2021

Nonlinear Investigation and Shape Optimization of Wave Energy Converter "Wave Carpet"

Abstract

Nonlinear Investigation and Shape Optimization of Wave Energy Converter "Wave Carpet"

by

Ninh Hai Do

Doctor of Philosophy in Engineering – Mechanical Engineering

and the Designated Emphasis in

Computational and Data Science and Engineering

University of California, Berkeley

Associate Professor M. Reza Alam, Chair

Wave energy converter "wave carpet" is an artificial flexible structure originally introduced and actively developed by the Theoretical and Applied Fluid Dynamics Laboratory (TAFLab) for nearly a decade. Wave carpet is submerged on shallow-water seabed and absorbs the energy of incoming surface waves. Several studies conducted by TAFLab members numerically and experimentally proved its concept and supported its technical feasibility.

Inheriting from the previous works, this thesis further extends the understanding of wave carpet by numerically investigating one of its possible configurations and heuristically optimizing its planar shape. The study concentrates on arbitrary-shaped wave carpet locally embedded on the seafloor. This type of wave carpet is simplified into a visco-elastic part of the surrounding rigid seabed. In order to prepare the simulation environment, we first solve the problem of wave-viscoelastic seabed interaction using the High-Order Spectral (HOS) method. As a result, the solution presents the propagation of periodic waves over the entire visco-elastic ocean bottom. Wave carpet is subsequently limited to a partial seafloor, and the simulation domain is repetitively wave-fed and marched over time by Runge-Kutta method. Eventually, the procedure results in a robust numerical model of a wave tank with a wave carpet of an arbitrary shape. The model is validated against several criteria to ensure its correctness. It is referred to as the HORSK model that resembles a wave tank.

The HOSRK wave tank is numerically stable, three-dimensional and possibly highly nonlinear. It serves as a computational environment simulating the real-world scenarios in which incoming waves propagate over a wave carpet and damp out. Through many experiments with the numerical wave tank, we find the optimal water deepness and the optimal restoring force and damping ratio, two main characteristics of the wave carpet, for the maximum

energy absorption. We further investigate a wide range of linear and nonlinear elliptical wave carpets, and we find that toward the optimal point the difference between linear and nonlinear carpets in terms of energy absorption is subtle.

To optimize the wave carpet shape, we propose the Neural Network-based optimization method for linear optimization and the Genetic Algorithm for nonlinear optimization. The results show that in most wave cases the optimal-shaped carpets can absorb approximately twice more energy than the baseline circular shape, and the nonlinear optimal shapes are very similar to the linear optimal ones.

To the Lord of God.

And to my Family.

Contents

List of Figures

List of Tables

Nomenclature

α	Learning rate of neural network.
β	Wave carpet rotation angle.
ϵ, μ	Wave steepness and water deepness, respectively.
η	Elevation.
ϕ	Velocity potential.
ζ, γ	Dimensionless damping ratio and dimensionless restoring force, respectively.
A	Wave carpet area.
$b^\star, k^\star$	Damping and stiffness coefficients, respectively.
e	Relative error.
g, ρ	Gravitational acceleration and water density, respectively, both normalized to 1.
h	Water depth.
N_F	Number of Fourier modes.
P	Pressure.
p	Wave carpet elongation factor.
r_0, a_n, ϕ_n	Fourier representation parameters.
S_s, S_b	Quantity S evaluated at surface, bottom.
S_t	Derivatives of quantity S w.r.t time t.
$S_{x,y,z}$	Derivatives of quantity S w.r.t x, y, z, respectively.
T	Period.
t, x, y, z	Time and spatial coordinates.

Acknowledgments

My Doctor of Philosophy in Berkeley is a long enlightened journey with a full gamut of emotions. As it comes to its end, I am writing with nostalgia in order to thank those who left their marks on it.

First and foremost, I would love to extend my heartfelt thanks to my advisor, Professor Mohammad-Reza ALAM, for his consistent support and patience during my PhD time. Professor ALAM has proved himself to be a typical example in scientific research, the one who is radical, attentive, open-minded and interdisciplinary. He provided me with his professional advice, conscientious direction and valuable resources without which I have by no means been able to go that far.

Second, I would like to deeply thank Professors Ömer SAVAŞ, Per-Olof PERSSON and Fai MA for their supporting role in my PhD program without which the journey would have certainly been much tougher. I was fortunate enough to enroll in Professor SAVAŞ' two classes and he showed me the beauty of fluid and physical experiments. Fortunately, also in these classes I learned that experimental work is only for those who are brave enough and I am definitely not the one. Professor Per-Olof PERSSON inspired me about the numerical methods through his works. He is very friendly to students and approachable. Overall, they were very kind to accept my invitation to my PhD Committee.

Third, I sincerely express my gratitude and appreciation to La Shana POLARIS, coordinator in Department of Statistics, Donna SAKIMA, coordinator in Department of Physics, Professors Raja SENGUPTA and William IBBS in Department of Civil Engineering, Christina TOBOLSKI, coordinator in Department of Math, Professors Panayiotis PAPADOPOULOS, Hayden TAYLOR and, again, Mohammad-Reza ALAM in Department of Mechanical Engineering. They provided me with the generous financial support by offering me Graduate Student Instructor and Reader positions in every single semester throughout the entire duration of my PhD program. I cannot emphasize the importance of their support more but if at any point it had discontinued, I would have left the program.

Fourth, I sincerely thank my friends and labmates for bearing with me all time. Particularly, thanks to Ritwik GHOSH, Rachael HAGER and David FERNANDEZ for their help in my preparation for the qualification exam. Thanks to Alexandre IMMAS for his patient collaboration with me in our joint project, through which I learned a lot about planning, consistency and professional attitude in group work. Thanks to Qiu CHEN and Louis COUSTON for their guide at the beginning stage of my project. Thanks to Mehdi MIRZAKHANLOO for regularly administering the TAFLab meetings. Thanks to Yong LIANG, Shuangjiu FU and Michael KELLY for the social activities that left in me nothing but joyful moments.

Last, but not least, I would like to deeply thanks Dr. Thu DUONG, now a postdoctoral researcher at the University of Pennsylvania, and Mr. KAP Thanh Long for your invaluable initial financial support that encouraged me to come to Berkeley. When I got admitted into the school, I could not afford the expenses. It was them who gave me a loan in time that helped me partially cover the tuition fee of the first semester. Their help is not just money but their trust in me that I will never give up. And that trust follows me to this end.

Part I

Model

Chapter 1

Introduction

1.1 Energy Overview

Energy plays a vital role in the socio-economic development of all countries in the world. David Stern and Nikos Tsafos showed the stable and strong correlation between the energy use per capita and the GDP per capita of many countries over several decades [94, 96]. The close relationship is also observed between the energy growth rate and the GDP growth rate in many nations and in the world [9, 79, 48]. For the sake of economic development, most countries increase their energy demand every year, leading to the increase of the global power usage overall. Figure 1.1 shows the energy consumption by different regions in the world over in the period of 1965-2020 [28, 19]. While the energy growth in most parts of the world is relatively stable, that of Asia where most developing economies are located quadratically rises over the last decades, leading to the doubled world energy consumption in 2020 compared to that of 1980. Our projection indicates that by 2050 the energy demand gets doubled in Asia, resulting in the rise of the world energy use by approximately 60%.

Although the overall power usage steadily grows over years, the energy consumption breakdown shows different trends for different types of energy. Figure 1.2 presents the world energy consumption by source in the period of 1965-2019 [29, 19]. Fossil fuels, including oil, coal and natural gas, occupy the majority power needs while non-fossil energy sources, such as nuclear power and renewable energy, fill the remaining minority part. Renewable energy consists of biofuels, hydro power and new renewable energy of other types. Hydro power share takes roughly the same percentage every year, indicating that its technology is mature and it is widely accepted. Modern renewable energy of other types, on the other hand, emerges in recent decades with a very modest trend. As of 2018, renewable energy fills less than 13% of the total energy demand, out of which new renewable energy of other types shares less than 6% with roughly 7000TWh out of 160,000TWh despite its technical potential 20 times larger [55].

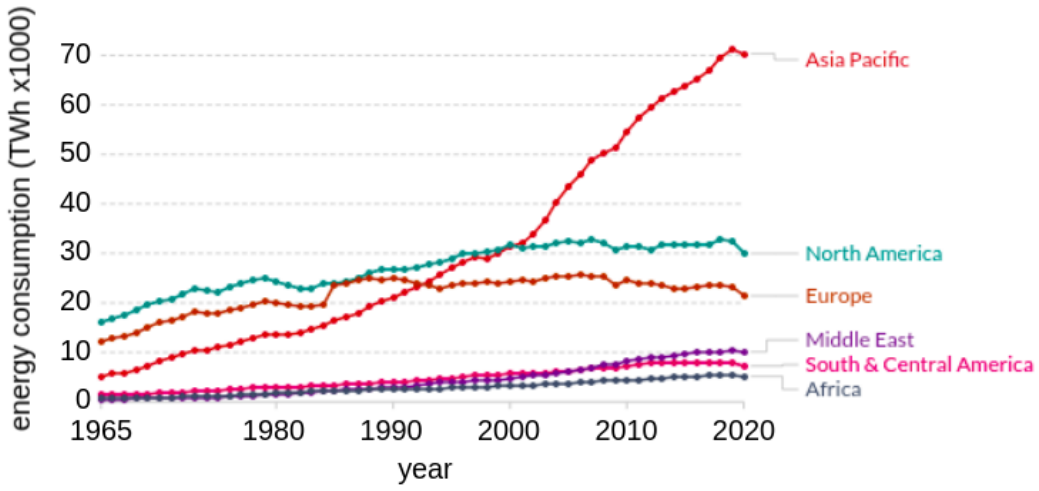


Figure 1.1: Primary energy consumption by world region in terawatt-hours (TWh) during the time interval of 1965-2020 [28, 19]. The regions expose different trends in different time ranges. Europe, North America and South & Central America reached a plateau in 1900s, 2000s, 2010s, respectively. Asia Pacific, Middle East and Africa follow nonlinear rising trends. Notably, Asia Pacific has soared since 2000s.

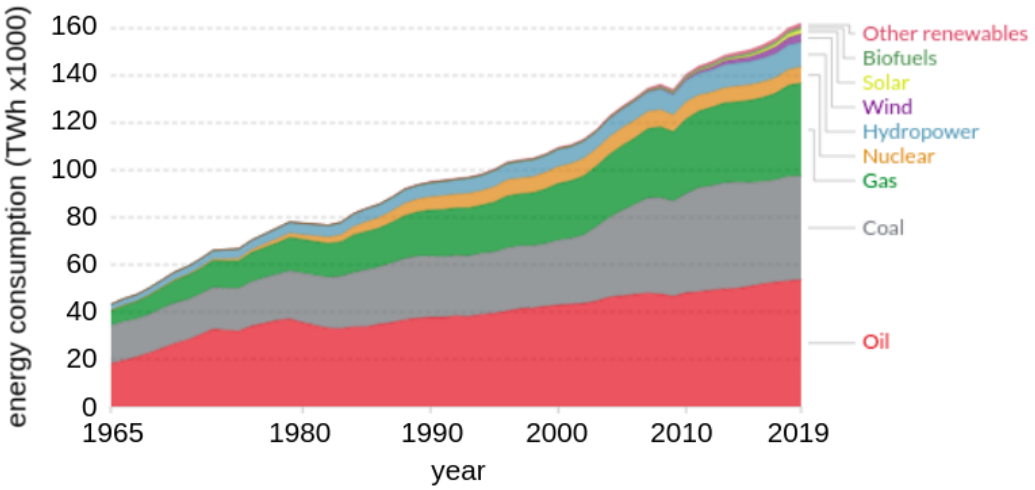


Figure 1.2: World energy consumption by source in terawatt-hours (TWh) over the time span of 1965-2019 [29, 19]. Fossil fuels, including oil, coal and gas, take the majority part over years. Nuclear and hydro-power needs tend to stabilize in the last decades. New renewables, covering all the rest, are emerging but their scale is relatively small compared to that of the traditional energy sources.

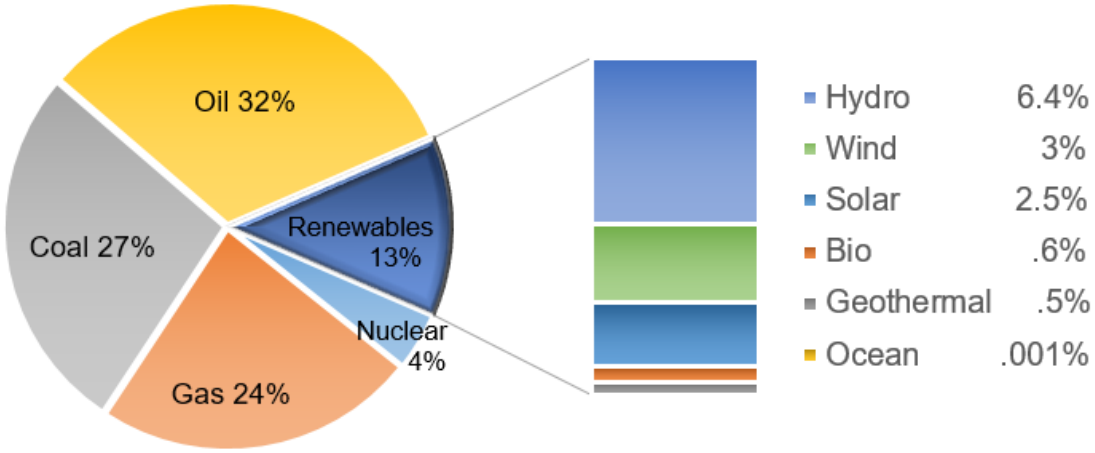


Figure 1.3: Energy shares by source in 2018 and renewable energy breakdown [29, 78]. It is typical in the recent years that the renewables take up 11-13% of the total energy needs. In 2018 wind and solar powers are relatively comparable to hydro-power. The triplet fills most of the renewables share. Biofuels and geothermal energy take up roughly 1% while ocean power is almost negligible.

Modern renewable energy covers different energy types including hydro-power, biofuels, wind, solar, geothermal and ocean energy. Figure 1.3 presents the energy shares in 2018 and renewable energy breakdown. The picture is also typical of the recent years with slight changes in percentages. In 2018, solar and wind energy takes 2.5% and 3%, respectively, and accounts for more than half of this type. Solar technologies include photovoltaics and concentrating solar power. Wind power uses wind turbines with horizontal or vertical axis of different designs. The other types of new renewables are biofuels, geothermal and ocean energy having much smaller shares, among which ocean power has the smallest scale despite its abundant technical potential comparable to the other sources.

1.2 Ocean Energy

Ocean power has enormous potential reserve of roughly 80000 TWh per year, mostly attributing to wave energy. It has many advantages over the other types of new renewable energy [10]. Compared to wind and solar energy, ocean power has higher density and can be exploited around the clock. Ocean resources along coastlines are abundant and in close proximity to populous areas. Ocean power can also come in different form including ocean current, tidal, wave, ocean thermal and osmotic energy, offering a wide choice of potential technology developments. Despite the advantages, ocean energy occupies the smallest share of energy by source. Most technologies are not mature and many projects are still in pilot

phase. The hurdles to its development come from technical difficulties due to harsh exploitation environment and the delay of favorable policies by government. Nowadays, when wind and solar technologies are mature, more interest and attention are shifted toward ocean energy, making it a potential field for research and development.

Tidal and wave energy are the main focus of research and development. The global potential of tidal energy is estimated at 1200 TWh per year in supply and 3000 GW in capacity. High and low tide is cyclic with the periods of the sun and the moon, so it is highly predictable, making engineering work easier to design efficient and sustainable systems. Tidal technologies are approaching its maturity with the typical design of horizontal-axis turbine installed on sea bed or a platform. As of 2016, two tidal barrages in France and Korea account for more than 90% of total installed capacity. The research and development (R&D) efforts come mostly from Europe, North America and China.

The global potential of wave energy is estimated at 8000 - 80000 TWh in supply. Its power density varies according to coastlines, typically 30kW/m^2, which is tens times greater than that of wind and solar energy, making it the most concentrated form of renewable energy [24]. Compared to tidal power, it is more promising but lagging behind. Wave power does not have the mature technologies due the complexity of wave ocean conditions and the lack of attention until recent years. Wave power is currently an active R&D area with various designs corresponding to different operating principles. The most common approaches include point absorber buoys, surface attenuators, oscillating water columns, and overtopping devices. The R&D activities shift from point wave energy converters to series. More designs come every year thanks to the governmental support of developed countries and design competitions. Most projects are, however, still in pilot phases and have not been realized in large scale yet.

1.3 Wave Carpet

Ocean energy gains more and more interest in the last decades [65]. Among its different forms, wave energy is under-exploited due to its immature technology. The harsh operating environment poses the technical difficulties to devising the effective ocean wave energy harvesting systems, increases the manufacturing cost and makes their scale-up more challenging. In an effort to address these issues, TAFLab developed the theory of wave carpet, an artificial structure that is embedded underwater and convert into energy the ocean waves propagating over itself. Due to its submergence and flexible architecture, the wave carpet can effectively avoid the impact of rough weather and have a great potential for scaling-up.

Inspired by the natural phenomenon that the offshore ocean waves are significantly damped when propagating over a muddy bank, TAFLab designed the structure that mimics the muddy bank behavior. The structure is composed of several damping-elastic springs connected in 2-dimensional series to make a wave carpet that can take any shape. Due to

its elasticity, the wave carpet will oscillate under the impact of hydrodynamic force cause by the surface waves, and absorb their energy thanks to the damping effect. On one hand, the surface waves significantly reduce their amplitude after passing the wave carpet, thereby posing less threat to the coastline. The wave carpet, on the other hand, have flexible structure and is submerged underwater so not exposing to the extreme weather above the surface. This considerably reduces the negative impacts of the harsh environment on the structure.

In 2012 Alam carried out the nonlinear analysis of a seafloor-mounted carpet for wave energy extraction [2] and introduced the concept of synthetic seabed. The early-stage wave carpet took over the whole length of computational domain. In 2014 Elandt et al. numerically and experimentally showed that a proper setup of seabed topology can focus and de-focus surface gravity wave [36]. The seabed can serve as a lens or curved mirror for wave energy. In the same year Boerner presented a real time hybrid model for wave carpet and optimized the power takeoff (PTO) parameters as well as the positioning of the PTO units [14]. In 2017 Alam et al. patented Carpet of Wave Energy Conversion (CWEC) [3].

As a part of the wave carpet theory, the thesis is mainly focused on simulating the working wave carpet in different wave cases for optimization purpose. Following [33] and [2, 70] we will implement a high-order, robust numerical method for modeling nonlinear gravity waves and visco-elastic bottom. Such a method is important for the application of using a "mud-resembling" seabed carpet to extract ocean wave energy. Subsequently, we use the update-stepping algorithm to simulate the working condition of the wave carpet in reality, that is, the incoming waves propagate from one side, over the carpet and damped at the end of the simulation domain. During the implementation, various validations are carried out to ensure the modelling correctness at each stage, resulting in the total correctness at the end. The final product will serve as a reliable approximate model of the wave carpet for the optimization purpose.

1.4 Related Works

There have been many studies on related topics, investigating wave-mud interaction and wave motion over submerged plate. Early experimental works showed that up to 80% of wave energy can be absorbed by a muddy bed through the propagation over just a few wavelengths. Sheremet et al. observed that mud-induced significant damping happens to both long and short waves [90]. Alam et al. numerically investigated the dissipation of broadband waves over muddy seabed. Shamsnia presented the analytical solutions to the wave-current-mud interaction problem [89]. Safak et al. modelled wave-mud interaction using nonlinear wave spectral model [85, 86]. Shibayama and An proposed a visco-elastic-plastic model for wave-mud interaction and further extended it to wave-current-mud interaction with validated experimental results [91, 73].

In an effort to look for a physical model for wave-mud, we found very few studies on waves over flexible structures but numerous one on the interaction of surface waves and a submerged rigid board. Liu et al. presented a beautiful analytical solution to the wave problem over a submerged horizontal plates [69, 68]. Beji and Battjes experimentally investigated nonlinear wave propagation over a horizontal bar, presenting different nonlinear wave profiles [11]. Liu et al., He et al., and Hsu et al. solved the wave-submerged plate interaction problem using numerical wave tank, sparse hydrodynamic particle method and boundary element method, respectively [67, 49, 52]. Karmakar et al. studied the inverse problem of using submerged pitching plate to control wave motion [57]. Windt et al. numerically investigated the 3D effects of a wave tank on the accurate determination of wave excitation force on a submerged rigid board, and validated the results with experimental data [101].

Although there have been extensive studies on wave-rigid board, few researchers paid attention to the rigid board optimization. Two related optimizations are conducted in-house by TAFLab members. Kelly et al. optimized the shape of a submerged wave energy converter to reduce the hydrodynamic loads [59]. Soheil et al. went the furthest to optimize the shape of submerged rigid board to maximize its energy absorption using BEM-based NEMOH package [37].

1.5 Thesis Outline

The thesis is structured the way that tells a story about my research course. It consists of 3 parts covering 7 chapters.

- **Part 1** includes Chapters 1, 2 and 3. This part proposes a new method of harvesting energy. It is focused on building the model and validating its performance in a computational environment.
- **Part 2** includes Chapters 4 and 5 concentrating on various optimization methods, each of which was attempted to solve the optimization problem. Each method has its pros and cons, but any choice among them can solve the problem to different extent of satisfaction.
- **Part 3** includes Chapters 6, 7 and 8 presenting the results of solving the linear and nonlinear optimization problems. It ends with the discussion of future work and the conclusion.

Chapter 1 offers a general overview about energy with focus on ocean energy and wave carpet. Despite the consistently increasing demand of energy over years and the huge potential of ocean power, tidal energy occupies a very small portion in the world energy picture while wave energy is not yet commercially exploited. Extending the effort of promoting

ocean energy, we propose the wave carpet model extracting wave energy. Wave carpet is a collaborative TAFLab work asynchronously developed by different members. This work focuses on the computational aspect of the wave carpet.

Chapter 2 describes the modelling procedures of wave carpet. We present its physical model and mathematical model. The former models wave carpet as a two-dimensional grid of connected damping springs with the massless carpet on top, mimicking muddy bank. The motion of the artificial wave carpet is governed by the physics of springs while the covering massless carpet connects the springs and transfer motions between them. The mathematical model further simplifies the physical wave carpet by embedding it into the seabed.

Chapter 3 presents various validations for the numerical solution to wave carpet. First, the numerical result is validated against the analytical solution for the nearly matching. Second, various simulations with carpet of different shapes are conducted to find the stability points for each case. Third, the energy conservation law is tested on the simulations to make sure the energy is preserved. Last, the convergence test is conducted to ensure the convergence of the numerical solution.

Chapter 4 presents the tools and optimization methods that are attempted to optimized the wave carpet shape. These include random search method, batch gradient descent, cross entropy method and genetic algorithm. All of these methods are random-based, in which the first one is sub-optimal while the other three are heuristically optimal.

Chapter 5 proposes the neural network-based method to optimize the carpet shape. We first give a background on neural network and its mathematical representation. Next, we discuss the method is used to train a neural network. We subsequently discuss fully-connected neural network, a typical neural network that is usually used to approximate complex function. In our case, it is the wave carpet numerical solution that is need to be replaced by an approximating light-weight function for optimization purpose. Finally, the complete neural network-based optimization algorithm is presented.

Chapter 6 outlines the software specifications. The software development takes up the majority workload of the project and its developed framework is a valuable legacy for further research and development. The deliverables include source code in different wave cases.

Chapter 7 presents the results of linear and nonlinear optimization. Before the optimization, all the hyper-parameters are tuned for the best performance. The dimentionless restoring force and damping ratio, for instance, are tuned beforehand for the benchmark circular-shaped carpet. These optimal values are fixed to further optimize the carpet shape parameters. The linear optimization was done using the neural network-based method while the nonlinear optimization was implemented using various methods, among which only the results of genetic algorithm and cross-entropy methods are reported.

Chapter 8 concludes the work with the open discussion of future work and the concise summary of achievements. The work is finally ended with the conclusion on the achievements on wave carpet. The intended future work should be considered an extended project, in which advanced machine learning techniques can potentially be applied to automate the tuning of carpet hyper-parameter adapting to different wave conditions.

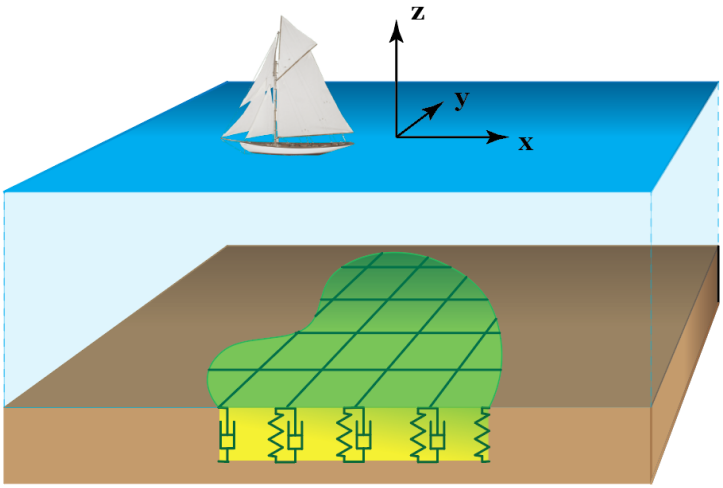
Chapter 2

Wave Carpet Model

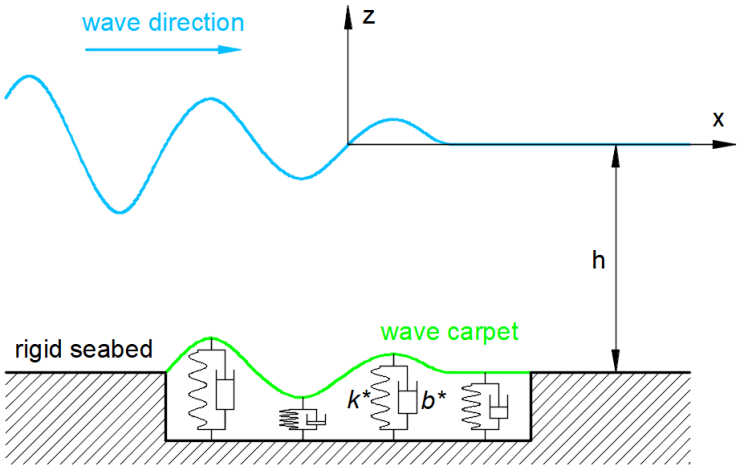
2.1 Physical Model

The wave carpet essentially simplifies the dynamics of a muddy bank by just imitating one of its most important characteristics for energy absorption, visco-elasticity. This physics can be achieved by a dense network of damping springs beneath the connecting thin and light-weight carpet layer. Figure 2.1a illustrates the conceptual design of a wave carpet with an arbitrary shape and Figure 2.1b shows its side view along with the core components.

The wave carpet shape is defined as the geometric shape in x-y plane. It is a flat 2-dimensional shape at the initial state when the springs are at rest. Under the sea load, the carpet modulates in accordance with the surface wave motion. The carpet motion is regulated by the the oscillation of underneath springs in vertical direction.



(a) 3-D scheme of wave carpet embedded on the sea floor in static condition. It takes an arbitrary shape supported by a grid of strings.



(b) Side-view composition of wave carpet in working condition. Springs underneath carpet layer are characterized by stiffness $k*$ and damping ratio $b*$.

Figure 2.1: Conceptual model of wave carpet is composed of a grid of connected damping springs. The springs are characterized by damping ratio and restoring force. The covering carpet mass is assumed to be negligible compared to hydrodynamic loads. The wave carpet can take and arbitrary shape.

2.2 Mathematical Formulation

Governing Equations

The visco-eleastic wave carpet is modeled as a grid of submerged springs connected together by a covering carpet. Its dynamics is thus governed by the fundamental mass-spring damping equation [77, 95]. The fluid environment is assumed to be inviscid, incompressible and irrotational. This assumption is valid for large-scale ocean waves as the Reynolds number $\mathrm{Re} = \dfrac{\rho\nu L}{\mu} \gg 1$ [72, 64, 99]. The hydrodynamics of ocean waves is governed by the Laplace equation for mass continuity. These two governing equations are subject to 4 kinematic and dynamic boundary conditions on the surface and bottom.

Setting $z = 0$ on the surface and $z = -h$ at the bottom with depth h, denoting $\eta_s(x, y, t)$ and $\eta_b(x, y, t)$ as the surface and bottom, respectively, representing the fluctuations around their mean values $0, -h$, the governing equations in terms of the velocity potential ϕ are written as follows [2, 4, 70]:

$$\nabla^2\phi = 0 \qquad -h + \eta_b < z < \eta_s \qquad \text{(mass continuity)} \tag{2.1}$$

$$b^\star\eta_{b,t} + k^\star\eta_b + P_b = 0 \qquad \text{at } z = -h + \eta_b \qquad \text{(damping mass-spring)} \tag{2.2}$$

subject to:

$$\begin{cases} \eta_{s,t} + \eta_{s,x}\phi_x + \eta_{s,y}\phi_y = \partial_z\phi & \text{at } z = \eta_s & \text{(kinematic surface BC)} \\ \phi_t + \dfrac{1}{2}\left(\phi_x^2 + \phi_y^2 + \phi_z^2\right) + g\eta_s = 0 & \text{at } z = \eta_s & \text{(dynamic surface BC)} \\ \partial_t\eta_b + \eta_{b,x}\phi_x + \eta_{b,y}\phi_y = \partial_z\phi & \text{at } z = -h + \eta_b & \text{(kinematic bottom BC)} \\ \phi_t + \dfrac{1}{2}\left(\phi_x^2 + \phi_y^2 + \phi_z^2\right) + g\eta_b + \dfrac{P_b}{\rho} = 0 & \text{at } z = -h + \eta_b & \text{(dynamic bottom BC)} \end{cases} \tag{2.3}$$

Notice that the inertia term is missing in the damping mass-spring Equation 2.2 as the wave carpet is further assumed to be so thin that its mass is negligible. Its inertial force is thus assumably small compared to the hydrodynamic loads, which considerably simplifies the problem. Also note that in this equation the damping coefficient $b^\star$ and the spring stiffness $k^\star$ are coupled with the bottom and the elevation, respectively, to govern its damping and restoring behavior.

These equations give a complete physical description of the system but they cannot be solved analytically. The equations is then rewritten in the form that facilitate the numerical methods.

System of Equations for Time Iteration

Since the surface and bottom behavior are of our interest, particularly the surface and bottom elevations and velocity potentials, the above system of governing equations is written in terms of the velocity potential ϕ at the bottom and surface following the approach by Alam [2]. Define:

$$\begin{cases} \Phi_s(x,y,t) = \phi(x,y,\eta_s,t) \\ \Phi_b(x,y,t) = \phi(x,y,-h+\eta_b,t) \end{cases} \tag{2.4}$$

where Φ's are the velocity potential at the surface and bottom elevations.

Manipulate algebra, the system of Equations 2.2 and 2.3 can be reduced in the form:

$$\begin{cases} \eta_{s,t} = -\eta_{s,x}\Phi_{s,x} - \eta_{s,y}\Phi_{s,y} + \left(1+\eta_{s,x}^2+\eta_{s,y}^2\right)\phi_z & \text{at } z=\eta_s \\ \Phi_{s,t} = -g\eta_s - \frac{1}{2}\left(\Phi_{s,x}^2+\Phi_{s,y}^2\right) + \frac{1}{2}\left(1+\eta_{s,x}^2+\eta_{s,y}^2\right)\phi_z^2 & \text{at } z=\eta_s \\ \eta_{b,t} = -\eta_{b,x}\Phi_{b,x} - \eta_{b,y}\Phi_{b,y} + \left(1+\eta_{b,x}^2+\eta_{b,y}^2\right)\phi_z & \text{at } z=-h+\eta_b \\ \Phi_{b,t} = g\left(\frac{1}{\gamma}-1\right)\eta_b - \frac{1}{2}\left(\Phi_{b,x}^2+\Phi_{b,y}^2\right) + \frac{1}{2}\left(1+\eta_{b,x}^2+\eta_{b,y}^2\right)\phi_z^2 & \\ \quad + \sqrt{gh}\zeta\left[-\eta_{b,x}\Phi_{b,x} - \eta_{b,y}\Phi_{b,y} + \left(1+\eta_{b,x}^2+\eta_{b,y}^2\right)\phi_z\right] & \text{at } z=-h+\eta_b \end{cases} \tag{2.5}$$

In the new system of equations the parameters $b^\star$ and $k^\star$ are replaced with expressions involving the dimensionless damping ratio γ and restoring force ζ, respectively:

$$\zeta = \frac{b^\star}{\rho\sqrt{gh}}, \qquad \gamma = \frac{\rho g}{k^\star} \qquad \text{and} \qquad \mu = kh \tag{2.6}$$

Notice from the system of Equations 2.5 that the LHS contains the derivatives with respect to time while the RHS involves the coupling derivatives with respect to space. Each iteration yields the values of the 2-dimensional field Φ and η at the bottom and surface. To advance by one time step, it is necessary to have the value of ϕ_z, the vertical velocity of the fluid, evaluated at the surface $z=\eta_s(x,y)$ and bottom $z=-h+\eta_b(x,y)$. This will be done through an iterative procedure over the components of the perturbation series of ϕ in the spectral domain and the proposed solution of the velocity potential by Alam that satisfies the system of boundary conditions [2]. The whole procedure is known as High-Order Spectral method introduced by Dommermuth [33] and discussed in the following section.

2.3 Numerical Method

High-Order Spectral Method

The goal of High-Order Spectral (HOS) approach is to obtain the unknown velocity term ϕ_z in the system of Equations 2.5. The core idea of the method lies in the spectral representation of the velocity potential that essentially transforms the the variable from the space domain in to the frequency domain. In the new domain, the derivatives of ϕ can be taken easily with respect to space, including the desirable velocity ϕ_z at the surface and bottom. The interested variables are then transformed back from the frequency domain to space domain for the time-marching purpose.

To begin, write velocity potential ϕ as a perturbation series:

$$\phi(x, y, z, t) = \sum_{m=1}^{M} \phi^{(m)}(x, y, z, t). \tag{2.7}$$

the vertical velocity ϕ_z can then be computed to M^{th} order:

$$\phi_z(x, y, z = z^\star, t) = \sum_{m=1}^{M} \sum_{p=0}^{M-m} \frac{\eta_b^p}{p!} \frac{\partial^{p+1}}{\partial z^{p+1}} \phi^{(m)}(x, y, z, t)|_{z=z^\star}, \quad \text{for } z^\star = \eta_s \text{ and } -h + \eta_b. \tag{2.8}$$

To get all $\phi^{(m)}$ and its p^{th}-order vertical derivatives evaluated at $z = 0$ and $z = -h$, iterate over the index m in the perturbation series. All the details are carefully provided in section 4 of the paper by Alam [2]. The steps are presented here in an way much closer to what will be actually implemented in our algorithm.

The main steps are as follows:

1. The iteration proceeds as follows:

   - *Initialization:* $m = 1$. Define $f^{(1)} = \Phi_s(x, y, t)$, $g^{(1)} = \Phi_b(x, y, t)$. Compute $\phi^{(1)}(x, y, z, t)$ from $f^{(1)}$ and $g^{(1)}$.
   - *Iteration:* $1 < m \leq M$. Define $f^{(m)}$ based on steps $1 \ldots m - 1$ as:

     $$f^{(m)} = -\sum_{p=1}^{m-1} \frac{\eta_s^p}{p!} \frac{\partial^p}{\partial z^p} \phi^{(m-p)}(x, y, z, t)|_{z=0} \tag{2.9}$$

     and similarly for $g^{(m)}$ using η_b and $z = -h$. Compute $\phi^{(m)}(x, y, z, t)$ from $f^{(m)}$ and $g^{(m)}$.

2. For each $m \in [1, M]$, computing $\phi^{(m)}$ based on $f^{(m)}$ and $g^{(m)}$ requires these steps:

a) Compute the corresponding Fourier amplitudes $\tilde{f}_n^{(m)}$ and $\tilde{g}_n^{(m)}$ for $n = (n_x, n_y) \in [-N, N]^2$;

b) Compute $A_n^{(m)}$ and $B_n^{(m)}$ for all $n = (n_x, n_y) \in [-N, N]^2$:

$$\begin{cases} A_n^{(m)} = \tilde{f}_n^{(m)} \\ B_n^{(m)} = \dfrac{1}{\tanh|\mathbf{k}_n|h} \left(\dfrac{\tilde{f}_n^{(m)}}{\cosh|\mathbf{k}_n|h} - \tilde{g}_n^{(m)} \right) \end{cases} \tag{2.10}$$

where $\mathbf{k}_n = \left(k_{n_x}, k_{n_y}\right)$ and $|\mathbf{k}_n| = \sqrt{k_{n_x}^2 + k_{n_y}^2}$.

c) Obtain the following expression for $\phi^{(m)}$ (but its computation is not needed):

$$\phi^{(m)}(x, y, z, t) = \sum_{n_x, n_y = -N}^{N} \left(A_n^{(m)} \frac{\cosh(|\mathbf{k}_n|(z+h))}{\cosh(|\mathbf{k}_n|h)} + B_n^{(m)} \frac{\sinh(|\mathbf{k}_n|z)}{\cosh(|\mathbf{k}_n|h)} \right) \mathrm{e}^{i\mathbf{k}_n \cdot \mathbf{x}} \tag{2.11}$$

where $\mathbf{x} = (x, y)$ are the coordinates in the horizontal plane. It is now possible to compute $\partial^p \phi^{(m)} / \partial z^p|_{z=0}$ and $\partial^p \phi^{(m)} / \partial z^p|_{z=-h}$ for all $p \in [0, m-1]$:

$$\begin{cases} \dfrac{\partial^{2p}}{\partial z^{2p}} \phi^{(m)} \Big|_{z=0} = \displaystyle\sum_{n_x, n_y = -N}^{N} |\mathbf{k}_n|^{2p} A_n^{(m)} \mathrm{e}^{i\mathbf{k}_n \cdot \mathbf{x}} & \text{(2.12a)} \\ \dfrac{\partial^{2p}}{\partial z^{2p}} \phi^{(m)} \Big|_{z=-h} = \displaystyle\sum_{n_x, n_y = -N}^{N} |\mathbf{k}_n|^{2p} \left(A_n^{(m)} \frac{1}{\cosh|\mathbf{k}_n|h} - B_n^{(m)} \tanh|\mathbf{k}_n|h \right) \mathrm{e}^{i\mathbf{k}_n \cdot \mathbf{x}} & \text{(2.12b)} \\ \dfrac{\partial^{2p+1}}{\partial z^{2p+1}} \phi^{(m)} \Big|_{z=0} = \displaystyle\sum_{n_x, n_y = -N}^{N} |\mathbf{k}_n|^{2p+1} \left(A_n^{(m)} \tanh|\mathbf{k}_n|h + B_n^{(m)} \frac{1}{\cosh|\mathbf{k}_n|h} \right) \mathrm{e}^{i\mathbf{k}_n \cdot \mathbf{x}} & \text{(2.12c)} \\ \dfrac{\partial^{2p+1}}{\partial z^{2p+1}} \phi^{(m)} \Big|_{z=-h} = \displaystyle\sum_{n_x, n_y = -N}^{N} |\mathbf{k}_n|^{2p+1} B_n^{(m)} \mathrm{e}^{i\mathbf{k}_n \cdot \mathbf{x}} & \text{(2.12d)} \end{cases}$$

The values of these p^{th}-order derivatives for $p \in [0, m-1]$ will enable to compute $f^{(m)}$ and $g^{(m)}$ according to Equation 2.9 and the values of the p^{th}-order derivatives for $p \in [0, N]$ at the end of the iteration will finally enable to compute ϕ_z according to Equation 2.8.

Above is the detailed explanation of the procedure. Steps 1 and 2 are a simple description of the iterative process required to compute $f^{(m)}$, $g^{(m)}$, or equivalently $A^{(m)}$ and $B^{(m)}$. In our algorithm, the values for $A^{(m)}$ and $B^{(m)}$ for all $m \in \{1 \dots M\}$ are stored instead of $f^{(m)}$, $g^{(m)}$, in order to minimize the number of operations to compute ϕ_z afterwards. Indeed, Equation 2.8 can be directly written in terms of $A^{(m)}$ and $B^{(m)}$.

In summary, given the space boundary conditions of elevation and velocity potential at the surface and bottom, it is possible to solve for velocity and evaluate the RHS of the system of Equations 2.5 at any order of nonlinearity. In the dimension of time, the Runge-Kutta method will be applied to march the system over time steps.

Runge-Kutta Method

The Equations 2.2 are solved on a bounded domain with periodic boundary conditions. We are interested in solving for for the surface and bottom elevations and velocity potentials, $\eta_s, \eta_b, \Phi_s, \Phi_b$ over time using the system of Equations 2.5. We notice that these equations are time-dependent partial differential equations, which we will numerically solve using the *method of lines*. In other words, we will discretize the spatial operators in order to compute an approximation to the right-hand side. Then, we will obtain a system of ordinary differential equations. This system of ODEs can then be solved using the Runge-Kutta method for for time integration.

Temporal discretization: For the standard explicit fourth-order Runge-Kutta method, supposing we know the solution $y(t_n)$, we compute the solution at the next timestep $y(t_{n+1}) = y(t_n + \Delta t)$ using four stages. The four stages $k_1, \ldots, k_4$ are given by

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \tfrac{\Delta t}{2}, y_n + \tfrac{\Delta t}{2} k_1\right) \\ k_3 &= f\left(t_n + \tfrac{\Delta t}{2}, y_n + \tfrac{\Delta t}{2} k_2\right) \\ k_4 &= f\left(t_n + \Delta t, y_n + \Delta t k_3\right), \end{aligned}$$

and then the new-time solution is given by

$$y_{n+1} = y_n + \Delta t \left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \right).$$

This method enjoys a relatively large region of stability, simple implementation, and fourth-order accuracy. It is shown in [33] that the CFL condition for solving the above equations with the fourth-order Runge-Kutta method is $\Delta t^2 \leq 8/|\mathbf{k}_N|$.

Spatial discretization: Suppose that we are given the values of the functions η_s, η_b, Φ_s, and Φ_b at a finite number of collocation grid points, $\mathbf{x}_i = (x_i, y_i)$, for $i = 1, 2, \ldots, 2N$. We then need to calculate an approximation of the right-hand side of Equations 2.5 in order to advance the solution one timestep using the temporal discretization described above. To do this, we use a *pseudo-spectral* discretization. By this we mean that we will calculate spatial derivatives in Fourier space and nonlinear terms in physical space. We find the following terms appear in the right-hand side: 1. η_s and η_b; 2. horizontal derivatives (*i.e.* ∂_x and ∂_y) of η_s, η_b, Φ_s, and Φ_b; 3. ϕ_z.

The quantities η_s and η_b are known and therefore do not require computation. In order to compute the horizontal derivatives, we perform spectral differentiation. Noticing that for a differentiable function f, we have the exact relationship $\widehat{\partial_x f} = i\xi\hat{f}$, where $\hat{f}$ denotes the continuous Fourier transform, and ξ is the variable in Fourier space. Analogously, for a grid function, we take the discrete Fourier transform, multiply each Fourier mode by a factor of ik (where k is the wave number), and then take the inverse Fourier transform. This results in an approximation to the derivative that has *spectral accuracy*. That is to say, the approximation error at the collocation points $\mathbf{x_i}$ scales like $\mathcal{O}(N^{-m})$ for every m. Products of such terms are computed simply by taking products in physical space.

We remark that there are two possible implementations of the above-mentioned spectral differentiation. It is a possible to construct a $N \times N$ dense *spectral differentiation matrix* D_N, such that multiplication by D_N results in the spectral approximation to the derivative. Such a matrix-vector multiplication requires $\mathcal{O}(N^2)$ operations. Instead, we opt to use the fast Fourier transform algorithm, which requires $\mathcal{O}(N \log N)$ operations per transform. We can therefore perform the spectral differentiation with two transforms, and one dot product (with linear complexity). Because of the constant term in the computational complexity, optimality of the matrix method or FFT method depends on the number of grid points. For large values of N, the FFT-based implementation results in faster runtimes.

It remains to compute the terms ϕ_z. Having written ϕ as a perturbation series

$$\phi(x, y, z, t) = \sum_{m=1}^{M} \phi^{(m)}(x, y, z, t),$$

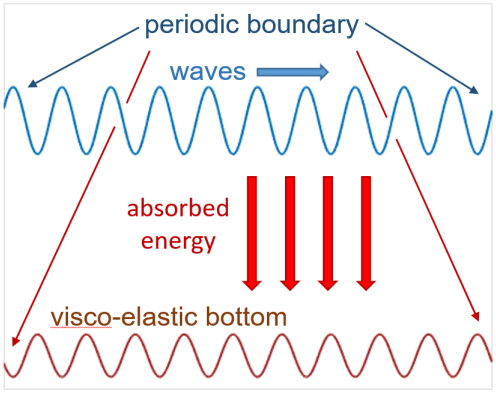
we can expand the Fourier series of $\phi^{(m)}$ as

$$\phi^{(m)}(x, y, z, t) = \sum_{n} \left(A_n^{(m)}(t) \frac{\cosh(|\mathbf{k}_n|(z+h))}{\cosh(|\mathbf{k}_n|h)} + B_n^{(m)} \frac{\sinh(|\mathbf{k}_n|z)}{\cosh(|\mathbf{k}_n|h)} \right) \mathrm{e}^{i\mathbf{k}_n \cdot \mathbf{x}},$$

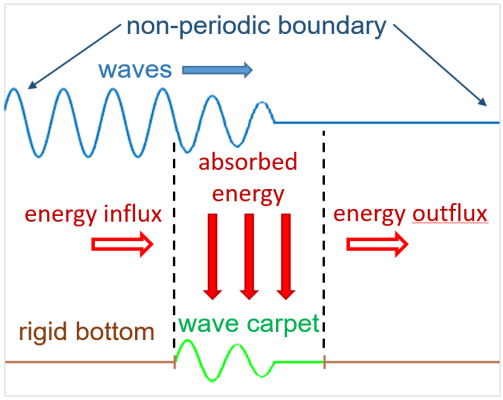
where the summation is taken over all pairs $n = (n_x, n_y)$, with $-N \leq n_x, n_y \leq N$, and the vector $\mathbf{x}$ represents the position in the horizontal plane, $\mathbf{x} = (x, y)$. Given this representation, all derivatives in x, y, and z can be taken explicitly, using known trigonometric formulas. The formulas obtains are given in Equations 2.12.

Wave Maker Algorithm

The wave carpet nonlinear system of equation is effectively solved in the space and time domain using the HOS method and Runge-Kutta numerical scheme. This methodology is based on the Fourier transform that assumes the periodic boundary domain. The real-life scenario is, however, not boundary-periodic, i.e. waves comes from the offshore side, propagates over the carpet and disperses onshore (see Figure 2.2b). In order to simulate this



(a) Scheme of periodic boundary simulation in which the end boundaries must match in x and y direction. Its solver and solution are referred to as HOS.



(b) Scheme of non-periodic boundary simulation in which waves can propagate from one side to the other. Its solver and solution are referred to as HOSRK or wave maker.

Figure 2.2: Periodic and non-periodic boundary solutions. The former is obtained by numerically solving the system of Equations 2.5 using High-Order Spectral method, called HOS solution. The latter is achieved by applying wave maker algorithm to the periodic boundary HOS solution, which is a combination of High-Order Spectral and Runge-Kutta methods, called HOSRK solution.

scenario, we use the numerical trick mimicking the wake maker. The entire procedure is outlined in the block diagram in Figure 2.3.

First, we generate a one-period wave profile consisting of 4 components: elevation and velocity potential at surface and bottom, called w. The wave profile w multiplied with a filter function f remains the wave part $f \cdot w$ at the beginning of the domain. This part serves as an analogous wave flapper in the wave maker. In the time loop, the HOS solver is used to march the wave domain at the current time step to the next timestep. The new wave profile multiplied with the filter function g remains the propagating wave which is subsequently fed by the periodic wave to form a for the next time step. The recurrent relation of the algorithm is given by:

$$\begin{cases} w_T = \{\eta_s, \phi_s, \eta_b, \phi_b\} & \text{for one } T \\ w_0 = w_{T_0} & \text{at } t = t_0 \\ w_{i+1} = \text{HOS}(g * w_i + f * w_{T_{i \mod T}}) & \text{at } t = t_{i+1} \end{cases} \tag{2.13}$$

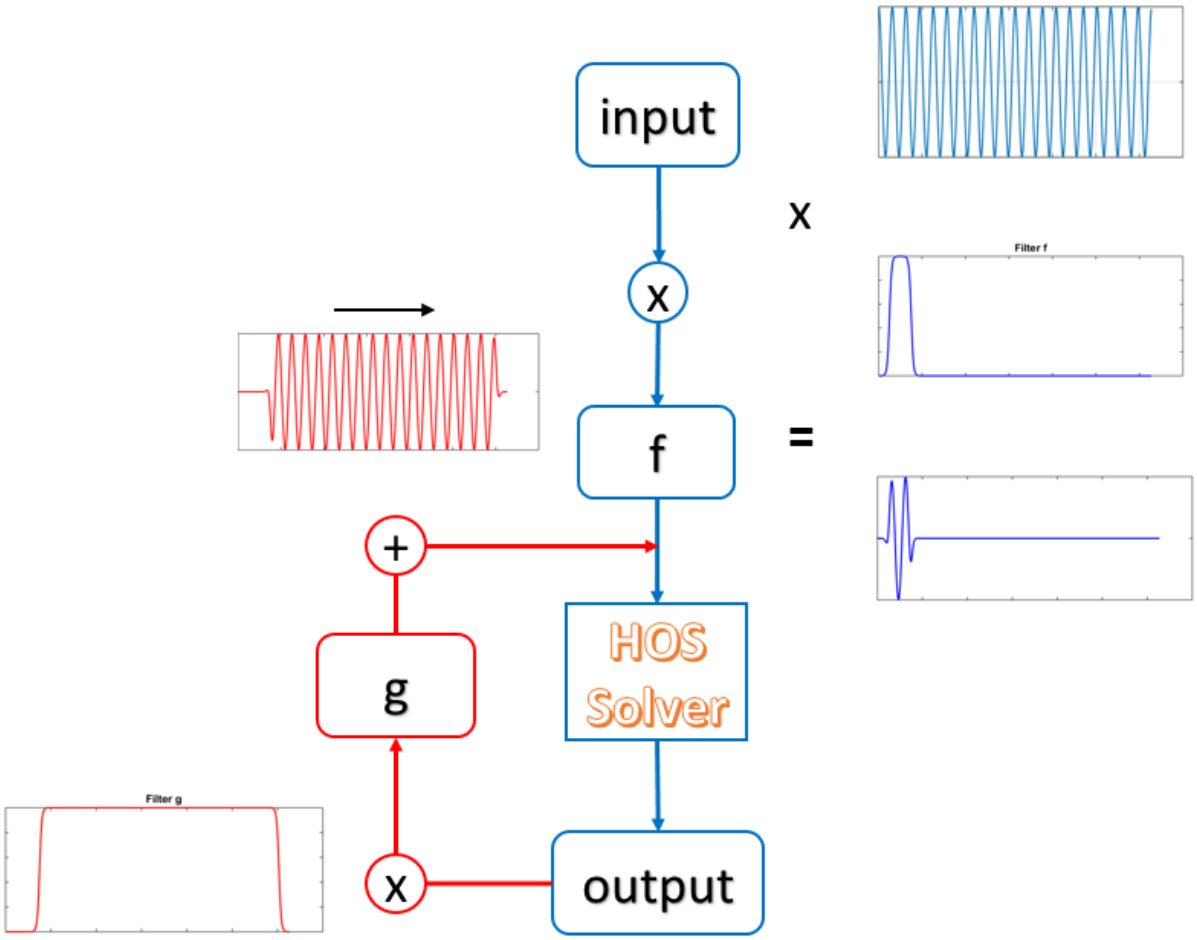


Figure 2.3: Block diagram for implementing wave-maker simulation. Periodic waves are filtered into a small group of waves that resembles those generated by a wave maker. Partial waves are fed into the HOS solver loop for their progression over time steps. Inside the HOS solver loop is another filter to ensure waves to damp out at the end, eliminating the reflecting waves.

The correctness of the numerical algorithm will be validated by the energy conservation law. That is, the average energy absorbed over time by the wave carpet should be equal to the reduction in energy flux over the wave carpet.

2.4 Wave Carpet Shape Representation

Up to this point we successfully simulate the computational wave maker analogous to the offshore incoming wave setting. We can embed the wave carpet anywhere at the domain bottom by setting the stiffness in the wave carpet region relatively low to that in the neighborhood, and the damping coefficient positive for the wave carpet, thereby dividing the domain bottom into two subregions: the undamped hard seabed and the visco-elastic wave carpet. For the 2-D computational domain, we use Fourier series to represent the carpet shape, in which the Cartesian coordinates x and y of the wave carpet is represented via the radius r in polar coordinates given by:

$$\begin{aligned} r(\theta) &= r_0 + \sum_{n=0}^{N_c} a_n \sin(n\theta + \phi_n) \\ x = pr\cos(\theta), &\qquad y = \frac{1}{p} r \sin(\theta) \end{aligned} \tag{2.14}$$

Thus, the carpet shape is represented via the parameters r_0, N_c terms of a_n, N_c terms of ϕ_n and p. To avoid complicated shapes and to ensure the carpet fit the computational domain, we set the constraints on the carpet parameters as follows:

$$\begin{aligned} r_0 &\in [0.4, 0.6] \\ a_n &\in [-0.4r_0, 0.4r_0] \\ \phi_n &\in [0, 2\pi] \\ p &\in [0.5, 2] \end{aligned} \tag{2.15}$$

In addition, we want to ensure that the carpet does not have loop. That is, r is always positive over the range $[0, 2\pi]$ of θ. This constraints involves the relationship between r_0 and the set of a_n. Obviously, the loose constraint of a_n in 2.15 cannot ensure that, while a more strict constraint on a_n, say $a_n \in [-\frac{r_0}{N_c}, \frac{r_0}{N_c}]$, eliminates several potentially good shape since it generates only simple ones. On one hand, it is not a trivial math problem to derive the equation for the correct no-loop constraint. On the other hand, this may result in a complicated nonlinear constraint that cause troubles for our optimization strategy in the next part. Thus, the no-loop constraint is treated as the on-the-fly condition that will be checked in the optimization running time.

The ultimate purpose is to optimize the wave carpet shape in order to maximize the energy absorption. Combining the aforementioned numerical methods and the shape representation, we have a HOS Runge-Kutta (HOSRK) solver that, given the carpet shape parameters as input, the solver outputs the absorbed energy. Eventually, we come up with the optimization problem:

$$\arg\max_{r,a_n,\phi_n,p} HOSRK(r, a_n, \phi_n, p) \tag{2.16}$$

subject to:

$$\begin{aligned} &A = const \\ &\text{All constraints in 2.15} \end{aligned} \tag{2.17}$$

The new constraint $A = const$ is to ensure that all the considered carpet shape have the same area for energy comparison purpose. The optimization problem is solved using various methods that will be thoroughly discussed in section 4.

Chapter 3

Wave Carpet Validation

3.1 Numerical versus Analytical Results

The numerical results are validated against the linear analytical solution. By setting the nonlinear mode parameter $M = 1$ to eliminate all of the nonlinear terms, the system of equations 2.5 reduces to the linear system that is numerically solved by the HOS method. In this section, various tests and error analyses are conducted to validate the linear HOSRK solver.

First, the numerically obtained elevation and velocity potential are compared to those of the linear analytical solution presented by Alam [2], according to which the surface and bottom elevations, and the velocity potential are given by:

$$\eta_s = a_s e^{i(kx-\omega t)} \tag{3.1}$$

$$\eta_b = a_b e^{i(kx-\omega t)} \tag{3.2}$$

$$\phi = (Ae^{kz} + Be^{-kz})e^{i(kx-\omega t)} \tag{3.3}$$

where,

$$A = -ia_s \frac{\omega^2 + gk}{2k\omega}, \qquad B = ia_s \frac{\omega^2 - gk}{2k\omega} \tag{3.4}$$

$$a_b = a_s \cosh kh \left(1 - \frac{gk \tanh kh}{\omega^2}\right) \tag{3.5}$$

The relative error between analytical and numerical results is defined as:

$$rel.\ err. = \frac{|\eta_{analytic} - \eta_{numeric}|}{\eta_{analytic}} \tag{3.6}$$

The maximum relative errors of surface and bottom elevations in one period are plotted against the number of timesteps per period in Figure 3.1. They are close to each other and

very small compared to 1, indicating the good match of the analytical and numerical elevations. The errors decrease when the timestep number increases, i.e. the time discretization gets finer, and approaching the double precision of floating point 10^{-16}.

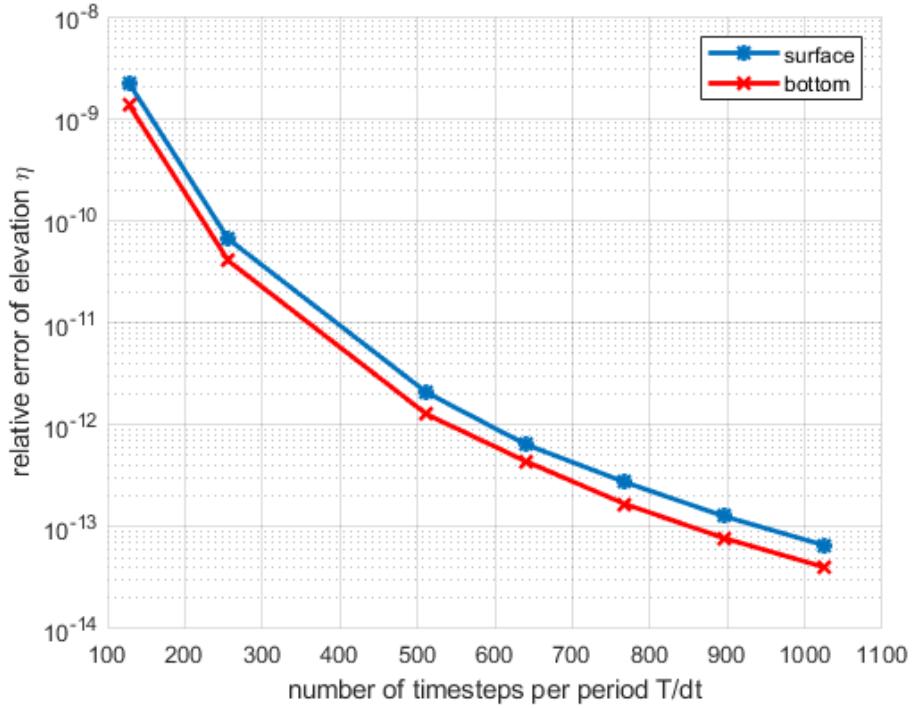
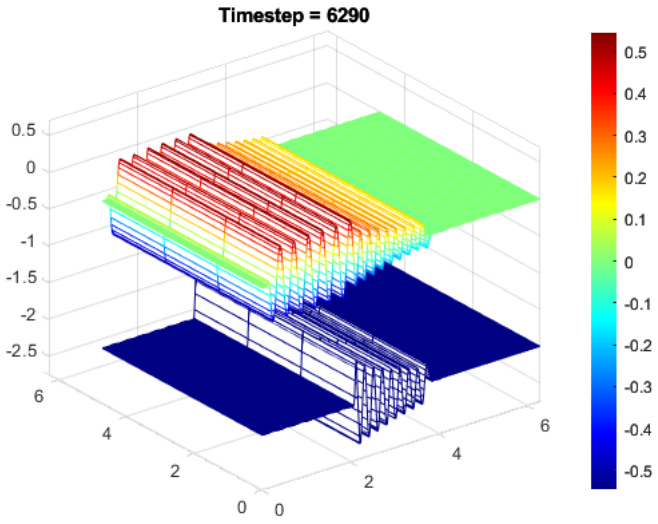


Figure 3.1: Error analysis of the HOS periodic boundary results.The maximum deviations of analytical and numerical elevations are plotted over the number of time steps per period. When the time gets finer, the errors approach the machine precision.

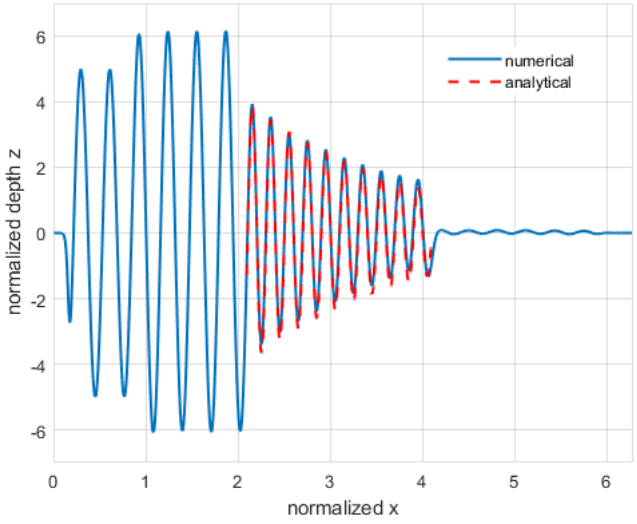
Second, the wave maker simulation is validated against the linear analytical solution presented by Alam [2], according to which the surface elevation over the visco-elastic bottom is given by:

$$\eta_s = (a_{s0}e^{\omega_i t})e^{i(kx-\omega_r t)} = a_s(t)e^{i(kx-\omega_r t)} \tag{3.7}$$

In equation 3.7, the first exponential factor associated with the imaginary frequency represents the damping effect, and the second exponential factor with the real frequency regulates the harmonic effect. Coupling together, they govern the damping harmonic oscillation of surface elevation over time.



(a) Visualization of a wave maker with a rectangular wave carpet. Surfaces waves are propagating and damping out over the bottom wave carpet.



(b) Visual comparison of numerical versus analytical surface elevation results. Numerical surface waves decay over the wave carpet with the similar pattern to that of the analytical solution.

Figure 3.2: Visualization of waves propagating over a rectangular wave carpet and qualitative comparison of the wave maker results with the analytical solution. Waves are damping out over the carpet and the damping wave profile matches pretty well with that of the analytical solution. Some minor deviations come from the approximation of the wave maker algorithm that make some part of seabed nearly rigid.

In the spatial domain the motion of a water particle over time is simulated by keeping position x constant and progressing time t forward. This actually visualizes the analytical results in order to qualitatively validate against the numerical ones. The latter visualizes how the numerical surface waves evolve over the visco-elastic seabed. As a result, the matching between two is illustrated in Figure 3.2b.

In Figure 3.2b, the numerical result does not perfectly match the analytical solution because of the minor numerical instability coming from two sources: the very high stiffness of bottom outside the carpet area and the abrupt change of stiffness between the sea bottom and the mimicking wave carpet areas. This minor instability reduces the numerical precision but it is not too significant for the numerical scheme to be used for shape optimization purpose.

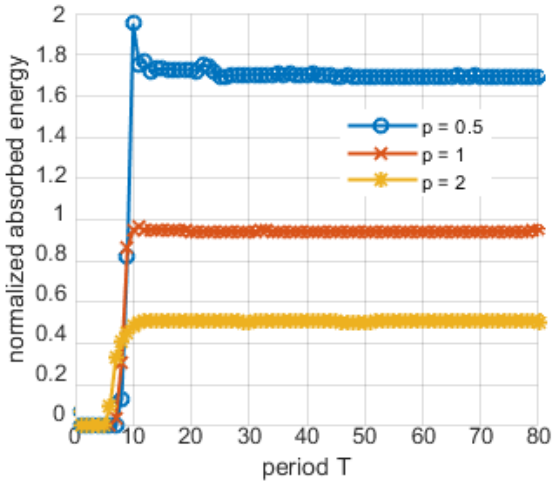
3.2 Stability

The stability test is conducted to find the stable time point of simulations. When a simulation reaches its stable point, the wave carpet absorbs relatively the same amount of energy in every period. The recorded energy absorption should be the average of absorbed energy amounts over a couple of past periods, say 3 previous periods, after the stable moment. Accordingly, the simulation length should be at least as long as 3 periods beyond the stable time point.

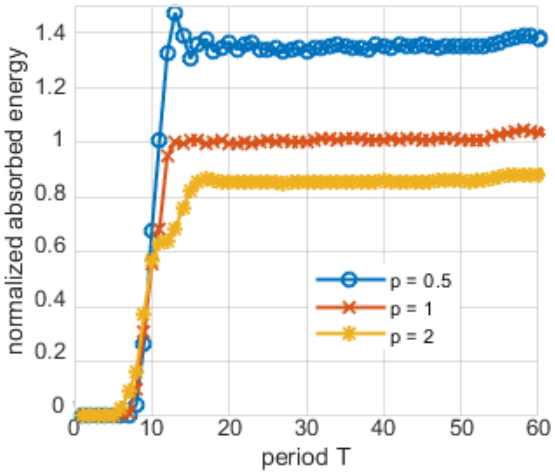
The stable test is carried out by letting numerical waves gradually propagate over the carpet until the estimated energy absorption in each period remains unchanged. The test should also include some typical carpet shapes such as a circular shape and extreme shapes, such as the longest and widest elliptical shapes. Doing this ensures that the simulation time is sufficiently long to cover all of the possible shapes in further optimization simulations.

Figure 5.2 presents the stability test for monochromatic unidirectional incoming wave case. It shows that the absorbed energy is nearly zero initially, then rising when the incoming waves hit the carpet, finally reach a plateau and relatively stay stable at period 14 for all three shapes. Thus, the absorbed energy averaged at period 14, 15, 16 or later should give the relatively accurate absorbed energy rate for the given wave case.

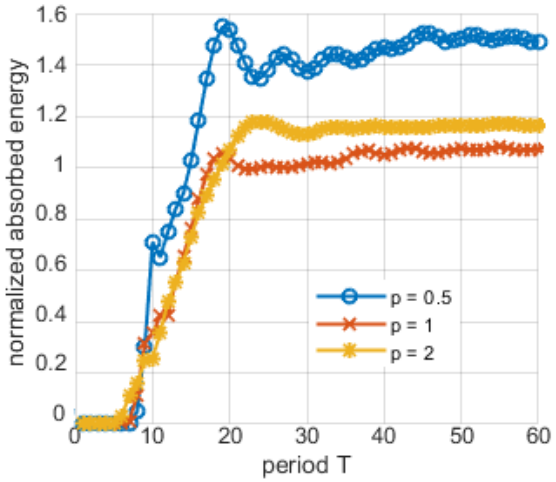
Unfortunately, the stable point is not the same for different incoming wave cases. It is necessary to conduct the stability test in each case for its stable time points. Figure 5.2 presents the stability test for three other different wave cases: Single-frequency unidirectional, single-frequency multi-directional and multi-frequency multi-directional. Unsurprisingly, more complicated cases have later stable time points. These wave cases are stable at $T = 12, 18$ and 28, respectively. As a result, the simulation lengths of these cases should



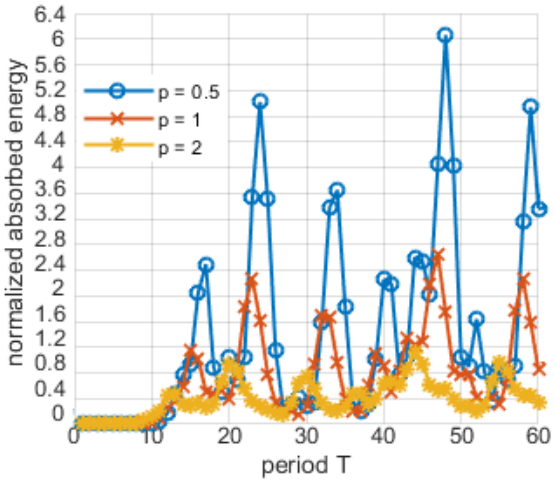
(a) Wave case I - Single-frequency unidirectional. The stable state forms after 12 periods. The simulation timespan should be 14 periods or more.



(b) Wave case IIa - Single-frequency multidirectional. Incoming waves are at angles of $0, -37^o, 37^o$ from y direction. The stable state forms after 18 periods. The simulation timespan should be 20 periods or more.



(c) Wave case IIb - Single-frequency multidirectional. Incoming waves are at angles of $0, -53^o, 53^o$ from y direction. The stable state forms after 28 periods. The simulation timespan should be 30 periods or more.



(d) Wave case III - Multi-frequency unidirectional. Incoming waves constitute of waves with $k = 10, 24, 27, 29, 30$. There is no stable time point. The simulation timespan should be at least 30 periods or more. The absorbed energy is averaged over periods 14-20.

Figure 3.3: Stability analysis of the wave maker simulations in different wave cases in which the stable time point varies accordingly. The stable time point must be after the incoming waves pass over the wave carpet. More complicated wave cases have later stable time points. Particularly, multi-frequency wave cases do not have ones because of irregular waves.

be at least 14, 20 and 30, respectively.

3.3 Energy Conservation

It is critical to verify the energy conservation in the closed simulation domain to ensures the numerical correctness. Furthermore, the upcoming optimizations are energy-based, the test makes the energy calculations more reliable. The simulation must abide by the energy conservation law, according to which the absorbed energy must match the energy reduction before and after waves pass the wave carpet. The damping mass-spring model has the absorbed energy given by:

$$E_{absorbed} = \frac{1}{T}\int_0^T b^\star \phi_z^2 dt \tag{3.8}$$

The energy flux passing a cross section, and thereby the energy reduction before and after the wave carpet, is given by:

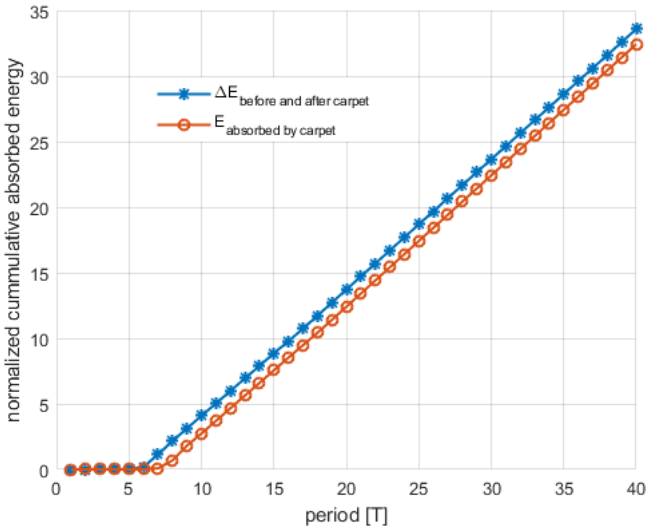
$$E_{flux} = -\rho \int_{-h}^{\eta} \phi_t \phi_x dz \tag{3.9}$$

$$\Delta E = \frac{1}{T}\int_0^T (E_1 - E_2) dt \tag{3.10}$$

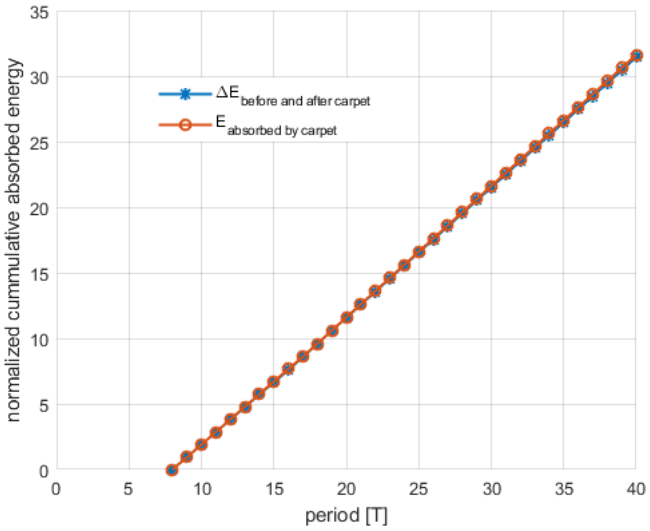
where, E_1 and E_2 are the energy fluxes at the cross sections just before and after the wave carpet. The derivation of energy flux equation is detailed in Appendix A.

The energy conservation law ensures $E_{absorbed} = \Delta E$. Figure 3.4 shows the matching starts from period 8. The abscissa is the period number, the coordinate is the normalized energy values. It can be seen in Figure 3.4 that $E_{absorbed}$ and ΔE match very well from period 8 throughout the simulation of up to 40 periods. The match test should be conducted from period 8 because it takes some time for waves to propagate over the carpet.

Figure 3.4 compares these two amounts of energy over n periods after the wave maker simulations reach the stable state. The good match of two quantities verifies the universal law and can serve as another indication for the correctness of the numerical scheme.



(a) Starting from period 0, the pattern of the energy reduction before and after the wave carpet is similar to that of the energy absorption by the wave carpet. The offset comes from the fact that the energy reduction is recorded earlier when incoming waves just enter the wave carpet region than the energy absorption is recorded when the waves are passing over the wave carpet.



(b) Starting from period 8, the energy reduction before and after the wave carpet matches very well with the energy absorption by the wave carpet in every period. There is no lag in recording time between these two quantities.

Figure 3.4: Conservation of energy in the simulation domain. The energy reduction before and after the wave carpet (blue line) must match with the energy absorption by the wave carpet in every period.

3.4 Convergence

One of the important criteria to ensure the numerical correctness is convergence. Unlike the stability that is dependent on wave cases, the convergence depends on the discretization. The convergence test is conducted in order to ensure that the temporal and spatial discretization is sufficiently fine for the converging results. The temporal discretization is represented by the number of time steps per period, and the spatial one is by the number of nodes in x or y coordinate. For Runga-Kutta numerical scheme, the former is required to be at least at the threshold of 256 time steps per period. Any number below the threshold leads to the numerical instability, also known as numerical blowup, while any number beyond that yields pretty similar results with insignificant improvement of accuracy.

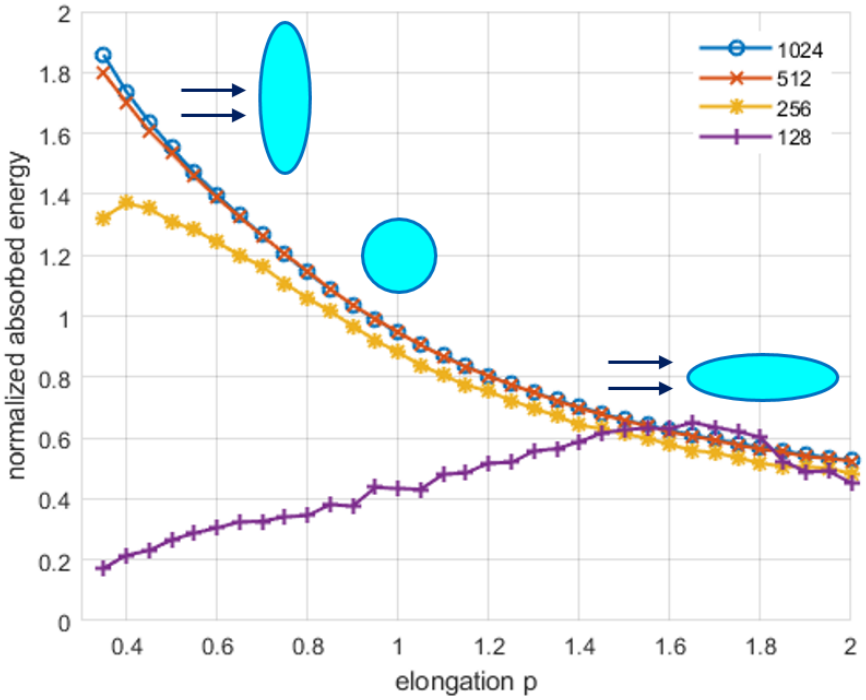


Figure 3.5: Convergence analysis of the wave maker simulation. The test investigates a range of elliptical-shaped carpets with varying aspect ratios, called elongation p, in four different spatial discretization schemes. The larger the elongation p is, the more stretched the wave carpet is along the waves direction. The simulation is not converging at mesh size 128 and 256, but starting to be at mesh size 512. Thus, the minimum number of Fourier modes is 512 in each x- and y-direction.

The spatial discretization, on the other hand, has a considerable influence on the convergence. Figure 3.5 shows the convergence test for the monochromatic unidirectional incoming wave. The test employs the elliptical carpets with the axis ratios ranging from 0.5 to 2. At the spatial discretization of 512×512, the absorbed energy converges. The further increase of spatial discretization fineness does not improve the results significantly, but it quadruples the computing costs. For the balance point, the spatial discretization is selected at 512×512 nodes.

Part II

Methods

Chapter 4

Optimization Methods and Tools

4.1 Overview

Optimization plays an important role in various engineering fields. Its research has been extensively conducted over centuries [18, 39, 56, 58]. At application level, our ultimate goal is to optimize the wave carpet shape. Before really diving into solving our optimization problem, we investigate different approaches to general optimization and see how they fits the problem. We first introduce the optimization problem to be solved, then concentrating on the attempted methods that effectively solve the problem to different levels of satisfaction.

Optimization Problem

The optimization problem involves the planar geometric shape of the wave carpet. Its suitable representation is thus considered in order to form the appropriate input for the objective function. The carpet shape is first represented in polar coordinates, then transformed into the x-y coordinates to be embedded in the mathematical model in Equations 2.2 and 2.3.

Applying Fourier transform to the polar shape and truncating the high-frequency harmonic terms give the general Fourier representation of the shape as in Equation 4.1. From the general form, phases are fixed to $\pi/2$ to derive the x-axis symmetric shapes for the symmetric wave cases. An elongation factor p is then introduced, enabling the shape to stretch in x- or y-dimension while maintaining its constant area. Finally, the polar coordinates are converted to the Cartesian coordinates as in Equation 4.3. For the asymmetric wave cases involving the asymmetric shapes, a rotation angle β is introduced with one more transformation step as in Equation 4.4.

$$r(\theta) = r_0 + \sum_{n=0}^{N_F} a_n \sin(n\theta + \phi_n) \qquad \text{general} \tag{4.1}$$

$$r(\theta) = r_0 + \sum_{n=0}^{N_F} a_n \sin(n\theta + \frac{\pi}{2}) \qquad \text{x-axis symmetric} \tag{4.2}$$

$$x = pr\cos(\theta), \qquad y = \frac{1}{p} r \sin(\theta) \tag{4.3}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.4}$$

subject to:

$$-\frac{2r_0}{N_F} \leq a_n \leq \frac{2r_0}{N_F} \tag{4.5}$$

$$p \in [0.5, 2] \tag{4.6}$$

$$\phi_n, \beta \in [0, 2\pi] \tag{4.7}$$

$$A = const \tag{4.8}$$

The polar coordinates facilitate the shape evolution during the optimizing process as the shape is represented by a few parameters that can be easily controlled. The set of parameters (a_n, ϕ_n, β, p) forms the input for the objective function. Picking 5 first Fourier harmonic terms results in 6 parameters for symmetric shapes (without ϕ_n and β), or 12 parameters for asymmetric shapes. The base circle radius r_0 in the Fourier representations, initially set to 0.5, is used for adjusting the carpet area to the constant value. In order to avoid complicated shapes and invalid carpets with loop, occurring when $r(\theta) < 0$ at some θ, the constraints on harmonic amplitudes are set based on the base circle radius as in Equation 4.5. These constraints are loose and cannot guarantee the positiveness of the polar radius $r(\theta)$ for all θ but helping get rid of a large number of loop carpets. A few invalid ones will be checked and eliminated at running time. To ensure the entire shape to stay within the computational region, the elongation factor is limited between 0.5 and 2. The most relaxing parameters are the harmonic phases and rotation angle that take their normal range from 0 to 2π.

In short, the optimization problem is to optimize the wave carpet shape for maximum energy absorption based on the HOS solver objective function. This core function takes the shape parameter set (a_n, ϕ_n, β, p) as input subject to the constant area constraint and the constraints on their domains. The HOS solver objective function and its adaptation to addressing the first constraint is illustrated by the block diagrams in Figure 4.1. The

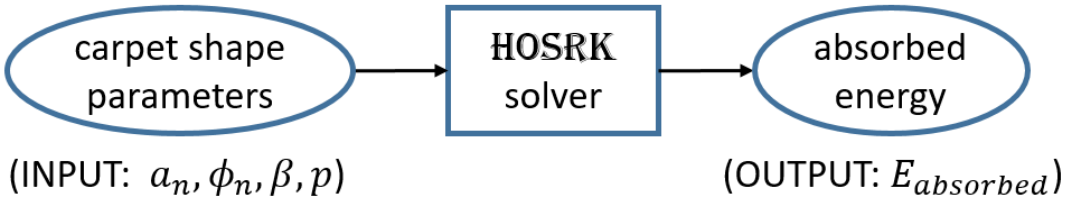
optimization problem is presented as follows:

$$\min_{a_n,\phi_n,\beta,p} HOSRK([a_n,\phi_n,\beta,p])$$

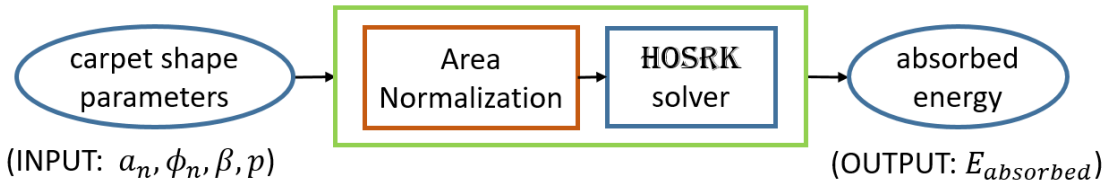
subject to:

$$\mathrm{A} = const$$

a_n, ϕ_n, β, p are bounded as in equations 4.5, 4.6 and 4.7



(a) The wave energy converter "wave carpet" is mathematically modelled as the HOSRK solver.



(b) The HOSRK solver is wrapped together with the area normalization into a new solver block to incorporate the nonlinear area constraint.

Figure 4.1: HOSRK solver block diagram and its modification to incorporate the nonlinear area constraint. They solves for the absorb energy given the the wave carpet shape parameters as input. These solvers serve as the objective function for the optimization problems.

Challenges and Approach

The optimization has a simple form: given a numerical function, find the optimal point. It is, however, very computationally expensive because it solves the problem in 3 dimensions: x-y plane and time. In specific, it takes roughly 1 hour to run a carpet instance in the simplest linear wave case and roughly 3 hours for the simplest nonlinear one. Other complicated wave

cases take even longer time. Consequently, some well-known optimization algorithms such as Gradient Descent [83, 21], Newton's Method [53, 76] cannot be applied as they require a large number of function evaluations in sequence. The second challenge comes from the nonlinear equality constraint of the constant carpet area. One of the most popular approaches to deal with nonlinear constrains is the Lagrange Multiplier method [12, 22, 62], but it is not applicable in this problem because of the long running time. Instead, we find another way to incorporate this constraint into the objective function to further simplify the problem.

High performance computing is found extremely useful in reducing the computing time. Two major sources are National Energy Research Scientific Computing (NERSC) and Berkeley Research Computing (BRC) that allow to run the application in parallel. This helps evaluate a large amount of carpet shapes in the time amount of evaluating one instance. This enables the application of highly parallel algorithms such as Monte Carlo search or parallel iteration-based methods, to name a few. Additionally, it allows the generation of data in batch, and based on the generated data, a neural network could be built and trained for the subsequent optimization. The latter method uses the alternative neural network that approximates the objective function so that the neural network itself can be used at a much cheaper cost compared to using the original objective function [66, 43].

Regarding the nonlinear equality constraint, a wrapper function is introduced to wrap around the nonlinear constraint and the objective function HOSRK solver. This wrapper first implements the area normalization, normalizing the areas of all input carpets into the standardized one before feeding them into the HOSRK solver, illustrated in the block diagram in Figure 4.1b.

Attempted Methods

The optimization problem could not be solved by the classic methods due to its complexity and the lack of analytical objective function. In addition, the non-convexity of the objective function makes the choice of appropriate optimization methods more challenging. As a result, the iterative methods such as Newton's method, Gradient Descent, etc. are not applicable. On one hand, these methods only work well with the convex objective function. On the other hand, they require many evaluations of the objective function until the solution converges, which is impractical for the HOSRK solver that takes hours for a single running.

The heuristic optimization methods are, therefore, considered to solve the optimization problem. These methods do not guarantee the finding of the optimal solutions but the nearly optimal solutions to the complicated optimization problems. Because of little understanding on the numeric HOSRK solver, the heuristic approach is the appropriate choice. Four methods are attempted, including the Response Surface Methodology (RSM), the Random Search (RS) method, the Cross-Entropy Method (CEM), Neural Network-based optimization method (NN) and the Genetic Algorithm (GA). RSM does not yield the correct result

other than that of the simplest wave case. RS and CEM solve for the sub-optimal solutions although the latter is very close to the optimal ones. NN and GA produce the optimal results as they are applied for linear and nonlinear problems, respectively.

4.2 Random Search Method

The Random Search method, also known as the Monte Carlo method, is based on the random sampling. It solves a deterministic problem in a probabilistic way. Without the need of estimating the gradient, the method evaluates the function at random data points, then aggregating the output. Thus, the result is not guaranteed to be optimal but expected to be much better than the average. How better the result is depends on the sample size. Despite its sub-optimality, the method gains its popularity due to its simplicity and broad scope of application.

The procedure of Random Search consists of three steps:

1. **Generation**: Generate inputs randomly from a probability distribution over the domain satisfying the optimization constraints.
2. **Evaluation**: Evaluate the deterministic objective function at the randomly generated inputs.
3. **Aggregation**: Aggregate the outputs, particularly max out the results.

The Random Search method was first attempted when the deterministic methods fail. In its application to the optimization problem, we follow exactly the above procedure with the aid of parallel computing. The inputs are randomly generated from a uniform distribution, instead of a popular normal distribution, due to the lack of understanding in the objective function. The use of parallel computing to evaluate the objective function at the cloud of inputs significantly reduces the implementation time. The aggregation of the results is so straightforward that the only $argmax$ function is applied to the input-output set to determine the optimal carpet shape [81, 7, 103].

Not only does the Random Search yield provisional results, but it also produces a large number of data points out of the objective functions. This serves two purposes: Help form an idea of the objective function, and; Provide a database for other heuristic methods. Particularly, the Response Surface Methodology and Neural Network-based optimization method take advantage of this database for the optimization purpose.

4.3 Response Surface Methodology

The Response Surface Method is widely used for solving the experimental problems. It establishes the relationship between the independent variables and the response variables. This relationship is commonly expressed in the form of a polynomial empirical model whose coefficients are normally determined via regression [20]. In our optimization problem, the explanatory variables are the carpet shape parameters, and the response variable is the absorbed energy. The polynomial empirical model is subsequently optimized to achieve the optimal results. Since any polynomial is derivable, the optimal results can be achieved via traditional methods, particularly stochastic gradient descent in our solution. The obtained model is, however, not the exact objective function but its approximator. It is necessary to verify the optimal results using the exact one, the HOSRK solver in our problem. Due to the approximating nature, the achieved results are not the truly optimal solutions but the approximately optimal, or heuristic, ones.

The Response Surface Method follows the below procedure:

1. Design the experiments. Two main designs are Box-Behnken design (BBD) and central composite design (CCD) [8].
2. Run the experiments. These can be real-world physical experiments or simulated computational ones.
3. Collect the outputs and set up the polynomial empirical model.
4. Solve for the polynomial coefficients using the least squares method.

Among the above steps, the design of experiments plays an important role in ensuring the correctness of the mathematical model. As the target is to fit the mathematical model to the response surface given the independent inputs, the experiments should be designed in such a way that the structure of response surface is best exposed and its construction is highly accurate. Box-Behnken design and central composite design are commonly selected as they are practical with a small number of experiments in constructing a approximate quadratic surface. The former is usually considered to be more powerful than the latter and other designs as its number of runs is smaller. One caveat is that Box-Behnken design does not cover the boundary cases [31, 93].

After the design of experiments, the objective function takes as inputs the selected set of independent variables and outputs the responses. The tuples of inputs and outputs form the data set for the mathematical model. The quadratic and cubic polynomial models are commonly used and their coefficients can be achieved using the least squares method. The polynomial empirical model is the approximate representation to the response surface. As a result, the output obtained by evaluating the model on every input should be verified by

the true objective function for the accurate response.

In the application of the Response Surface Method to the wave carpet problem, the experiments are designed using Box-Behnken method. Parallel computing is applied to evaluating the objective function, so the number of evaluations per running is far more than the required number of evaluations by the Box-Behnken design. Thus, the extra inputs are randomly generated. The quadratic empirical model is given in the form [60, 20]:

$$y(x_1, x_2, \ldots, x_n) = \alpha + \sum \beta_i x_i + \sum \gamma_{ij} x_i x_j + \sum \delta_i x_i^2 \tag{4.9}$$

where, x_i's are the independent variables, y - the response, $\alpha_i, \beta_i, \gamma_{ij}, \delta_i$ - the polynomial coefficients obtained by the least squares method.

In the matrix form, the Equation 4.9 is rewritten into:

$$\mathbf{y}(\mathbf{x}) = \mathbf{w}\mathbf{x} \tag{4.10}$$

where,

$$\mathbf{y} = \begin{bmatrix} y \end{bmatrix} \quad - \quad \text{the scalar response in the wave carpet problem} \tag{4.11}$$

$$\mathbf{w} = \begin{bmatrix} \alpha & \beta_1 & \ldots & \beta_n & \gamma_{12} & \ldots & \gamma_{(n-1)n} & \delta_1 & \ldots & \delta_n \end{bmatrix}$$
$$- \quad \text{the vector of polynomial coefficients} \tag{4.12}$$

$$\mathbf{x} = \begin{bmatrix} 1 & x_1 & \ldots & x_n & x_1x_2 & \ldots & x_{n-1}x_n & x_1^2 & \ldots & x_n^2 \end{bmatrix}^T$$
$$- \quad \text{the matrix of independent variables} \tag{4.13}$$

The inputs and outputs of the experiments form the data set utilized for the least squares method to determine the coefficients of the polynomial model. Assuming that there are m experiments, corresponding to the set of m inputs and outputs, the matrices of inputs and outputs are given by, respectively:

$$\mathbf{Y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \tag{4.14}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \ldots & x_{n,1} & x_{1,1}x_{2,1} & \ldots & x_{n-1,1}x_{n,1} & x_{1,1}^2 & \ldots & x_{n,1}^2 \\ 1 & x_{1,2} & \ldots & x_{n,2} & x_{1,2}x_{2,2} & \ldots & x_{n-1,2}x_{n,2} & x_{1,2}^2 & \ldots & x_{n,2}^2 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & x_{1,m} & \ldots & x_{n,m} & x_{1,m}x_{2,m} & \ldots & x_{n-1,m}x_{n,m} & x_{1,m}^2 & \ldots & x_{n,m}^2 \end{bmatrix}^T \tag{4.15}$$

The least squares solution is given by [41]:

$$\mathbf{w} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y} \tag{4.16}$$

4.4 Batch Gradient Descent

Gradient Descent is an iterative derivative-based method for solving the convex optimization problem. If the objective function is differentiable, its gradient indicates the steepness. Starting from an arbitrary point, walking along or opposite the steep direction leads to the local maximum or minimum, respectively. If the objective function is convex, the achieved result is guaranteed to be globally optimal. In order to adjust the step size so that the walking does not pass the optimal point, a step factor α is introduced coupling with the gradient, making the walking step $\alpha\Delta\mathbf{f}$. In common practice, the step factor is initially set large to speed up the convergence, then gradually reduced when it approaches the optimal point to ensure the convergence. The Gradient Descent update to find the minimum is simply given as follows:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha\Delta\mathbf{f} \tag{4.17}$$

where, $\mathbf{x}_0$ is an arbitrary exploring point, $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$ are the exploring points or the previous and this iterations, respectively, α - the step factor as discussed above, $\mathbf{f}$ - the objective function. In contrast, the Gradient Ascent update to find the maximum takes the positive walking step [6, 47].

Gradient Descent fails to find the optimum in the non-convex optimization problems which potentially have multiple extrema and saddle points. If the objective function is not convex, the process may converge to a local extremum or a saddle point around which the exploring point is initiated. To minimize this chance, the arbitrary exploring points should be generated in batch so that they spread over the objective function surface and are around as many potential extrema as possible. The Gradient Descent procedure is independently implemented on these initial points until convergence. To expedite the speed, the process should be run in parallel. This is particularly helpful if the objective function has a short running time so that the non-parallelizable iterative part is not a bottleneck. The output are subsequently aggregated for the final results. As based on the random sampling, the method is heuristic and does not guarantee the true optimal results. The uncertainty comes from the lack knowledge about the optimality of the non-convex objective function. The results are, however, ensured to be at least as good as that of the Gradient Descent. In the scope of the project, this method is called Batch Gradient Descent [25, 39].

In the HOSRK optimization problem, the direct application of either Gradient Descent or Batch Gradient Descent is not feasible as the objective function non-convex and it is the bottleneck in terms of running time. On one hand, the HOS solver is not easily differentiable. It is a numerical solution, so its derivative must be taken numerically. This requires a couple of its evaluations that at least doubles the running time and computing workloads. On the other hand, Gradient Descent is a sequential method in which the iterations are sequentially dependent on the previous ones. With little knowledge of the non-convex func-

tion and its uncertain convergence to the optimal point, that may require several iterations leading to the running time exceeding the limits. To overcome these obstacles, one approach is to eliminate the bottleneck by approximating the expensive objective function HOS solver by a easily handled cheap function. The Gradient Descent or Batch Gradient Descent is then implemented on this approximating objective function. Among the presented methods, Response Surface Methodology and Neural Network-based method follow this approach in combination with Batch Gradient Descent and Gradient Descent, respectively. The advance feature of Batch Gradient Descent over Random Search method is that from a randomly generated initial data set the candidates are refined over iterations by being progressing towards the extrema in batch.

4.5 Cross Entropy Method

Cross Entropy Method is an optimization approach based on random sampling, efficient for solving both discrete and continuous problems [15]. Loosely speaking, Cross Entropy Method is an iterative improvement of Random Search Method in such a way that it repeats the random search several time, each iteration comes up with a better sampling. Thus, Cross Entropy Method involves some sequentiality that is a trade-off for the improvement of the result thanks to the several iterations of sampling compared to a single sampling in Random Search Method.

Cross Entropy Method is commonly used in machine learning, particularly in investigating the optimal hyper-parameters [5]. In our optimization problem, we do grid search for the optimal hyper-parameters and directly apply the Cross Entropy Method for optimizing the absorbed energy as it is flexible and requires little knowledge about the objective function. The number of iterations is proportional to the amount of time required for solving the problem.

Cross Entropy Method finds an approximate optimal sampling by iterating two steps: Generate a sample of inputs from a probability distribution, and; Update the parameters of the probability distribution to produce a better sample in the next iteration. The second step involves minimizing the entropy of the probability distribution and the new distribution so that the samples converge to the optimal value. The detailed procedure is outlined as follows [13, 82]:

1. Randomly sample candidates from a probability distribution.
2. Run the objective function on the samples and collect the results.
3. Pick n% top candidates and estimate the statistics.

4. Update the probability distribution with the statistics, then repeat step 1 with the new distribution.

The first step assumes a probability distribution with the parameter set $\mathbf{P}$. The commonly used one is the normal distribution $\mathcal{N}(\mu, \sigma^2)$ with $\mathbf{P} = \{\mu, \sigma\}$ constructed in such a way that μ and σ fit the constrained ranges of independent variables. A random sample of inputs is then generated and fed into the objective function for the responses, among which 10% top candidates are selected. As the normal distribution is chosen, its parameters can be easily estimated and updated for the next iteration of sampling. That the new distribution is built on the top 10% candidates effectively moves the sampling region towards the optimal point and minimizes the distribution entropy. As a result, the sampling over iterations gets more compact and closer to the optimal solution. At the last iteration, the final result can be achieved by aggregating the top 10% candidates of the last sample.

Cross Entropy Method does not guarantee the absolutely optimal solution, but approximately, similar to that of other random sampling-based methods. The random process, however, tends to move the sampling region towards the optimal point over iterations. The number 10% is a hyper-parameter that should be tuned by trial and error. Some probability distribution parameters can be treated as hyper-parameters. The standard deviation σ in the normal probability, for instance, can be tuned or fitted to the top sample. The hyper-parameters decide the convergence speed of the procedure.

The application of Cross Entropy Method to the wave carpet optimization problem follows the aforementioned procedure. The first sampling comes from the normal distribution with parameters covering the computational domain. The subsequent samplings are based on the iteratively updated normal distribution with mean and standard deviation of the top candidates. The objective function evaluation on the samples is implemented in parallel for each iteration, which significantly reduce the procedure runtime. The hyper-parameter 10% is used for picking up the top candidates.

4.6 Genetic Algorithm

Genetic algorithm is a global optimization technique of the evolutionary algorithm family [92, 35, 102]. It reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. The algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects elite individuals from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. At a high abstract level, the algorithm is outlined in the block diagram (see Figure 4.2).

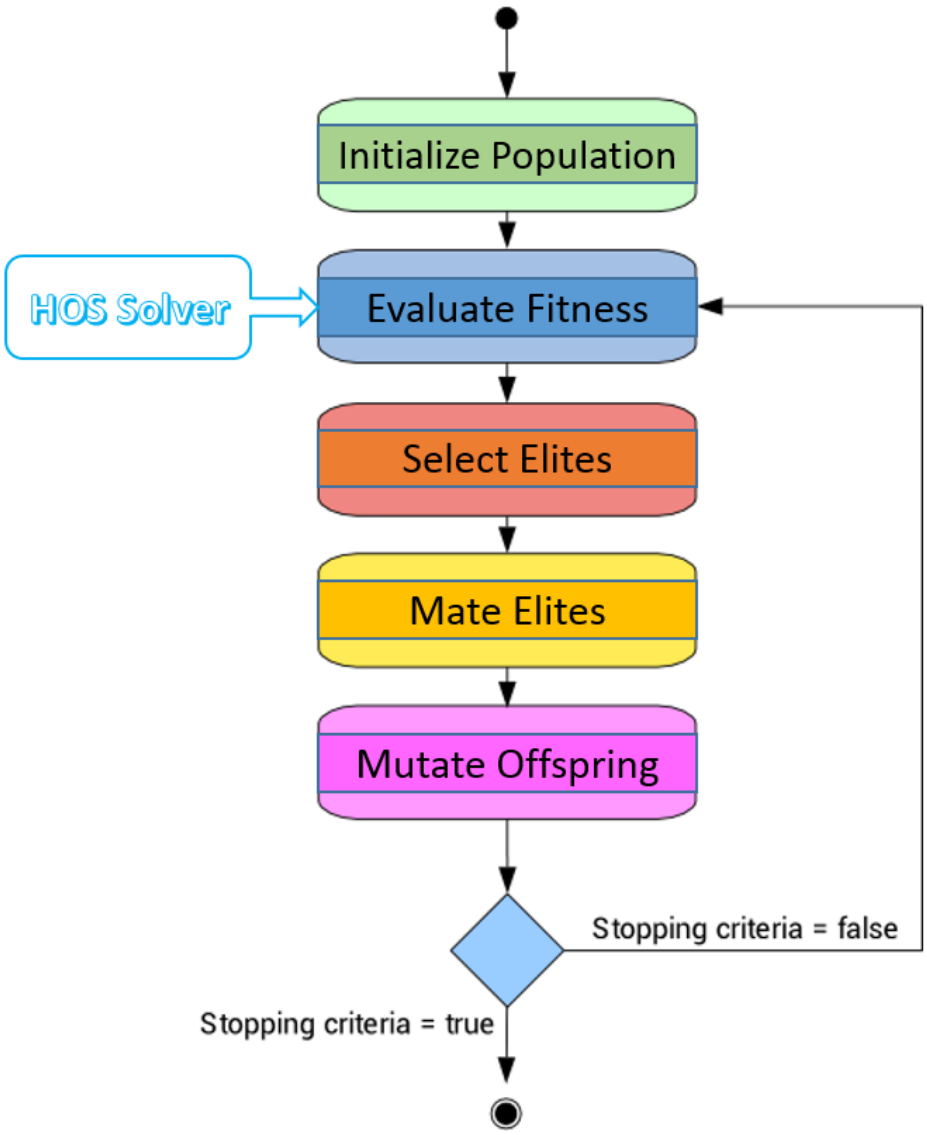


Figure 4.2: Genetic algorithm block diagram. The algorithm consists of 5 steps and a loop. A swarm of individuals is first initialized and they are evaluated for their fitness using the HOSRK solver. The elites are then selected and mated for potentially better newborn individuals. In order to avoid the convergence to a local optimum, some randomly selected newborns are mutated. The new generation is subsequently evaluated. This procedure reiterates until the stopping criteria are matched.

A typical genetic algorithm procedure consists of 5 steps: 1. Initializing population; 2. Evaluating fitness; 3. Selecting elites; 4. Mating elites; 5. Mutating offsprings. Most steps have many options. By trial and error, the best option is selected for each step [30, 71, 75].

1. **Initialize population:** In our problem, each carpet shape is one individual. Each individual has a chromosome consisting of genes. Each parameter among $r_0, a_{1-N_c}, \phi_{1-N_c}$ or p is a gene. Thus, initilizing population is to randomly generate N_p sets of parameters, where N_p is the number of individuals in the population. Here we select $N_p = 288$ for the parallelism purpose, discussed later, and the random generation is Gaussian.

2. **Evaluating fitness:** This is the most strainghtforward step for our problem. Fitness hereby means the amount of energy absorbed by each wave carpet (or individual). The fitness of each individual is evaluated by using the HOSRK solver on its chromosome (or wave carpet parameters). The evaluations are parallelized since the individuals are independent of each other. This feature is one of the most advantages of the genetic algorithm over other optimization methods.

3. **Selecting elites**: After being evaluated, each individual is dubbed a value indicating how elite it is in the population. We want to select the most elite individual to mate them in order to produce the offsprings for the next generation. We consider two methods: Selecting Best and Selecting Tournament. The former involves selecting the k best individuals while the latter groups the individuals into roughly k groups and selects the best in each group. Surprisingly, testing both methods on the polynomial approximation of the HOSRK solver indicates that Selecting Tournament results in a better solution than Selecting Best. Thus, we pick Selecting Tournament option for this step.

4. **Mating elites:** Given two elite individuals, we mate them and produce two offsprings. We consider two mating ways: Cross Uniform and Cross Blend. The former loops through each gene of the individual and decides to swap them based on a probability p. The latter takes the average of each gene and randomly adds some noise to the value. The noise range is predetermined and scaled based on the gene amplitude. After born, the offsprings are evaluated for their fitness again and placed into the population while their parents are deleted. Again, testing on the polynomial approximate function shows that Cross Uniform leads to a better result than Cross Blend. Thus, we pick Cross Uniform for this step.

5. **Mutating offspring:** In order to avoid converging to a local optimum, we randomly mutate some offsprings before moving to the next generation. Offsprings hereby include all the offsprings as the results of mating process and the individuals that were not selected to mate. Thus, we loop through all of the individuals, select them with probability p and mutate the selected ones according to $\mathcal{N}(\mu, \sigma^2)$ where μ and σ^2 are the predetermined arrays of mean and variance of the genes.

At the end of each generation, the algorithm goes to the next one or stops depending on the **Stopping Criteria**. There exist various options for Stopping Criteria. For instance, the algorithms stops when the average fitness is below a typical tolerance, or when it reaches

a certain number of generations. For the problem with high computational cost, a second approach is taken and the maximum number of generation is set to be $N_g = 40$.

It is important to note that our HOSRK solver is very computationally expensive. Typically, it takes 16 minutes for one run of linear case and 28 minutes - nonlinear case on one CPU. Thus, there is a strong need to parallel the running process. The bottleneck is at the fitness evaluation part of each generation cycle. We can evaluate the fitness in parallel on a supercomputer using SCOOP, a python-based open-source library for distributed computing across nodes. Typically for our problem, we request 12 nodes with 24 cores each, making up 288 CPU nodes. That is the reason why we picked $N_p = 288$ the number of individuals in the population. Given the problem size and the computing resources, the genetic algorithm running time is estimated to take roughly 12 hours.

4.7 Parallel Computing

Parallel computing plays a big role in the project, without which it cannot be done. The optimization problem is set up in 3-dimensional domain. As a result, it is so computationally expensive that the objective function takes at least 1-3 hrs to run on a single simple setting of wave carpet as inputs. Because the optimization procedures require a large number of evaluations of the objective function, their sequential execution is infeasible. Fortunately, the choice of heuristic methods facilitates parallel computing as these methods were designed in such a way that the execution can be implemented in batch.

Parallel computing requires a heterogeneous grid or a computer cluster, also aka a supercomputer, a distributed task module/library and a parallel computing algorithm. A supercomputer is a system of connected standalone computers, aka nodes, that provide hardware for parallel computing. A software package, commonly in form of a module or library written in a specific programming language, is required for the communication between computers. At the highest level, a parallel computing algorithm is needed to enable the parallel of computation stream at the bottle necks of solving the computational problems [74].

The project have access to two supercomputers, National Energy Research Scientific Computing (NERSC) and Berkeley Research Computing (BRC). Both of the systems have similar hardware but different configurations. The former allows the request of 512 nodes, each of 64 cores, making the capacity of executing 512×64 computations at the same time. The latter does 24 nodes with 24 cores each, allowing 24×24 parallel computations. A core is a single processing unit that can standalone run a process.

The thing that actually performs computing is a core in CPU. Nowadays most standalone/desktop computer can perform parallel computing with multi-core CPU. There are

two levels of parallel computing in terms of software: multi-threading and multi-processing. The former refers to a single running process/program that forks out multiple flows of executions. The latter relates to multiple running processes of the same kind, each performing a flow of executions. Most computing software support multi-threading computations inherently. Multi-processing is normally not that common, but needs to be programmed. A running program can be of either type, or in the combination of both type. Parallel threads or processes can perform similar or different tasks. Most commonly, one lets parallel flow implement tasks in roughly equal running time so that they can be synchronized at the end [63].

In terms of hardware, parallel computing can be done on either a standalone computer with multiple processing cores or a system of connected computers, or nodes as mentioned earlier. The former is so straight forward that the parallel can be obtained by the built-in features of the applications or the coding modules of programmings languages. The latter is a little more complicated as it involves the communications between the standalone computers. The communications can be obtained only via the third-party parallel computing modules, among which are SCOOP, Dask and MPI.

Scalable COncurrent Operations in Python, or SCOOP, is a distributed task module supporting parallel computing on a supercomputer. It was designed in such a way that minimum knowledge of parallel computing is required but a simple API mainly including a modest number of mapping functions [51]. In addition, SCOOP was intended for, but not limited to, evolution algorithms and Monte Carlo simulations, which fits perfectly with our optimization strategies. The architecture of SCOOP lies in the deployment of workers and broker, in which workers are the independent sub-processes that execute different flows of computations, and broker works as a mediator that essentially allows the communications between workers, and thus, between standalone computers. This mechanism, however, creates a bottleneck at the broker if there is a very large number of parallel sub-processes. Our experiments show that SCOOP handle well that of $O(10^2)$ workers in parallel, the order of $O(10^3)$ workers creates a serious congestion problems.

Dask is quite different from SCOOP. Dask is a Python library that was mainly designed for boosting up parallel computing by leveraging the Scientific Python Stack, of which are two commonly known library NumPy and pandas [27]. Dask achieves its goal by creating its new data structures on top of those provided by the leveraged packages and optimizing the memory usage and the scheduling. As a result, these enhances the parallel computing efficiency with higher speed, minimum memory footprint and dynamic task scheduling [80]. These significant improvements are intended for computing on both a single machine and a cluster. While it is remarkable for local computing, the effect is not much different from that of other supporting modules. Dask uses the distributed scheduler, different from the single machine scheduler supporting local computing, to enable cluster computing. The distributed scheduler centrally manages and dynamically distributes computing tasks spread over the cluster. It centralizes the coordination of several sub-processes across multiple computers

and the responses to the concurrent requests of several clients. As a result, this creates a bottleneck at the central point that does not allow a very large number of parallel computers. Our experiments show that a couple of hundred of concurrent processes is the limitation of parallel computing using Dask.

Apart from these two aforementioned modules, Message Passing Interface (MPI) is a communication protocol supporting both point-to-point and collective communications across parallel computers. In fact, MPI is a standard application interface serving multiple purposes for end users and module developers. It is also full-featured so that it can be used as a message-passing library for direct parallel computing programming [44]. The fact that MPI allows point-to-point communication makes it a real difference from several other modules, including SCOOP and Dask discussed above, as it allows parallel processes to directly communicate with each other, thereby not creating a bottleneck. MPI is, therefore, has a large potential of scaling-up. Our experiments successfully run on the scale of $O(10^5)$ parallel processes. Larger scale can be possibly achieved according to NERSC experts' advice. Originally, MPI was written in low-level programming languages, making its application more challenging than other modules. There are extension packages today in high-level programming languages eliminating this drawback, making MPI the dominant model used in high-performance computing today [26].

Different supercomputers may be incompatible with all parallel-computing modules. This is mainly due to the cluster configurations and the scaling capability of the module. NERSC and HPC BRC are the two supercomputers that we have access to. The former allows very large computing scale while the latter does smaller. Accordingly, they are configured in such a way that the centrally managing module is not suitable for the first but works well for the second. In particular, one should use MPI to scale up computing on NERSC, and the other modules on HPC BRC, taking into account the ease of programming. Indeed, MPI is widely applicable for most cases but it has a steep learning curve due to its low-level nature that make the programming job burdensome. The other modules, on the other hand, were developed on top of the low-level packages with particularly for the ease of use, thereby being more appropriate for small and medium-scale computing problem.

The final consideration for the choice of supercomputers and modules is the applicable optimization methods to solving the problem. Some modules has very good support for different algorithms, significantly reduced the engineering workload. Genetic algorithm, for instance, is the one of this kind. While MPI requires the full implementation of the algorithm, SCOOP provides the convenient interface that supports most of its features. As genetic algorithm is one of the main methods in our project, the choice of the module and its compatible machine is an important part in achieving the computing efficiency and the reliable results.

Chapter 5

Neural Network-based Optimization

5.1 Background

Artificial neural network had a long research history since 1940s. Its simplest form, a perceptron, is a binary classifier that linearly separates the patterns into two categories based on a set of weights. Its next evolution is the multi-layer neural network consisting of multiple perceptrons. It was, however, proved that the simple connection of basic perceptrons cannot simulate nonlinear processes, for instance, as simple as the xor circuit. The subsequent introduction of nonlinear activation functions solved this problem. But over a long time, the research of neural network stagnated due to the lack of computing power and an algorithm for the neural network to learn effectively. It was not until computers achieved greater processing power and the Backpropagation Algorithm [84] was devised that neural network became a useful tool. Its research then regained the popularity and its applications widely spread over various fields, dominantly in artificial intelligence and machine learning. Nowadays, its applications emerge more and more in different fields thanks to its usefulness as a universal function approximator. In the scope of the project, we investigate the potential of neural network for solving optimization problem.

A simple description of neural network is presented in Equation 5.3. Loosely speaking, the neural network can be considered a black-box function taking as input $\mathbf{x}$ and outputting $\mathbf{y}$, where $\mathbf{x}$ and $\mathbf{y}$ are the variables of our interest. Inside the black-box is a sequence of matrix operations in which the output of the previous layer serves as the input of the following layer. Among its core components are the weights and biases that act on the input to transform its dimension. The affine transformation is followed by the operation of the activation function σ. This step essentially makes the neural network nonlinear, without which a sequence of linear matrix operations can be reduced to a single linear matrix operation, and a neural network without activation is not able to simulate a nonlinear process. The combination of affine transformation and nonlinear activation completes one layer of the neural network. The next layers follow the same pattern with different sets of weights

and biases and, commonly, the same activation function. The final layer may omit the activation function as it is intended to transform the input dimension into the desired output dimension. The visualization of a neural network with multi-dimensional input and scalar output is presented in Figure 5.1.

$$\mathbf{M_{i+1}}\left(\mathbf{W_i}, \mathbf{b_i}, \mathbf{x_i}\right) = \mathbf{W_i} \cdot \mathbf{x_i} + \mathbf{b_i} \quad - \quad \text{Affine transformation at layer (i+1)-th} \tag{5.1}$$

$$\mathbf{x_{i+1}} = \sigma_{i+1}(\mathbf{M_{i+1}}) \quad - \quad \text{Nonlinear activation at layer (i+1)-th} \tag{5.2}$$

$$\mathbf{x} = \mathbf{x_0} \xrightarrow{\mathbf{x_0}} \ldots \xrightarrow{\mathbf{x_i}} \mathbf{W_i} \cdot \mathbf{x_i} + \mathbf{b_i} = \mathbf{M_{i+1}} \xrightarrow{\mathbf{M_{i+1}}} \sigma_{i+1}\left(\mathbf{M_{i+1}}\right) = \mathbf{x_{i+1}} \xrightarrow{\mathbf{x_{i+1}}} \ldots \xrightarrow{\mathbf{x_n}} \mathbf{y} = \mathbf{x_n}$$
$$- \quad \text{Chain of neural network operations} \tag{5.3}$$

For the sake of mathematical simplicity, the dimension of input and weights in each layer can be augmented in 5.4. The set of weights and biases is hereafter referred as the weights or the neural network parameters. As a result, the neural network scheme takes the form 5.6. This simplifies the differentiation with respect to weights needed for the backpropagation.

$$\overline{\mathbf{x}}_\mathbf{i} = \begin{bmatrix} \mathbf{x_i} \\ 1 \end{bmatrix} \quad , \quad \overline{\overline{\mathbf{W}}}_\mathbf{i} = \begin{bmatrix} \mathbf{W_i} & \mathbf{b_i} \end{bmatrix} \quad - \quad \text{Dimension augmentation} \tag{5.4}$$

$$\overline{\mathbf{M}}_{\mathbf{i+1}}\left(\overline{\mathbf{W}}_\mathbf{i}, \overline{\mathbf{x}}_\mathbf{i}\right) = \begin{bmatrix} \mathbf{W_i} & \mathbf{b_i} \end{bmatrix} \begin{bmatrix} \mathbf{x_i} \\ 1 \end{bmatrix} = \overline{\mathbf{W}}_\mathbf{i} \cdot \overline{\mathbf{x}}_\mathbf{i} \quad - \quad \text{Affine transformation } \overline{\mathbf{M}}_{\mathbf{i+1}} = \mathbf{M_{i+1}} \tag{5.5}$$

$$\overline{\mathbf{x}} = \overline{\mathbf{x}}_\mathbf{0} \xrightarrow{\overline{\mathbf{x}}_\mathbf{0}} \ldots \xrightarrow{\overline{\mathbf{x}}_\mathbf{i}} \overline{\mathbf{W}}_\mathbf{i} \cdot \overline{\mathbf{x}}_\mathbf{i} \xrightarrow{\overline{\mathbf{M}}_{\mathbf{i+1}}} \sigma_{i+1}\left(\overline{\mathbf{M}}_{\mathbf{i+1}}\right) \xrightarrow{\overline{\mathbf{x}}_{\mathbf{i+1}}} \ldots \xrightarrow{\overline{\mathbf{x}}_\mathbf{n}} \overline{\mathbf{x}}_\mathbf{n} = \overline{\mathbf{y}} \tag{5.6}$$

The working of neural network starts with the random generation of all weights and biases. At this point, the new-born model simulates a random process that outputs random results for all input it takes in. In order for the neural network to be useful, it should produce the desired results given the valid input, verified by a true function. This goal can be achieved through a training process in which a neural network is trained on the set of collected data. The training data set is generated using the true function or collected from a true process in reality. After being trained, the neural network has its weights and biases converging to the state that can closely imitate the behavior of the true process. The training is implemented using Backpropagation algorithm and Stochastic Gradient Descent [16, 17, 61]. The former involves a forward pass and a backward pass using the chain rule on the pre-defined loss. The latter is to update the weights and biases with a fraction of the loss gradient. Figure 5.1 illustrates the procedure with forward pass and backpropagation. The details will be discussed in section 5.2.

In recent years, neural network is widely utilized as a means of implementing several AI and machine learning algorithms in many areas of science and engineering [87, 50, 97]. Despite its various applications in different fields, loosely speaking, they all come down to

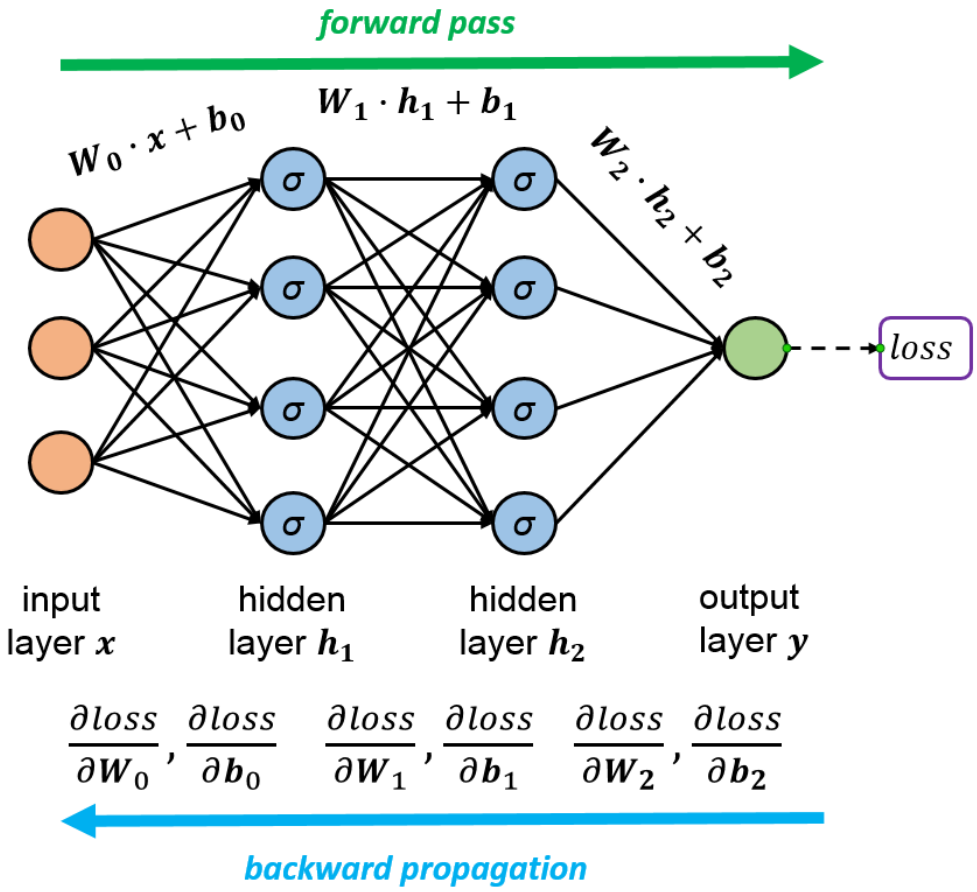


Figure 5.1: Neural network scheme and its operations. Notice that there is a component σ at each hidden node, called activation function. It plays an important role in adding the nonlinearity to the model, thereby enabling the model to approximate the nonlinear processes.

the universal approximation ability of the neural network. That is, the neural network, when appropriately built, is essentially able to approximate any function up to any level of precision given sufficient data. This is well-known as the Universal Approximation Theorem [23], and the neural network is regarded as the universal approximator. Apart from the other applications of neural network, this thesis investigates the potential use of the universal approximator for optimization purpose. The idea comes from the fact that the objective function HOS solver is too computationally expensive to be used directly, so a cheaper substitute could be useful. This leverages on the outstanding approximation ability of the neural network to approximate the true function, it is then used as the alternative objective function for optimization goal.

5.2 Training a Neural Network

Training a neural network is a process in which the neural network, starting from the new-born state with randomly generated parameters, has its parameters converging to the mature state that the neural network is able to produce the desired results given the valid input. This objective can be achieved by solving the optimization problem in which the objective function is the Euclidean distance from the desired results and the neural network output, commonly called the loss function. The loss function is to be minimized with respect to the neural network parameters. The optimization problem is defined in 5.8.

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 \quad - \quad \text{loss function} \tag{5.7}$$

$$\min_{\mathbf{W_i}, \mathbf{b_i}} L = \min_{\mathbf{W_i}, \mathbf{b_i}} \frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 \tag{5.8}$$

where, $\hat{\mathbf{y}}$ and $\mathbf{y}$ indicate the desired results, generated by the true function, and the neural network output, respectively; Euclidean distance or *norm-2* $\|\cdot\|_2$ is to squeeze the difference vector into a scalar value, square and fraction $\frac{1}{2}$ are to facilitate the derivative-estimating task; and $\mathbf{W_i}, \mathbf{b_i}$ are the weights and biases, or the neural network parameters. The optimization problem can be solved by the combined usage of backpropagation and stochastic gradient descent. The former is a computing mechanism to effectively take the gradient of the objective function with respect to the weights, and the latter is a randomized algorithm to optimize the neural network parameters.

Backpropagation consists of a forward pass and a backward pass. It leverages on the chain rule and dynamic programming to effectively perform the calculations. The forward pass is as straightforward as the neural network as presented in scheme 5.3. It computes the intermediate term $\mathbf{x_i}$ in each layer starting from the first hidden one to the loss. Backward pass iterates from the loss back to the first hidden layer to compute the loss gradient with respect to the intermediates terms and weights. As the neural network is rewritten in recursive form 5.10, the loss gradient with respect to any weight is the multiplicative chain of the consecutive differentiations of intermediate terms with respect to its previous ones. The differentiation constituents in equation 5.10 turned out to be the pieces of information that can be obtained in the forward pass as in equations 5.11 - 5.15. To avoid the repetitive computations of differentiations and matrix multiplications, dynamic programming stores all intermediate terms in each computing steps of forward and backward passes. This significantly expedites the process as the matrix operation is the algorithm bottle neck.

$$\begin{cases} \mathbf{y} = \mathbf{x_n} = \sigma_n(\mathbf{M_n}) \\ \mathbf{M_{i+1}} = \mathbf{W_i} \cdot \mathbf{x_i} \\ \mathbf{x_i} = \sigma_i(\mathbf{M_i}) \end{cases} \tag{5.9}$$

$$\frac{dL}{d\mathbf{W_i}} = \frac{dL}{d\mathbf{y}} \frac{d\mathbf{y}}{d\mathbf{M_n}} \cdots \frac{d\mathbf{M_{i+2}}}{d\mathbf{x_{i+1}}} \frac{d\mathbf{x_{i+1}}}{d\mathbf{M_{i+1}}} \frac{d\mathbf{M_{i+1}}}{d\mathbf{W_i}} \tag{5.10}$$

$$\frac{dL}{d\mathbf{y}} = \mathbf{y} - \hat{\mathbf{y}} \tag{5.11}$$

$$\frac{d\mathbf{y}}{d\mathbf{M_n}} = \frac{d\mathbf{x}_n}{d\mathbf{M_n}} = \frac{d\sigma_n}{d\mathbf{M_n}} \tag{5.12}$$

$$\frac{d\mathbf{x_{i+1}}}{d\mathbf{M_{i+1}}} = \frac{d\sigma_{i+1}}{d\mathbf{M_{i+1}}} = \sigma'_{i+1}(\mathbf{M_{i+1}}) \tag{5.13}$$

$$\frac{d\mathbf{M_{i+2}}}{d\mathbf{x_{i+1}}} = \mathbf{W_{n+1}} \tag{5.14}$$

$$\frac{d\mathbf{M_{i+1}}}{d\mathbf{W_i}} = \mathbf{x_i} \tag{5.15}$$

Along with the backward pass occurs the Stochastic Gradient Descent to update the network parameters. The training data set could be very large, so letting its whole through each pass of forward-backward could make a heavy computing workload. On the other hand, passing each data point sequentially prolongs the training process. Instead, a number of data points are randomly picked and forms a mini-batch. Passing each mini-batch through the neural network and update its parameters is the trade-off between the computational workload and the training time. This enables parallel computing as data points are independent from each other, but it raises a question on how to update the network parameters. Mini-batch Stochastic Gradient Descent is an adaptive version of Batch Gradient Descent that collectively updates the weights by the average loss gradient over the mini-batches. This is surprisingly effective in training the network in terms of speeding up and converging the process. The training procedure is presented in algorithm 1.

First, data are collected so that every data point is a pair of input $\mathbf{x_i}$ and true output $\hat{\mathbf{y}}_\mathbf{i}$. The collected data set is split up into training and testing subsets of sizes n and m, respectively. Neural network is trained on the training subset and tested on the testing subset to avoid over-fitting. The training procedure randomly picks a mini-batch of size b from the training subset, feeds it to the network and updates the weights with the fractional loss gradients averaged over the mini-batch (lines 8-11 of algorithm 1). These steps are repeated until the training subset is used up, which makes one training episode. The whole process proceeds with a number of episodes p which is the number of times the training procedure loops through the training subset. At the end of each episode or, more frequently, after a certain number of updates, the network is tested on the testing subset. As long as it gets

Algorithm 1 Neural Network Training

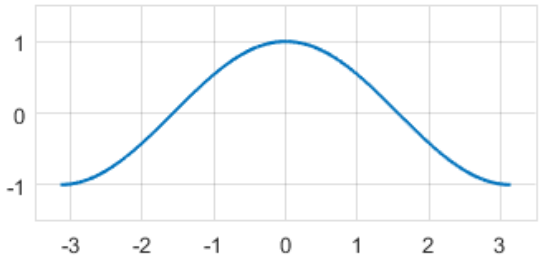
procedure NNTRAINING
  Input: Neural network at random state
  Output: Neural Network at optimal state
  Prepare a data set $\mathcal{D} = \{\mathbf{x_1}, \ldots, \mathbf{x_{m+n}}\}$.
  Split $\mathcal{D}$ into training set $\mathcal{D}_n$ and testing set $\mathcal{D}_m$ with n and m data points, respectively.
  Pick a batch size b and a number of episodes p.
  for $i = 1..p$:
    for $j = 1..\left\lceil \frac{n}{b} \right\rceil$:
      Form a mini-batch $\mathcal{B}$
      Estimate $\nabla_W L$ on $\mathcal{B}$
      Update $\mathbf{W} = \mathbf{W} - \alpha * \frac{1}{b} \sum_{k=1}^{b} \nabla_W L_k$
    Test neural network on $\mathcal{D}_m$ and save its state if L reduces.
  Restore neural network
return Neural Network at optimal state

better with reduced loss, its state are saved. The returned network has the best state over the training process.

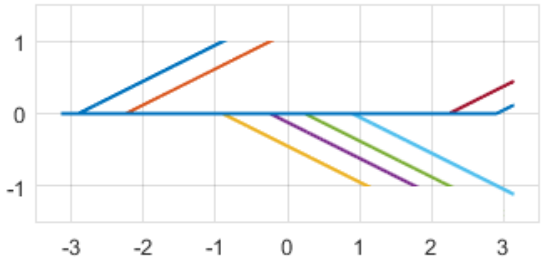
5.3 Fully-Connected Neural Network

Neural network emerges with different architectures throughout its course of evolution, among which are three fundamental types: fully-connected neural network, convolutional neural network and recurrent neural network. In chronological order, fully-connected neural network was completely built on top of perceptron in response to the need of simulating different processes; convolutional neural network was devised to solve image-relating problems; and recurrent neural network was constructed to deal with sequential data problems such as time series or natural language processing. Several other architectures and algorithms were built on top of these types or any combinations of them. Although differing in their architectures, all neural networks boil down to approximators to different processes. For the purpose of approximating a numerical function, fully-connected neural network is recommended [104, 38].

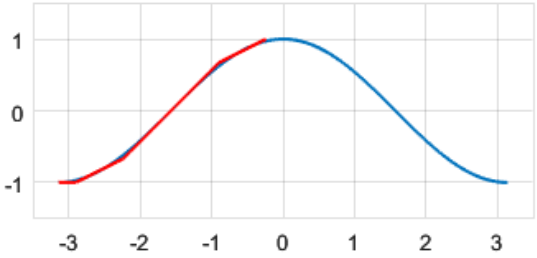
Fully connected neural network consists of a series of fully connected layers that connect every node in one layer to every node in the other layer. The number of layers determines how deep a neural network is. According to the Universal Approximation Theorem, a neural network with a single layer can approximate any function up to any precision but the number



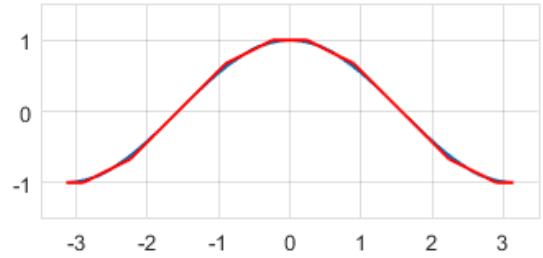
(a) Original function to be approximated. In this example, it is $y = \cos(x)$.



(b) Appropriate building blocks *ReLU*'s to be used for approximating the true function.



(c) Some building blocks are aligned to closely fit the true function.



(d) The true function is completely approximated by the building blocks whose mathematical representation is a neural network.

Figure 5.2: Demonstration of the universal approximation ability of a neural network. Starting from a function to be approximated, one can always select appropriate building blocks *ReLU*'s and arrange them closely around the true function. The mathematical representation of the fitting building blocks is a neural network.

of nodes would increase exponentially. Instead, the increase of layer numbers significantly reduces the required number of nodes to achieve the same precision. Figure 5.1 illustrates a typical structure of a fully connected neural network with two hidden layers taking as input multi-dimensional variables and outputting a scalar value [42, 1].

In order to demonstrate the approximation ability of a fully connected neural network, let us consider an example. We want to find a neural network representation of the nonlinear function 5.16:

$$\hat{y} = f(x) = \cos(x) \tag{5.16}$$

The exact function 5.16 is visualized in Figure 5.2a. Starting off from some appropriate building blocks *ReLU*'s plotted in Figure 5.2b we can construct arbitrary zigzag lines.

Carefully calibrating some lines to partially fit the exact function as in Figure 5.2c. Their mathematical forms are presented as follows:

$$h_1 = \text{ReLU}(0.5 * x + 1.45) \tag{5.17}$$
$$h_2 = \text{ReLU}(0.5 * x + 1.12) \tag{5.18}$$
$$h_3 = \text{ReLU}(0.5 * x + 0.45) \tag{5.19}$$
$$h_4 = \text{ReLU}(0.5 * x + 0.12) \tag{5.20}$$
$$h_5 = \text{ReLU}(0.5 * x - 0.12) \tag{5.21}$$
$$h_6 = \text{ReLU}(0.5 * x - 0.45) \tag{5.22}$$
$$h_7 = \text{ReLU}(0.5 * x - 1.12) \tag{5.23}$$
$$h_8 = \text{ReLU}(0.5 * x - 1.45) \tag{5.24}$$

Manually fitting them around the exact function, we come up with Figure 5.2d showing the single broken line approximating the exact function. The mathematical form is presented as follows:

$$y = h_1 + h_2 - h_3 - h_4 - h_5 - h_6 + h_7 + h_8 - 1 \tag{5.25}$$

Writing the Equations 5.17 - 5.25 in the form 5.26 we obtain a neural network with one hidden layer and an output layer:

$$x \longrightarrow \begin{bmatrix} 0.5 & 1.45 \\ 0.5 & 1.12 \\ 0.5 & 0.45 \\ 0.5 & 0.12 \\ 0.5 & -0.12 \\ 0.5 & -0.45 \\ 0.5 & -1.12 \\ 0.5 & -1.45 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = \mathbf{M_1} \longrightarrow \text{ReLU}(\mathbf{M_1}) = \mathbf{h_1} \longrightarrow \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}^T \begin{bmatrix} \mathbf{h_1} \\ 1 \end{bmatrix} = y \tag{5.26}$$

It is noticed that as a result of manual calibration the output layer is particularly simpler than the hidden layer, i.e. with unit weights and without bias, as we have to keep it fixed to easily adjust the broken lines. The complexity of all layers will be equally share if it is the result of training from a random weight generation [40].

The example demonstrates that a nonlinear function can be approximated by a neural network by just using the combination of ReLU-based building blocks aligned along the exact function. The neural network representation 5.26 is certainly not a unique approximation to the exact function. With the same number of broken lines numerous fittings can be formed, or an arbitrary number of broken lines can be selected to adjust the precision of

the approximation, leading to numerous solutions. As the number of broken lines increases, the approximation can be arbitrarily close to the exact function, and the fact is that any broken line can be transformed into ReLU's. Therefore, a neural network representation can approximate any continuous nonlinear function up to arbitrary precision.

The example is far from a complete proof of the universal approximation theorem. Indeed, it provides an intuitive understanding about how neural network can approximate a function and why it can do that so well. The demonstration does not, however, explain why stochastic training is able to converge the weights to the optimal state. This is still an active research topic and partially discussed by Jentzen and Riekert [54]. To the extent of applying the neural network theory to solving an engineering problem, delving deep into theoretical questions of neural network is unnecessary but an overview and intuitive understanding of how it works as a background knowledge.

5.4 Neural Network-based Optimization Algorithm

The typical structure of a fully connected neural network that is applied to solving our problem consists of an input layer, an output layer and a number of hidden layers in between. The network takes multi-dimensional input, the carpet shape parameters (a_n, ϕ_n, β, p), and yields scalar output, the absorbed energy amount. The neural network architecture is summarized in the table 5.1. Thanks to its good approximating property, its hyper-parameters need not be tuned carefully to obtained the good results. Here we build a neural network with two hidden layers of size 256. The activation function is the Rectifier Linear Unit, $ReLU(\mathbf{x}) = max(0, \mathbf{x})$ element-wise, which is recommended for approximation task. The input size ranges from 6-12 depending on the carpet shape configuration, i.e. symmetric shape vs asymmetric shape. The neural network is initialized randomly (newborn). It is then trained on the prepared data set until mature. The mature neural network is finally used for optimization purpose.

Table 5.1: NEURAL NETWORK ARCHITECTURE

	Input Layer **x**	Hidden Layer **h1**	Hidden Layer **h2**	Output Layer **y**
Size	6-12	256	256	1
Activation	None	ReLU	ReLU	None

The entire optimization procedure is as follows:

1. *Generating data*: Since neural network is data-hungry, data should be sufficiently generated for its training, particularly at $O(1e+4)$ data points for the carpet problem. A data point is a Gaussian randomly generated tuple (a_n, ϕ_n, β, p). The true HOS solver is then used for evaluating these data points. It would require $O(1e+4)$ computing hours for running in sequence. To reduce the running time, data are generated in parallel using NERSC supercomputer: 512 compute nodes are requested, each has 32 cores, generating $512 \times 32 = 16384$ data points in one hour.

2. *Building and training a neural network*: A neural network specified in table 5.1 is built and trained on the generated data set until its maturity. We target the relative testing error to be less than 5%. In practice, the testing error is roughly 1% after 100-epoch training, which satisfies the expected criterion.

3. *Optimizing carpet shape*: the mature neural network serves as an objective function for the upcoming optimization. The optimization algorithm is the gradient ascent applied repeatedly to a set of initial random carpet instances. The number of instances is selected beforehand at 1024. It can be much larger without considerably affecting the running time as the neural network evaluates each instance very fast. The reason for optimizing carpets from a set of instances instead of a single one is to avoid the local maxima. From a large number of starting points the chance one converges to the global maximum is accordingly high.

4. *Validating carpet shapes and selecting the best shape*: top 10% out of 1024 converging carpet shapes are selected for validation, i.e. roughly 100 carpets. Since the neural network is just an approximating function to the HOS solver. The validation of the top carpets must be carried out to obtain the true energy absorption. The accepted energy difference error between the neural network and the HOS solver is less than 5%. Among the top carpet shape the best carpet with the most true energy absorption is selected.

The entire procedure is presented in algorithm 2.

Algorithm 2 Neural Network-based Optimization

procedure NNO
  Input: HOS Solver
  Output: $(a_n, \phi_n, \beta, p)_{optim}$
  Generate a set of rand shapes $\mathcal{D} = \{(a_n, \phi_n, \beta, p)\} \sim \mathcal{N}$
  Evaluate $\mathcal{D}$ using HOS solver $\{(a_n, \phi_n, \beta, p) \mapsto E\}$
  Build a neural network NN
  Train NN on data from $\mathcal{D}$
  Generate a set of initial shapes $\mathcal{C}$
  for each carpet (a_n, ϕ_n, β, p) in $\mathcal{C}$:
    do M times or until convergence:
      $(a_n, \phi_n, \beta, p) = (a_n, \phi_n, \beta, p) + \alpha \nabla NN(a_n, \phi_n, \beta, p)$
    Evaluate $NN(a_n, \phi_n, \beta, p)$
  Select top 10% of $\mathcal{C}$ called $\mathcal{C}_0$
  for each carpet (a_n, ϕ_n, β, p) in $\mathcal{C}_0$:
    Evaluate $HOS(a_n, \phi_n, \beta, p)$
return Maximum carpet of $\mathcal{C}_0$

Part III

Results

Chapter 6

Software Specification

6.1 Existing Codebase

The research takes advantage of computational method, so most percentage of workload is coding. As a results, a complete software codebase was developed for various wave cases for both 2-D and 3-D, linear and nonlinear simulations. The coding was, however, not started from scratch. It actually reused the 2-D and 3-D rigid bottom solver developed by Professor Yuming Liu [70] and 2-D and 3-D initial wave generator by Dr. Qiuchen Guo [45, 46]. Beyond that points, all the coding was built up for solving our particular problems.

Professor Yuming Liu developed a HOS solver for waves propagating over arbitrarily topographical bottom based on HOS method by Dommermuth [34]. The solver is numerically robust and it was written in ForTran ensuring its fast computational performance. The solver consist of two files:

- `rigid_bottom_2D.f` – 2-dimensional solver for wave elevation and velocity potential of waves propagating over arbitrarily rigid topographical bottom.
- `rigid_bottom_3D.f` – 3-dimensional solver for wave elevation and velocity potential of waves propagating over arbitrarily rigid topographical bottom.

The rigid bottom HOS solver requires the initial periodic wave profile which can be generated by the wave generators. The generators are coded by Dr. Guo based on the analytical solution by Alam [2]. The original module generates 2-D waves. It is then expanded to include 3-D feature.

- `rigid_bottom_wave_generator.m` – 2-dimensional and 3-dimensional solver for wave elevation and velocity potential of waves propagating over arbitrarily topographical bottom.

6.2 Code Design

The coding is intended to effectively solve the wave carpet optimization problem with strict constraints on computing time and resources. The algorithm takes as the core module the 3-D visco-elastic bottom HOSRK solver leveraging on the 3-D rigid bottom HOS solver. The computing flow is described in section 2.3. Briefly, it is outlined in 4 steps:

- **Step 1:** Initially generate a periodic wave profile
- **Step 2:** Solve for periodic wave elevation and velocity potential.
- **Step 3:** Filter periodic waves.
- **Step 4:** Solve for incoming wave elevation and velocity potential

Among these steps, steps 2 and 4 are the bottlenecks that need the careful planning. The design philosophy targets to obtaining three objectives:

1. save the most computing resources
2. run as fast as possible
3. minimize coding workload

First, the visco-elastic bottom HOS solver is obtained by modifying the Dirichlet boundary condition on the bottom in the 3-D rigid bottom HOS solver into the Neumann boundary condition. Note that step 2 and step 4 use the same solver with different working principles. The former intakes the whole initial wave profile once and produces the output while the latter takes as input the filtered periodic waves and marching each step to the next one. In linear and regular wave cases where the input data are small, the module can read in the whole file, which save the I/O time. Otherwise, in nonlinear and irregular waves the module reads data step by step, which saves the memory. This is the trade-off between running time and memory usage.

Second, the fact that different wave cases have to read in different wave profiles hints another opportunity to further optimize the coding design. Some wave profiles are periodic or repetitive. Linear waves, for instance, repeats itself after every wavelength. Both linear and nonlinear uni-directional waves have the same profile across the y-axis. This enables that, instead of reading the whole wave profiles data, only one wavelength is read for linear waves and only a x-axis wave profile data are read for uni-directional wave. This effectively reduce the reading time and memory by up to 20 and 512 times, respectively. In the worst-case scenario of nonlinear polychromatic multi-directional waves, this strategy cannot be applied as the nonlinear waves are neither periodic nor y-axis uniform, resulting in the I/O of a huge data file of 44 GB. It shows that, without this design strategy, all the wave cases

should have read wave data files of 44 GB, which had been a big problem for the simulations.

Third, any optimization procedures requires several evaluations of the objective function. As thoroughly discussed in previous sections, the HOS solver-based objective function is very computationally expensive, leading to a bottleneck that significantly prolongs the execution time. The coding design and algorithms are selected in the way that maximizes the numbers of independent evaluations of the objective function. This enables the implementation of parallel computing as carefully discussed in section 4.7. The parallel computing, though not saving the computing resources, effectively reduces the running time by $O(10^3) - O(10^5)$ times. This ensures the second design principle.

Last, the software takes the most advantage of open sources and available modules to reduce the coding workload. Such modules are, unfortunately, written in different programming languages while the core HOSRK solvers are in another one. It is necessary to link them together under the umbrella of the integrated modules. Depending on algorithms and simulations, this can be done differently. In some cases, the core modules are converted into binary code. In some other cases, a wrapper function or a bash script is used to executed the module sequentially. It is observed that the first approach noticeably boosts the running time although it is not always feasible as it is system-dependent. Overall, this strategy significantly reduces the coding effort, ensuring the third design principle.

6.3 Complete Codebase

The entire coding work is hosted at
`https://github.com/vincentndo/UCBerkeley_PhD_Research_-_Wave_Carpet`
It consists of two parts, each of which includes several modules.

HOSRK solvers

Initial periodic wave generator:

- `viscoelastic_bottom_wave_generator.m` – 2-D and 3-D generator of periodic surface and bottom profile, generating elevation and velocity potential of surface and bottom.

Periodic HOS solver:

- `viscoelastic_bottom_periodic_2D.f` – 2-D solver of periodic waves over a viscoelastic bottom for elevation and velocity potential over time.
- `viscoelastic_bottom_periodic_3D.f` – 3-D solver of periodic waves over a viscoelastic bottom for elevation and velocity potential over time.

Incoming wave processor:

- `viscoelastic_bottom_incoming_3D.f` – 2-D and 3-D processor producing incoming waves over a viscoelastic bottom.

Incoming HOSRK solver:

- `viscoelastic_bottom_incoming_2D.f` – 2-D solver of periodic waves over a viscoelastic bottom for elevation and velocity potential over time.
- `viscoelastic_bottom_incoming_3D.f` – 3-D solver of periodic waves over a viscoelastic bottom for elevation and velocity potential over time.

Optimization algorithms

This part includes 4 modules corresponding to 4 above-discussed algorithms:

- `random_search.m` – optimization using random search method.
- `genetic_algorithm_DEAP_SCOOP.py` – optimization using genetic algorithm based on the DEAP platform.
- `cross_entropy.py` – optimization using cross entropy method.
- `neural_network.py` – optimization using a neural network.

All of the methods require the data to be prepared beforehand using parallel computing mechanism. Effective parallel computing requires the usage of either open-source MPI or SCOOP that is discussed in section 4.7. This leads to some extra wrapper modules included in the same part.

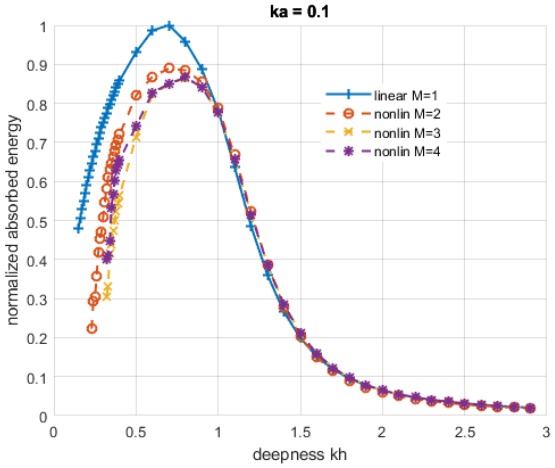
Chapter 7

Linear and Nonlinear Optimization

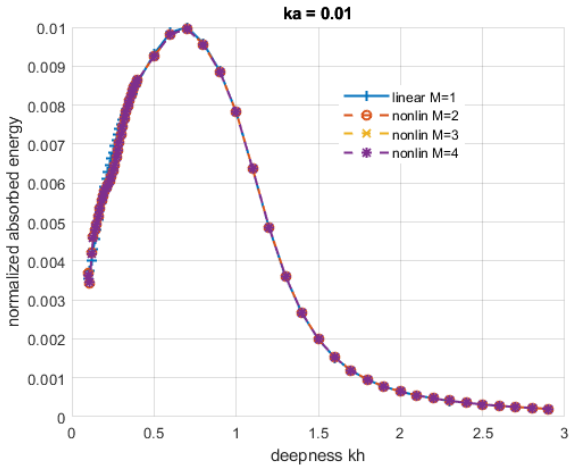
7.1 Nonlinearity Investigation

The HOS method is intended to solve the nonlinear problem, and the HOSRK solver that simulates the wave maker inherits its feature. It is useful to investigate the nonlinearity of the problem before moving to the shape optimization part as we are interested in the optimal shapes in both linear and nonlinear wave cases.

The nonlinearity is characterized by ka and kh. The former is the surface wave stiffness, and the latter is the water deepness. In other words, the larger ka is, the steeper the surface elevation is, and the more nonlinear the wave is. In contrast, the larger kh is, the deeper and the more linear the water is. Figure 8.1 shows the absorbed energy of a baseline circular wave carpet over the range of kh at two typical ka values 0.1 and 0.01.



(a) The linear and nonlinear difference is clearly visible when the steepness ka is large, in which linear wave carpet absorbs more energy than nonlinear one. Beyond the nonlinear mode $M = 3$ there is no much difference of the energy absorption. Wave carpet absorbs the most energy amount at deepness $kh \in (0.6, 0.8)$.



(b) The nonlinear effect does not show up in low-steepness waves no matter how much nonlinearity is added (by increasing the nonlinear mode M). It is the well-known linearization of waves by reducing the wave steepness. The most energy absorption of wave carpet also occurs at $kh \in (0.6, 0.8)$.

Figure 7.1: Nonlinearity investigation of the baseline circular-shaped wave carpet in different steepness ka over a wide range of kh. By trial and error, the steepness $ka = 0.1$ at maximum exposes most the nonlinearity of wave carpet. Any value beyond that risks blowing up the simulation while lower steepness reduces the nonlinear effect. It is observed that, in all trials, the maximum energy absorption occurs at deepness $kh \in (0.6, 0.8)$.

Figure 8.1 presents that the nonlinearity exposes very clearly with $ka = 0.1$ while, with $ka = 0.01$, nonlinear waves are almost linear no matter how big kh is. For $ka = 0.1$, it is expected that the difference in the absorbed energy amounts is large when the water is shallow, i.e. kh is small, and it becomes indistinguishable when we increase the water depth. For all investigated M's and ka's, the maximum absorbed energy amount is achieved at $k \approx 0.7$, so we will take $kh = 0.7$ and $ka = 0.1$ for all simulations, which is regarded as the optimal values. Moreover, it is important to note that the nonlinearity converges at $M = 3$, i.e. further increase of M does not changes the absorbed energy that much. Thus, we only take $M = 3$ in all nonlinear simulations to demonstrate the effect of nonlinearity.

In the next sections, we will present the results of carpet shape optimization in four investigated wave cases combined from different wave characteristics: monochromatic vs polychromatic and unidirectional vs multi-directional, each combination is studied in both linear or nonlinear modes, i.e. $M = 1$ and $M = 3$. For each wave case, we simulate the scenarios occurred in reality. The incoming waves arrive from offshore, propagate over the visco-elastic seabed (the wave carpet) and damp out when approaching the coast. The absorbed energy is estimated for each carpet shape, based on which the genetic algorithm is applied to find the optimal carpet shape.

7.2 Tuning parameters

The energy absorption of wave carpet is not only characterized by its shape, but it is also affected by the system parameters and it own dynamics. It is tremendously complicated to include all of these parameters into the optimization problem as the searching space is exponentially large. Instead, those parameters other than shape parameters are treated as hyper-parameters which are optimized beforehand on the basic circular-shaped carpet. These hyper-parameters include water steepness ka and water deepness kh characterizing the simulation system, and restoring force γ and damping ratio ζ featuring the wave carpet dynamics.

It is not only in this section that these hyper-parameters are tuned. The nonlinear investigation gives the idea about the best simulation setting with water steepness $ka = 0.1$ and water deepness $kh = 0.7$. This section extends the tuning to the carpet dynamic parameters. Specifically, the restoring force and damping ratio are searched over their equidistant grid of values within the suitable ranges. The outcome is a heat map of energy absorption over the grid as presented in Figure 7.2.

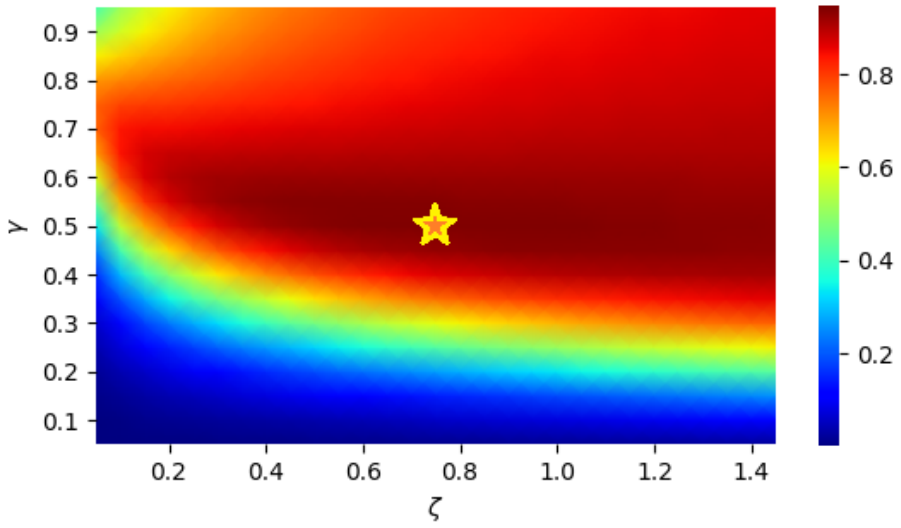


Figure 7.2: Investigation of the energy absorption of the basic circular-shaped wave carpet over a grid of dimensionless restoring force γ and damping ratio ζ values. Color indicates the normalized amount of absorbed energy. For this basic carpet shape, the energy absorption maximizes at $\gamma = 0.5$ and $\zeta = 0.75$, marked by the yellow star. It is the same of both linear and nonlinear cases.

Table 7.1: HYPER-PARAMETERS OF ALL SIMULATIONS

	Notation	Value
Domain size		$2\pi \times 2\pi$
Mesh size		512×512
Wave steepness	$\epsilon = ka$	0.1
Water deepness	$\mu = kh$	0.7
Restoring force	γ	0.5
Damping ratio	ζ	0.75

The heat map represents the magnitude of energy absorption with varying restoring force and damping ratio on the abscissa and coordinate, respectively. The energy absorption is low in response to the low dynamic values. It is because the low restoring dynamics make the carpet more rigid that it does not take over much energy, and the low damping dynamics does

not absorb much energy. The energy absorption increases in accordance with the increasing dynamic parameters to the maximum point. It then fades out when the damping is getting too large that damps the carpet so quickly. The optimal values for the basic circular-shaped carpet are $\gamma = 0.5$ and $\zeta = 0.7$. This is consistent for linear and nonlinear cases.

Table 7.1 summarizes the hyper-parameters for all of the simulations presented in the paper. These hyper-parameters are picked or tuned for different purposes. In particular, the wave steepness $\epsilon = ka$ is selected close to its highest possible value before the numerical simulation becomes unstable and blows up. The water shallowness $\mu = kh$ was searched within its range of 0.2 - 4 for the maximum energy absorption given ϵ. Similarly, the restoring force γ and the damping ratio ζ were grid-searched over their ranges of $0.1 - 0.9$ and $0.1 - 1.4$, respectively, for the maximum energy absorption given ϵ and μ.

7.3 Investigation of Elliptical Shapes and Optimization Plan

The turning of hyper-parameters significantly increases the carpet energy absorption, but there is still more room to its further rise. The potential improvement comes from selecting the correct carpet shape, which turns into the optimization problem. Before that, the primitive idea is to investigate a series of systematically changing shapes for maximum energy absorption. Starting from the basic circular-shaped carpet, we define the elongation factor $p \in [0.5, 2]$ transforming the circular shape into elliptical ones according to the Equation 4.3 while still retaining the same area. The energy absorption of elliptical wave carpets versus elongation p is presented in Figure 7.3. It is obvious that wave carpets with larger capture width, i.e. smaller p or shorter carpet in wave direction, absorb more energy than those with smaller capture width, i.e. larger p or longer carpet in wave direction.

The system of equations are well discussed in [2]. We used High-Order Spectral (HOS) method to numerically solve the problem, assuming the periodic boundary conditions in the computational domain. This results in the numerical scheme that takes as input the wave state and marches it into a new state in every time step. This is hereafter referred to as HOS numerical scheme.

Since the wave carpet problem does not satisfy the periodic boundary conditions, i.e. waves propagate from one side over the wave carpet and are damped towards the other side, this paper extended the HOS method to simulate the wave maker. To achieve this, the wave carpet problem is solved at each time step with periodic boundary conditions. Then waves are damped on one end to eliminate the reflection effect, and other waves are fed into the other end to mimic the wave flapper. The new wave setting is solved again by the aforemen-

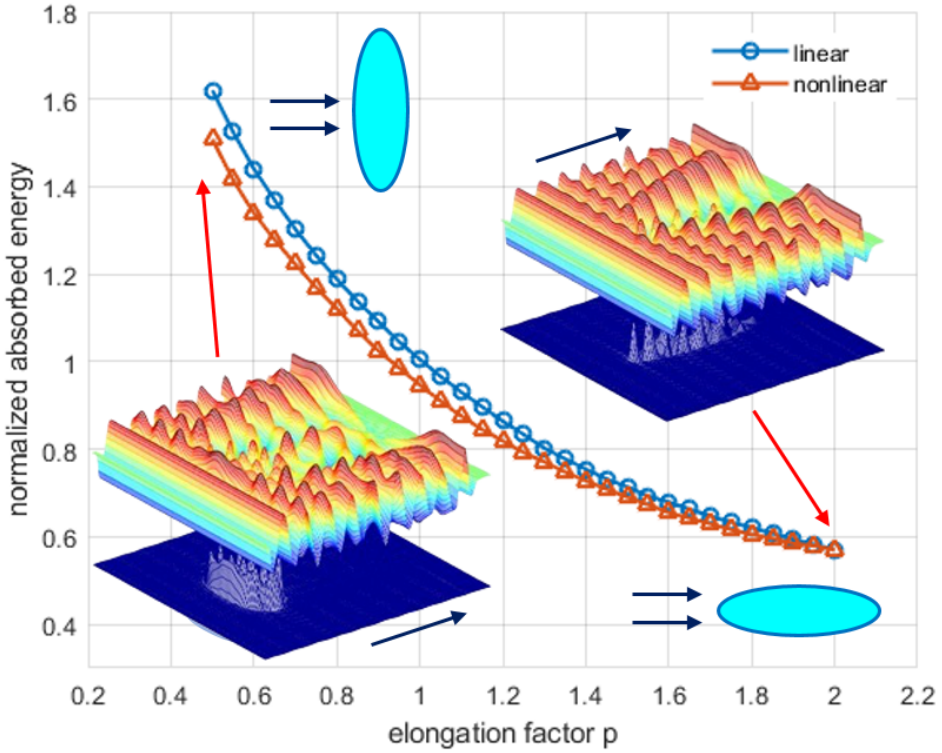


Figure 7.3: Linear and nonlinear investigation of the energy absorption of the elliptical-shaped wave carpet. It is known that nonlinear wave carpet absorbs less energy than linear one, and the greater capture-width a wave carpet has, the more energy it absorbs. The capture width is characterized by the elongation factor p, in which low p is associated with high capture width.

tioned numerical scheme. These operations occur repeatedly over time steps, resulting in the simulated wave maker that lets waves propagate from one end, over the wave carpet, and vanish at the other end without reflection. The wave maker application is further developed into the so-called HOS solver that takes as input the wave carpet configuration and outputs the absorbed energy.

The discussion of the HOS numerical scheme and the HOS solver is out of the scope of this paper. It concentrates on how to use their application to achieve the optimal carpet shape. Before solving the optimization problem, the simulation settings are outlined and the sufficient validations are provided to ensure that the HOS solver correctly solves the wave carpet problem. The simulation settings involve the hyper-parameters that are

fixed for all simulations, presented in Table 7.1. The changeable ones such as period T, wave number k, are called experimenting parameters that will be addressed from case to case of waves in later section. The validation checks the correctness and convergence of the numerical solution and its compliance with the universal physics law - conservation of energy.

The following four wave cases are investigated:

- **Wave case I** - Single-Frequency Uni-Directional (1F 1D): symmetric uniform waves coming from one direction.
- **Wave case II** - Single-Frequency Multi-Directional (1F MD): symmetric uniform waves coming from multiple directions.
- **Wave case III** - Multi-Frequency Uni-Directional (MF 1D): symmetric uniform waves coming from multiple directions.
- **Wave case IV** - Multi-Frequency Multi-Directional (MF MD): symmetric non-uniform waves coming from multiple directions.
- **Wave case V** - Asymmetric Multi-Frequency Multi-Directional (A MF MD): asymmetric non-uniform waves coming from multiple directions.

in which, wave cases I, II, IV and IV are used for linear optimization, and wave cases I, II and III for nonlinear optimization.

Table 7.2: PARAMETERS OF EACH SIMULATION

Wave Case	Wave Number k	Period T	Wave Dir ^{0}
1F 1D	20	1.81	0
1F MD	20	1.81	0, $\pm$ 37
MF MD	20, 24	1.81, 1.66	$\pm$ 37, $\pm$ 53
A MF MD	20, 24	1.81, 1.66	37, 53

The wave parameters, including period T, wave number k and wave direction are presented in Table 7.2. Combining with Table 7.1, it forms a whole picture for each simulation of wave carpet in each wave case. The optimal energy absorption obtained by the approximating neural network is validated against that of the HOS solver as described in line 12 versus line 15 in algorithm 2 and presented in column 4 of Table 7.3. This is a critical step to ensure the close approximation of the neural network to the true HOS solver. The approximation errors were aimed to be less than 5% and are presented in column 5 of Table 7.3.

7.4 Linear Optimization Results

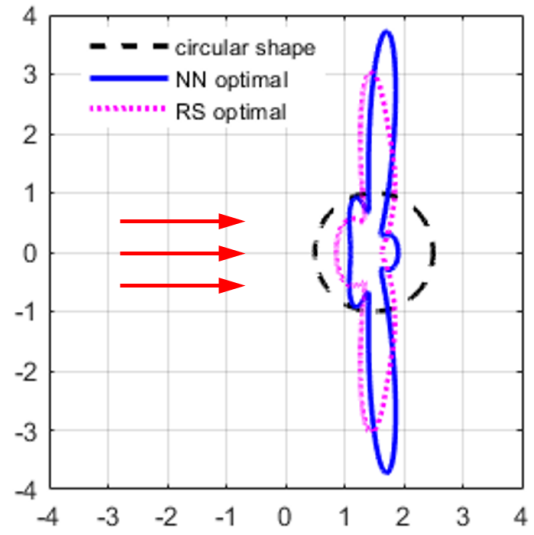
The linear optimization employs the random search and the neural network approximation methods. The choice of these methods is intended to reserve more computing resources for nonlinear optimization. The linear simulations are implemented in four wave cases with the parameters discussed in Section 7.3. Each wave cases is treated as an individual optimization problem. The random search takes advantage of the generated data by the neural network. It thereby reduces the computing time and cost by half.

The optimal carpet shapes in these wave cases are presented in Figure 7.4, and their comparison with the baseline and those of the random search method is summarized in Table 7.3. The baseline is the basic circular shape in each wave cases. All absorbed energy amounts are normalized by the baseline in the respective wave case. The best shape of random search method is the shape with the maximum energy absorption among the randomly generated carpet shapes. Thus, the comparison with the random search shape points out the need to optimize the shape instead of picking the best random shape since the neural network-based optimal shapes considerably absorbed more energy in each wave case.

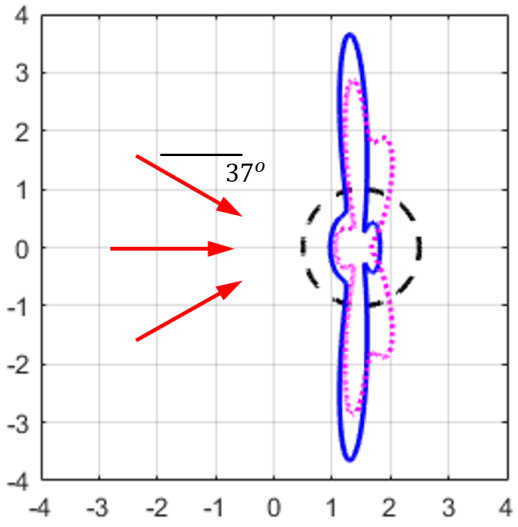
Quantitatively, the energy gains compared to the baseline are summarized in the table 7.3, in which the baseline is the circular carpet shape in the same wave cases.

Table 7.3: THE ENERGY GAINS OF NEURAL NETWORK VS RANDOM SEARCH METHODS IN DIFFERENT WAVE CASES

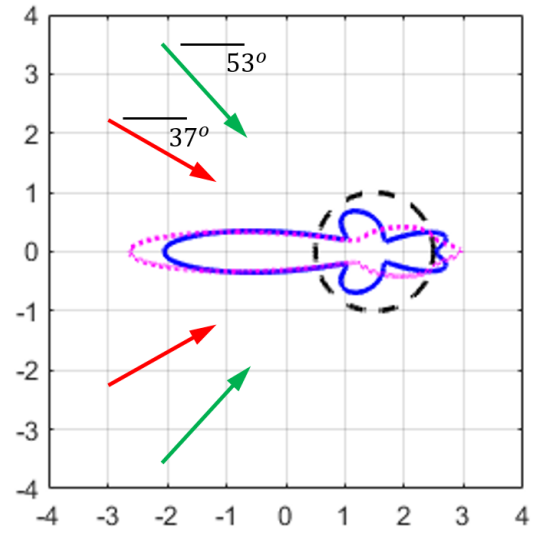
Wave Case	Baseline	Rand Search	NN	HOS	Error
1F 1D	1	1.78	2.18	2.182	0.1%
1F MD	1	1.52	2.01	2.021	0.5%
MF MD	1	2.26	2.99	3.014	0.8%
A MF MD	1	3.04	7.47	7.14	4.6%



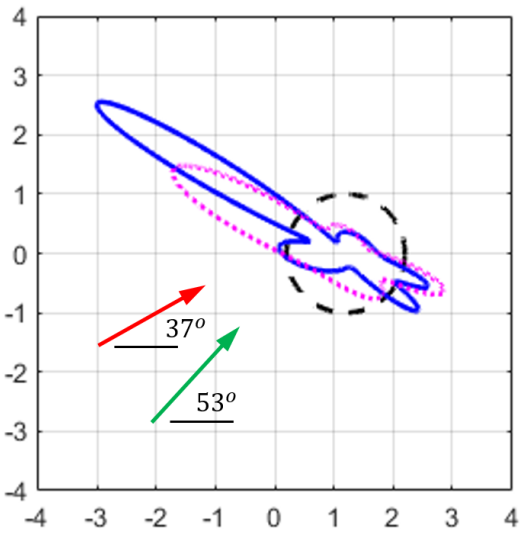
(a) Wave case I - Single-frequency unidirectional (1F 1D). Energy gain: 2.18



(b) Wave case II - Single-frequency multi-directional (1F MD). Energy gain: 2.01



(c) Wave case IV - Multi-frequency multi-directional (MF MD). Energy gain: 2.99



(d) Wave case V - Asymmetric multi-frequency multi-directional (A MF MD). Energy gain: 7.47

Figure 7.4: Optimal and sub-optimal carpet shapes in different linear wave cases. The optimal shapes (blue) are achieved by the Neural Network-based Optimization method. The sub-optimal shapes (dotted pink) are by the Random Search method. The dash-line circle (black) is the baseline circular shape with the same area. Arrows indicate wave directions. Different arrow colors indicate different wave frequencies.

To summarize the results, the optimal shape is found in each investigated wave case and the approximation errors are less than the common target value of 5%. It is important to note that the neural network is an heuristic approach, this means the optimal shapes are not truly optimal but the closest ones to the global optima found by this method. The heuristic property is common for several search methods such as response surface methodology, stochastic gradient descent, genetic algorithm [30], etc. unless the objective function is convex, which is not the case of our problem.

Despite the heuristic property, the advantage of the neural network method is its fully paralleled computation at the very beginning and its requirement of only two rounds of evaluating the objective function. These features enable the method to optimize a computationally expensive objective function as demonstrated in the paper. Compared to the neural network, the random search cannot reach as close to the optimal shapes although it helps improve the energy gains and spend the same amount of time and computing resources; the stochastic gradient descent and the genetic algorithm requires, roughly estimated, $N/2$ times more of computing time and resources as they need to evaluate the objective function sequentially N times, where N is the numbers of iterations of generation, respectively. The response surface methodology using a polynomial to approximate the response surface was attempted but poorly approximating the response surface, i.e. the energy absorption, of our problem due to the input multi-dimensionality.

It is noticed that the approximation errors increase in accordance with the complexity of the wave cases. In specific, the single-frequency unidirectional wave case has the smallest error while the single-frequency multi-directional and multi-frequency multi-directional have the larger errors, and the asymmetric multi-frequency multi-directional has the much larger error than the other cases. Our further investigating simulations indicate that the multi-directional wave cases require more time for the simulation to reach their stable states, the fact that all the simulations are implemented in the same number of periods $14T$ makes these wave cases less stable than the simple wave case, thereby introducing more errors to the neural network approximation. The asymmetric multi-frequency multi-directional wave case has the largest error because its input has 12 components compared to 6 components of the other cases. Therefore, it should need more data points for the neural network to learn and to perform as well as the other cases.

It is interesting to look at optimal shapes of the wave cases and to attempt interpreting the results. First, the overall shapes are oblong and have some complex curves in the middle. These strange curves come from the fact that the input constraints in equation 4.5 are neither general enough to universally represent all possible shapes nor stringent enough to eliminate the complicated ones. Since the input constraints are not very selective, the resulting complicated curves have to be accepted as a part of the optimization results. Second, since the optimal results are somewhat similar to the elliptical shapes, one may raise a question if the optimal shape should be replaced by the elliptical shapes. The answer is no for two reasons:

1. The elliptical shape is not optimal as it absorbs less energy; 2. Cases 1, 2 and 4 have the oblong shapes normally to the resultant wave direction while the oblong shape in case 2 is tangent to the resultant wave direction and its long axis seems to shorter that those of the other cases. Thus, these elliptical shapes cannot always be easily guessed without the optimization procedure. It is, however, somewhat intuitive that the oblong shapes normal to wave direction absorb more energy than those tangent to wave direction. This intuition is qualitatively verified by Figure 7.3, where the broad elliptical wave carpet absorbs more energy than long elliptical one [32].

7.5 Nonlinear Optimization Results

The nonlinear optimization employs the cross-entropy and the genetic algorithm methods. Compared to the random search and neural network, these methods require less parallel computing flows, but taking longer computing time as they are semi-sequential. The simulations are implemented with the wave cases I, II and III and and with the same settings as the linear section 7.3. Due to the limited computing resources, the multi-directional wave cases are not simulated. Two optimization methods are applied separately and their results are summarized in Table 7.4.

Table 7.4: THE ENERGY GAINS OF CROSS-ENTROPY METHOD VS GENETIC ALGORITHM IN DIFFERENT WAVE CASES

Wave Case	Baseline	Linear	Cross Entropy	Genetic Algorithm
1F 1D	1	2.182	2.048	2.067
1F MD	1	2.021	1.938	1.977
MF 1D	1	1.987	1.950	1.974

It is observed that the results of the cross entropy-based optimization are sub-optimal while these of the genetic algorithm-based optimization is optimal. Both of the results are closely less than these of the linear optimization. This is consistent with the elliptical shape investigation that the nonlinear carpet absorb less energy than the linear one. These methods directly use the true objective function HOSRK, so there exist no error as in the neural network-bases optimization method. The baselines are, again, the absorbed energy of the circular shapes in the same linear wave cases. The nonlinear optimal shapes are presented in Appendix C.

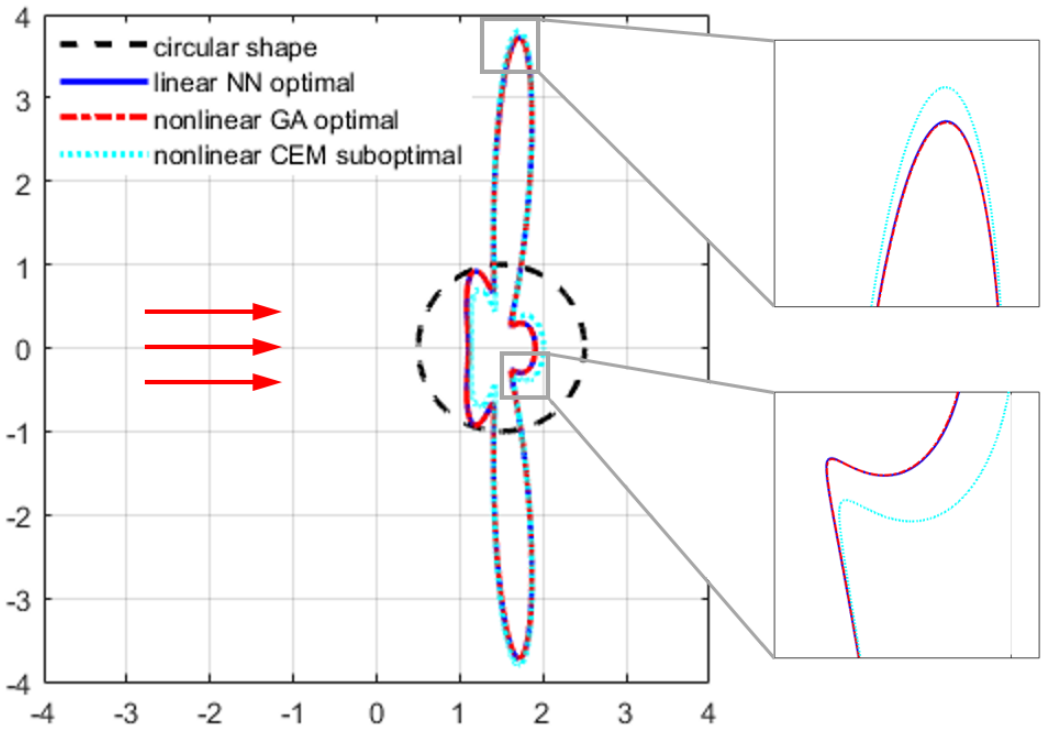
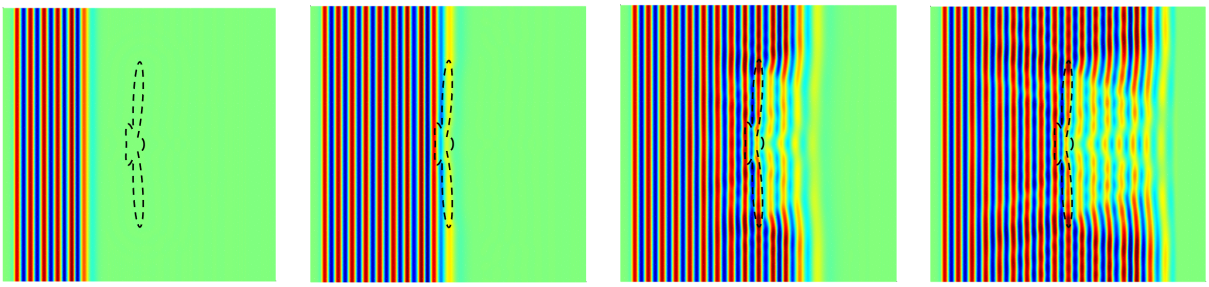
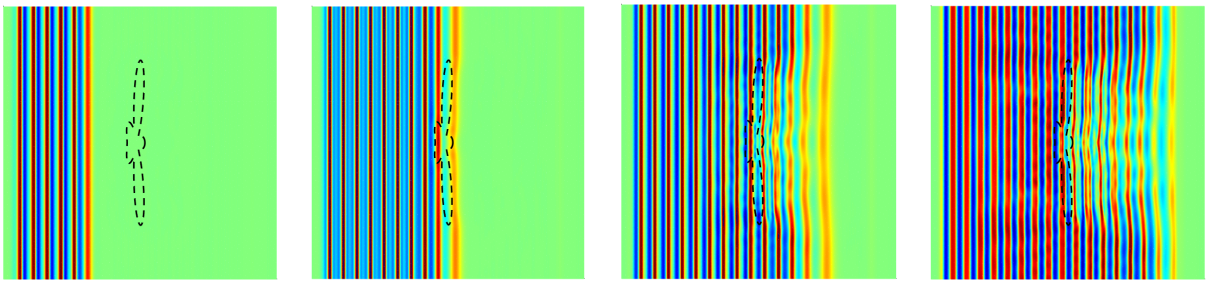


Figure 7.5: Nonlinear optimal and sub-optimal carpet shapes versus linear optimal shape in wave case I - Single-frequency unidirectional (1F 1D). The nonlinear optimal shape (dash-dotted red) is achieved by the Genetic Algorithm method. The sub-optimal shape (cyan-dotted) is by the Cross-Entropy method. The linear and nonlinear optimal shapes are almost the same while the nonlinear sub-optimal shape is slightly different from the optimal ones.

The nonlinear optimal shapes are indeed very similar to the linear optimal shapes in all three wave cases I, II and III. Figure 7.5 presents these shapes in wave case I only, in which the NN-based linear optimal shape and GA-based nonlinear optimal shapes are indistinguishable while the CEM-based nonlinear optimal shape is a little offset from the optimal one. Figure 7.5 only shows the shapes in wave case I as as wave cases II and III follow the same pattern. In numerical terms, the difference of linear and nonlinear optimal shapes can be subtly discerned through comparing three sets of slightly different parameters in Tables C.2, C.3 and C.4 in Appendix C.



(a) Linear. 4T **(b)** Linear. 8T **(c)** Linear. 12T **(d)** Linear. 16T



(e) Nonlinear. 4T **(f)** Nonlinear. 8T **(g)** Nonlinear. 12T **(h)** Nonlinear. 16T

Figure 7.6: Patterns of linear versus nonlinear waves propagating over the optimal wave carpet in wave case I captured at different time points. Despite of nearly the same optimal carpet shapes, linear waves have a consistent pattern over time while nonlinear waves have varying patterns over time due to the nonlinear effect.

Another visual investigation of linear versus nonlinear difference is shown in Figure 7.6. Four top color maps present linear waves, and four bottom ones are nonlinear waves. Linear waves have consistent wave patterns as the wave amplitude is stable over time. Nonlinear waves, due to the nonlinear effect, have wave patterns change over time indicating the variation in wave amplitude.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

Starting from the inherited rigid bottom HOS solver, we extended it into the periodic visco-elastic bottom HOS solver, and further into the non-periodic visco-elastic wave carpet HOSRK solver. Incorporating the original rigid bottom solver, these form the entire framework for simulating various wave cases with different wave conditions and rigid bottom topographies or visco-elastic bottom. This numerically robust model is highly flexible to allow the adjustment of the visco-elastic region and its viscosity-elasticity. Thanks to this feature, we extensively investigated and successfully solved the particular problem of wave carpet shape optimization in linear and nonlinear modes.

We investigated different aspects of the wave carpet model, including its stability, nonlinearity and optimal hyper-parameters. Independent from the hyper-parameters, the carpet exposes its most nonlinearity at steepness $ka = 0.1$ and maximizes its energy absorption at deepness $kh = 0.7$. The optimal carpet hyper-parameters for its most energy absorption are restoring force $\gamma = 0.5$ and damping ratio $\zeta = 0.75$. All the analyses are conducted after the stable time points when the simulations reach their stable state.

We reviewed and suggested 4 heuristic methods for the optimization problems: Random Search, Neural Network, Cross-Entropy and Genetic Algorithm. The Random Search and Neural Network methods are attempted for the linear optimization. The optimal carpet shapes are achieved in all investigated wave cases by Neural Network-based optimization method, compared to the sub-optimal shapes by the Random Search. We found that the optimal-shaped carpets absorbs energy approximately twice as much as the baseline circular-shaped one.

The Cross-Entropy and Genetic Algorithm methods are attempted for the nonlinear optimization. The optimal shapes are achieved by the latter method while the former leads

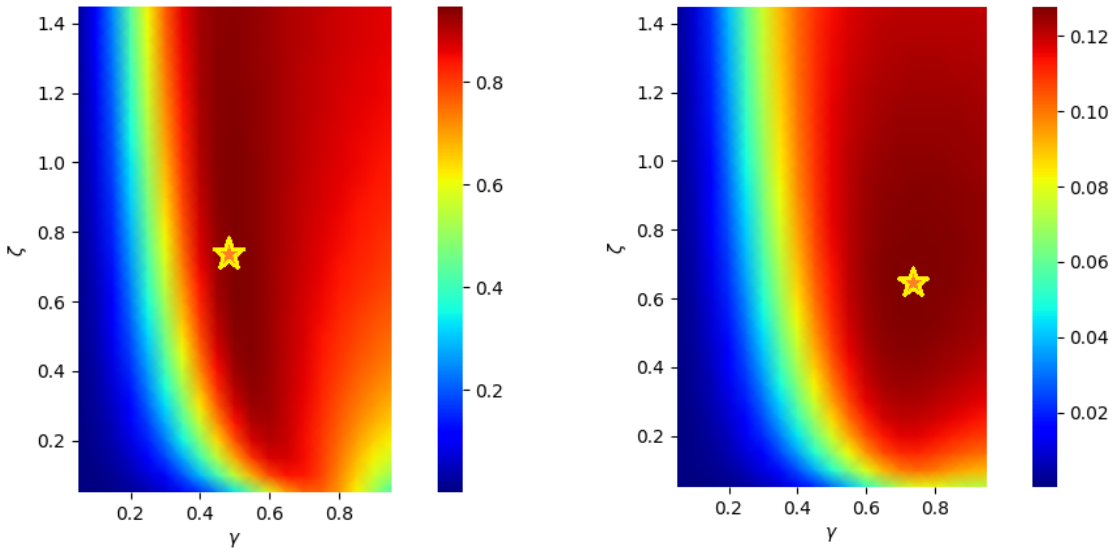
to the suboptimal ones. The nonlinear wave carpets absorb aprroximately 7% less energy than the linear ones in the same wave cases. However, the nonlinear optimal shapes are very similar to the linear optimal ones despite the differences in the energy absorption and the waves pattern. This suggests that, for the sake of simplicity, the Genetic Algorithm can be both applied for both the linear and nonlinear optimizations.

The project does not end at the wave carpet shape optimization. Its true legacy lies at the complete framework for modelling a flexible wave maker encompassing any surface waves and bottom conditions. Parallel computing setup empowers its potential usage for complicated and computationally expensive problems, such as ones involving the 3-D model or heavily dependent on the solver executions. As a suggestion, further potential research effort should be focused on the development of a parameters auto-tuning wave maker adapting to different sea states.

8.2 Future work

In this study, we developed a computational environment that simulates the wave maker using High-Order Spectral method, called the HOSRK solver. The wave maker simulates the real scenario that incoming waves passing over the wave carpet damp out and transfer the energy to the carpet. The HOSRK solver takes as input the carpet parameters and hyper-parameters and output the absorbed energy. Many experiments were carried out in the simulation environment for optimizing the wave carpet. As a result, the optimal carpets were achieved in terms of their shape parameters including restoring force γ and damping ratio ζ, and shape parameters for the different wave cases [6].

Such optimal carpets are, however, static. They were optimized for the corresponding wave cases and are sub-optimal for the other wave cases. Our goal is to build a dynamic system so that its hyper-parameters can automatically change to the optimal values adapting to the change of sea state. This can be done by developing a smart wave carpet that can learn over time with the reinforcement learning algorithms such as imitation learning and actor-critic. Within the scope of that project, it is recommended to concentrate on the linear and nonlinear 2-dimensional wave carpet problem, ignoring the carpet shape, since it is impractical to change the shape after the carpet has been installed. The 2-D problem, corresponding to the 2-D HOSRK solver, is computationally solvable in terms of time complexity, thereby enabling the data generation and the application of computationally expensive reinforcement learning algorithms. Without the loss of generality, the solution to the 2-D problem serves a good purpose of demonstrating the application of reinforcement learning to the 3D problem as long as the training data are sufficiently generated in a short time.



(a) Heat map of absorbed energy vs hyper-parameters γ and ζ in sea state $k = 20$, $ka = 0.1$, $kh = 0.7$. The optimal point is $\gamma = 0.5, \zeta = 0.75$.

(b) Heat map of absorbed energy vs hyper-parameters γ and ζ in sea state $k = 30$, $ka = 0.07$, $kh = 1.0$. The optimal point is $\gamma = 0.75, \zeta = 0.65$.

Figure 8.1: Investigation of the optimal hyper-parameters γ and ζ at two different sea states. It is observed that the optimal point of hyper-parameters is not static when the sea state changes. This suggests the future work of auto-optimizing the hyper-parameters according to the variation of sea state.

The further investigation indicates that building a smart wave carpet using reinforcement learning is feasible based on two facts. First, the wave carpet has different optimal hyper-parameter points in different wave cases, as featured in Figures 8.1a and 8.1b. In other words, when the sea state changes, the carpet should learn to alter its hyper-parameters to reach its new optimal state. Second, the HOSRK solver is sensitive to neural network that is the core of the intended reinforcement leaning algorithms. Thus, the HOSRK solver-based wave carpet can be trained by the neural network with newly collected data over time to learn new states.

Acknowledgement

This research used the Savio computational cluster resource provided by the Berkeley Research Computing program at the University of California, Berkeley (supported by the UC Berkeley Chancellor, Vice Chancellor for Research, and Chief Information Officer).

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231.

This research used resources of Wikipedia, the free encyclopedia [100].

Bibliography

[1] C.C. Aggarwal. *Neural Networks and Deep Learning*. Cham: Springer, 2018, p. 497. ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463-0.

[2] M.-R. Alam. "Nonlinear analysis of an actuated seafloor-mounted carpet for a high-performance wave energy extraction". In: *Royal Society of London Proceedings Series A* 468 (Oct. 2012), pp. 3153–3171. DOI: 10.1098/rspa.2012.0193.

[3] M.-R. Alam, M. Lehmann, and M. Shakeri. *Carpet of Wave Energy Conversion (CWEC)*. US Patent (US 9,777,701 B2), Oct. 3rd 2017.

[4] M.-R. Alam, Y. Liu, and D.K.P. Yue. "Bragg resonance of waves in a two-layer fluid propagating over bottom ripples. Part I. Perturbation analysis". In: *Journal of Fluid Mechanics* 624 (Apr. 2009), pp. 191–224. DOI: 10.1017/S0022112008005478.

[5] B. Amos and D. Yarats. "The Differentiable Cross-Entropy Method". In: *ICML*. 2020.

[6] J.A. Anderson. *An Introduction to Neural Networks*. The MIT Press, Mar. 1995. ISBN: 9780262315883. DOI: 10.7551/mitpress/3905.003.0011. URL: https://doi.org/10.7551/mitpress/3905.003.0011.

[7] J.S. Arora. "18 - Global Optimization Concepts and Methods for Optimum Design". In: *Introduction to Optimum Design (Second Edition)*. Ed. by Jasbir S. Arora. Second Edition. San Diego: Academic Press, 2004, pp. 565–592. ISBN: 978-0-12-064155-0. DOI: https://doi.org/10.1016/B978-012064155-0/50018-5. URL: https://www.sciencedirect.com/science/article/pii/B9780120641550500185.

[8] A.Y. Aydar. "Utilization of Response Surface Methodology in Optimization of Extraction of Plant Materials". In: *Statistical Approaches With Emphasis on Design of Experiments Applied to Chemical Processes*. Ed. by Valter Silva. Rijeka: IntechOpen, 2018. Chap. 10. DOI: 10.5772/intechopen.73690. URL: https://doi.org/10.5772/intechopen.73690.

[9] M. Azam. "Energy and economic growth in developing Asian economies". In: *Journal of the Asia Pacific Economy* 25.3 (2020), pp. 447–471. DOI: 10.1080/13547860.2019.1665328. URL: https://doi.org/10.1080/13547860.2019.1665328.

[10] J. Bard. *"Implementing Agreement on Ocean Energy Systems (IEA-OES), Annual Report 2007*. URL: https://www.webcitation.org/6ZhLw1yYn?url=http://pt21.ru/docs/pdf/28_e.pdf. Last visited on 07/29/2021. International Energy Agency - Ocean Energy Systems, July 2015.

[11] S. Beji and J.A. Battjes. "Experimental investigation of wave propagation over a bar". In: *Coastal Engineering* 19.1 (1993), pp. 151–162. ISSN: 0378-3839. DOI: https://doi.org/10.1016/0378-3839(93)90022-Z. URL: https://www.sciencedirect.com/science/article/pii/037838399390022Z.

[12] D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. 1st ed. Athena Scientific, 1996. ISBN: 1886529043.

[13] P.-T. de Boer, D.P. Kroese, S. Mannor, and R.Y. Rubinstein. "A Tutorial on the Cross-Entropy Method". In: *Annals of Operations Research* 134 (February 2005), pp. 19–67. DOI: https://doi.org/10.1007/s10479-005-5724-z.

[14] T. Börner and M.-R. Alam. "Wave Carpet Optimization via Real Time Hybrid Modeling". In: vol. Volume 9: Ocean Renewable Energy. International Conference on Offshore Mechanics and Arctic Engineering. V009T09A036. May 2015. DOI: 10.1115/OMAE2015-42365. URL: https://doi.org/10.1115/OMAE2015-42365.

[15] Z.I. Botev, D.P. Kroese, R.Y. Rubinstein, and P. L'Ecuyer. "Chapter 3 - The Cross-Entropy Method for Optimization". In: *Handbook of Statistics*. Ed. by C.R. Rao and Venu Govindaraju. Vol. 31. Handbook of Statistics. Elsevier, 2013, pp. 35–59. DOI: https://doi.org/10.1016/B978-0-444-53859-8.00003-5. URL: https://www.sciencedirect.com/science/article/pii/B9780444538598000035.

[16] L. Bottou. "Online Algorithms and Stochastic Approximations". In: *Online Learning and Neural Networks*. Ed. by David Saad. revised, oct 2012. Cambridge, UK: Cambridge University Press, 1998. URL: http://leon.bottou.org/papers/bottou-98x.

[17] L. Bottou. "Stochastic Gradient Learning in Neural Networks". In: AT&T Bell Laboratories. Holmdel, NJ, 1991.

[18] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441.

[19] BP. *Statistical Review of World Energy*. 2021. URL: https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html (visited on 07/29/2021).

[20] K. Carley, N. Kamneva, and J. Reminga. *Response Surface Methodology - CASOS Technical Report*. URL: http://www.casos.cs.cmu.edu/publications/papers/CMU-ISR-04-136.pdf. Last visited on 07/29/2021. CMU, ISRI and CASOS, Oct. 2004, p. 32.

[21] K. Chandra, E. Meijer, S. Andow, E. Arroyo-Fang, I. Dea, J. George, M. Grueter, B. Hosmer, S. Stumpos, A. Tempest, and S. Yang. "Gradient Descent: The Ultimate Optimizer". In: *ArXiv* abs/1909.13371 (2019).

[22] A.R. Conn, N.I.M. Gould, and P.L. Toint. "A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds". In: *SIAM Journal on Numerical Analysis* 28.2 (Apr. 1991), pp. 545–572. DOI: 10.1137/0728030.

[23] B.C. Csáji. "Approximation with Artificial Neural Networks". MSc thesis. Budapest, Hungary: Faculty of Sciences, Eötvös Loránd University, 2001.

[24] B. Czech and P. Bauer. "Wave Energy Converter Concepts: Design Challenges and Classification". In: *IEEE Industrial Electronics Magazine* 6.2 (2012), pp. 4–16. DOI: 10.1109/MIE.2012.2193290.

[25] I. Dabbura. *Gradient Descent Algorithm and its Variants*. Dec. 2017. URL: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3 (visited on 07/29/2021).

[26] L. Dalcin. *MPI for Python*. 2021. URL: https://mpi4py.readthedocs.io/en/stable/index.html (visited on 07/29/2021).

[27] Dask Development Team. *Dask: Library for dynamic task scheduling*. 2016. URL: https://dask.org.

[28] Our World in Data. *Primary Energy Consumption by World Region*. 2021. URL: https://ourworldindata.org/grapher/primary-energy-consumption-by-region (visited on 07/29/2021).

[29] Our World in Data. *World Energy Consumption by Source*. 2021. URL: https://ourworldindata.org/grapher/energy-consumption-by-source-and-region (visited on 07/29/2021).

[30] K. Deb. "An introduction to genetic algorithms". In: *Sadhana* 24 (1999), pp. 293–315.

[31] *Design of Experiments. Lesson 11: Response Surface Methods and Designs*. URL: https://online.stat.psu.edu/stat503/lesson/11. Last visited on 07/29/2021. PennState Eberly College of Science.

[32] N.H. Do and M.-R. Alam. "Data-Based Approach to Optimizing the Ocean Wave Energy Carpet Using Deep Neural Network". In: vol. 9: Ocean Renewable Energy. International Conference on Offshore Mechanics and Arctic Engineering. V009T09A027. Aug. 2020. DOI: 10.1115/OMAE2020-18865. URL: https://doi.org/10.1115/OMAE2020-18865.

[33] D.G. Dommermuth and D.K.P. Yue. "A high-order spectral method for the study of nonlinear gravity waves". In: *Journal of Fluid Mechanics* 184 (Nov. 1987), pp. 267–288. DOI: 10.1017/S002211208700288X.

[34] D.G. Dommermuth and D.K.P. Yue. "A high-order spectral method for the study of nonlinear gravity waves". In: *Journal of Fluid Mechanics* 184 (1987), pp. 267–288. DOI: 10.1017/S002211208700288X.

[35] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. 2nd. Springer Publishing Company, Incorporated, 2015. ISBN: 3662448734.

[36] R.B. Elandt, M. Shakeri, and M.-R. Alam. "Surface gravity-wave lensing". In: *Phys. Rev. E* 89 (2 Feb. 2014), p. 023012. DOI: 10.1103/PhysRevE.89.023012. URL: https://link.aps.org/doi/10.1103/PhysRevE.89.023012.

[37] S. Esmaeilzadeh and M.-R. Alam. "Shape Optimization of Wave Energy Converters for Broadband Directional Incident Waves". In: *Ocean Engineering* 174 (November 2016), pp. 186–200. DOI: 10.1016/j.oceaneng.2019.01.029.

[38] S. Ferrari and R.F. Stengel. "Smooth function approximation using neural networks". In: *IEEE Transactions on Neural Networks* 16.1 (2005), pp. 24–38. DOI: 10.1109/TNN.2004.836233.

[39] C.A. Floudas and P.M. Pardalos. *Encyclopedia of Optimization*. 2nd ed. Springer, 2009. ISBN: 978-0-387-74758-3.

[40] B. Fortuner. *Can Neural Networks Solve Any Problem? - Visualizing the Universal Approximation Theorem*. Mar. 2017. URL: https://towardsdatascience.com/can-neural-networks-really-learn-any-function-65e106617fc6 (visited on 07/29/2021).

[41] D.A. Freedman. *Statistical Models: Theory and Practice*. 2nd ed. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511815867.

[42] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618.

[43] P. Grohs, D. Perekrestenko, D. Elbrächter, and H. Bölcskei. "Deep Neural Network Approximation Theory". In: *IEEE Transactions on Information Theory* PP (Jan. 2019). DOI: 10.1109/TIT.2021.3062161.

[44] W. Gropp and E. Lusk. *An Introduction to MPI - Parallel Programming with Message Passing Interface*. URL: https://www.mcs.anl.gov/research/projects/mpi/tutorial/mpiintro/ppframe.htm. Last visited on 07/29/2021. Argonne National Laboratory, 2002.

[45] Q.-C. Guo. "Rogue waves in unidirectional seas: statistics and prediction". PhD thesis. Berkeley, CA 94720, USA: Department of Mechanical Engineering, University of California, 2018.

[46] Q.-C. Guo and M.-R. Alam. "Prediction of Oceanic Rogue Waves Through Tracking Energy Fluxes". In: vol. Volume 3A: Structures, Safety and Reliability. International Conference on Offshore Mechanics and Arctic Engineering. V03AT02A014. June 2017. DOI: 10.1115/OMAE2017-62261. URL: https://doi.org/10.1115/OMAE2017-62261.

[47] K. Gurney. *An Introduction to Neural Networks*. USA: Taylor & Francis, Inc., 1997. ISBN: 1857286731.

[48] R. Hannesson. "Energy and GDP growth". In: *International Journal of Energy Sector Management* 3 (June 2009), pp. 157–170. DOI: 10.1108/17506220910970560.

[49] M. He, W.-H. Xu, X.-F. Gao, and B. Ren. "SPH Simulation of Wave Scattering by a Heaving Submerged Horizontal Plate". In: *International Journal of Ocean and Coastal Engineering* 01.02 (2018), p. 1840004. DOI: 10.1142/S2529807018400043. URL: https://doi.org/10.1142/S2529807018400043.

[50] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[51] Y. Hold-Geoffroy, O. Gagnon, and M. Parizeau. "Once you SCOOP, no need to fork". In: *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*. ACM. 2014, p. 60.

[52] H.-H. Hsu and Y.-C. Wu. "Scattering of water wave by a submerged horizontal plate and a submerged permeable breakwater". In: *Ocean Engineering* 26.4 (1998), pp. 325–341. ISSN: 0029-8018. DOI: https://doi.org/10.1016/S0029-8018(97)10032-4. URL: https://www.sciencedirect.com/science/article/pii/S0029801897100324.

[53] A. Izmailov and M. Solodov. *Newton-Type Methods for Optimization and Variational Problems*. Mar. 2014. ISBN: 978-3-319-04246-6. DOI: 10.1007/978-3-319-04247-3.

[54] A. Jentzen and A. Riekert. *A proof of convergence for stochastic gradient descent in the training of artificial neural networks with ReLU activation for constant target functions*. 2021. arXiv: 2104.00277 [math.NA].

[55] T. Johansson, K. McCormick, L. Neij, and W.C. Turkenburg. "The Potentials of Renewable Energy". In: *Renewable Energy: A Global Review of Technologies, Policies and Markets* (Jan. 2012). DOI: 10.4324/9781849772341.

[56] N. Jorge and S. J. Wright. *Numerical Optimization*. second. New York, NY, USA: Springer, 2006.

[57] D. Karmakar and C.G. Soares. "Wave Motion Control Over Submerged Horizontal Plates". In: *Journal of Offshore Mechanics and Arctic Engineering* 140.3 (Dec. 2017). 031101. ISSN: 0892-7219. DOI: 10.1115/1.4038500. URL: https://doi.org/10.1115/1.4038500.

[58] C.T. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, 1999. DOI: 10.1137/1.9781611970920. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611970920. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611970920.

[59] M. Kelly and M.-R. Alam. "Shape Optimization of a Submerged Pressure Differential Wave Energy Converter for Load Reductions". In: vol. Volume 10: Ocean Renewable Energy. International Conference on Offshore Mechanics and Arctic Engineering. V010T09A030. June 2019. DOI: 10.1115/OMAE2019-96390. URL: https://doi.org/10.1115/OMAE2019-96390.

[60] A. Khuri. *Response Surface Methodology and Related Topics*. Jan. 2006. ISBN: 978-981-256-458-0. DOI: 10.1142/5915.

[61] D.P. Kingma and J.L. Ba. "ADAM: A Method For Stochastic Optimization". In: *ICLR*. Dec. 2015.

[62] D. Klein. "Lagrange Multipliers without Permanent Scarring". In: (Aug. 2021).

[63] V. Kumar. *Introduction to Parallel Computing*. 2nd. USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0201648652.

[64] P.K. Kundu. *Fluid Mechanics*. Cambridge, Massachusetts: Academic Press, Aug. 1990. ISBN: 9780198520115.

[65] M. Lehmann, R. Elandt, H. Pham, R. Ghorbani, M. Shakeri, and M.-R. Alam. "An artificial seabed carpet for multidirectional and broadband wave energy extraction: Theory and Experiment". In: *Proceedings of 10th European Wave and Tidal Energy Conference*. Aalborg, Denmark, Sept. 2013.

[66] S.-Y. Liang and R. Srikant. "Why Deep Neural Networks for Function Approximation?" In: *ICLR*. 2017.

[67] C.-R. Liu, Z.-H. Huang, and W.-P. Chen. "A Numerical Study of a Submerged Horizontal Heaving Plate as a Breakwater". In: *Journal of Coastal Research* 33.4 (2017), pp. 917–930. DOI: 10.2112/JCOASTRES-D-16-00152.1. URL: https://doi.org/10.2112/JCOASTRES-D-16-00152.1.

[68] Y. Liu, H.-J. Li, and Y.-C. Li. "A new analytical solution for wave scattering by a submerged horizontal porous plate with finite thickness". In: *Ocean Engineering* 42 (Mar. 2012), pp. 83–92. DOI: 10.1016/j.oceaneng.2012.01.001.

[69] Y. Liu and Y.-C. Li. "An alternative analytical solution for water-wave motion over a submerged horizontal porous plate". In: *Journal of Engineering Mathematics* 69 (Apr. 2011), pp. 385–400. DOI: 10.1007/s10665-010-9406-8.

[70] Y. Liu and D.K.P. Yue. "On generalized Bragg scattering of surface waves by bottom ripples". In: *Journal of Fluid Mechanics* 356 (1998), pp. 297–326. DOI: 10.1017/S0022112097007969.

[71] M. Mitchell. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998. ISBN: 0262631857.

[72] B.R. Munson, D.F. Young, and T.H. Okiishi. *Fundamentals of Fluid Mechanics*. 5th Edition. Hoboken, New Jersey: Wiley, Mar. 2005. ISBN: 0471675822.

[73] N.A. Nguyen and T. Shibayama. "Wave-Current Interaction with Mud Bed". In: *Coastal Engineering 1994*, pp. 2913–2927. DOI: 10.1061/9780784400890.211. URL: https://ascelibrary.org/doi/abs/10.1061/9780784400890.211.

[74] *Ocean Wave Interaction with Ships and Offshore Energy Systems (13.022)*. URL: https://hpc.llnl.gov/training/tutorials/introduction-parallel-computing-tutorial. Last visited on 07/29/2021. Lawrence Livermore National Laboratory.

[75] A. Pétrowski and S. Ben-Hamida. "Evolutionary Algorithms". In: *Evolutionary Algorithms*. John Wiley & Sons, Ltd, 2017. Chap. 1, pp. 1–32. ISBN: 9781119136378. DOI: https://doi.org/10.1002/9781119136378.ch1. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119136378.ch1.

[76] B.T. Polyak. "Newton's method and its use in optimization". In: *European Journal of Operational Research* 181.3 (2007), pp. 1086–1096. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2005.06.076. URL: https://www.sciencedirect.com/science/article/pii/S0377221706001469.

[77] S.S. Rao and Y.F. Fah. *Mechanical vibrations; 5th ed. in SI units*. 5th Edition. Pearson, Sept. 2010.

[78] REN21. *Renewables 2020 Global Status Report*. 2020. URL: https://www.ren21.net/reports/global-status-report/ (visited on 07/29/2021).

[79] J.C. Robledo and V. Sarmiento. "The Relationship between Energy Consumption and GDP: Evidence from a Panel of 10 Latin American Countries". In: *Cuadernos de Economia - Latin American Journal of Economics* 50 (Nov. 2013), pp. 233–255. DOI: 10.7764/LAJE.50.2.233.

[80] M. Rocklin. "Dask: Parallel Computation with Blocked algorithms and Task Scheduling". In: *Proceedings of the 14th Python in Science Conference*. Ed. by Kathryn Huff and James Bergstra. 2015, pp. 130–136.

[81] H.E. Romeijn. "Random search methodsRandom Search Methods". In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Boston, MA: Springer US, 2009, pp. 3245–3251. ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0_556. URL: https://doi.org/10.1007/978-0-387-74759-0_556.

[82] R.Y. Rubinstein and D.P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2004. ISBN: 038721240X.

[83] S. Ruder. *An overview of gradient descent optimization algorithms*. 2016. URL: http://arxiv.org/abs/1609.04747.

[84] D. Rumelhart, G. Hinton, and R. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. DOI: https://doi.org/10.1038/323533a0.

[85] I. Safak, C. Sahin, J.M. Kaihatu, and A. Sheremet. "Modeling wave-mud interaction on the central chenier-plain coast, western Louisiana Shelf, USA". In: *Ocean Modelling* 70 (2013). Ocean Surface Waves, pp. 75–84. ISSN: 1463-5003. DOI: https://doi.org/10.1016/j.ocemod.2012.11.006. URL: https://www.sciencedirect.com/science/article/pii/S1463500312001667.

[86] I. Safak, A. Sheremet, J. Davis, and J.M. Kaihatu. "Nonlinear wave dynamics in the presence of mud-induced dissipation on Atchafalaya Shelf, Louisiana, USA". In: *Coastal Engineering* 130 (2017), pp. 52–64. ISSN: 0378-3839. DOI: https://doi.org/10.1016/j.coastaleng.2017.09.014. URL: https://www.sciencedirect.com/science/article/pii/S0378383917305288.

[87] H. Salehinejad, Julianne Baarbe, Sharan Sankar, J. Barfett, E. Colak, and S. Valaee. "Recent Advances in Recurrent Neural Networks". In: *ArXiv* abs/1801.01078 (2018).

[88] P.D. Sclavounos. *Ocean Wave Interaction with Ships and Offshore Energy Systems (13.022)*. URL: https://ocw.mit.edu/courses/mechanical-engineering/2-24-ocean-wave-interaction-with-ships-and-offshore-energy-systems-13-022-spring-2002/lecture-notes/lecture4.pdf. Last visited on 07/29/2021. MIT OpenCourseWare, 2002.

[89] s.H. Shamsnia. "An Analytical Study on Wave-Current-Mud Interaction". In: *Water* 12 (Oct. 2020), p. 2899. DOI: 10.3390/w12102899.

[90] A. Sheremet, S. Jaramillo, S.-F. Su, M.A. Allison, and K.T. Holland. "Wave-mud interaction over the muddy Atchafalaya subaqueous clinoform, Louisiana, United States: Wave processes". In: *Journal of Geophysical Research: Oceans* 116.C6 (2011). DOI: https://doi.org/10.1029/2010JC006644. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2010JC006644.

[91] T. Shibayama and N.A. Nguyen. "A Visco-Elastic-Plastic Model for Wave-Mud Interaction". In: *Coastal Engineering in Japan* 36.1 (1993), pp. 67–89. DOI: 10.1080/05785634.1993.11924574. URL: https://doi.org/10.1080/05785634.1993.11924574.

[92] D. Simon. *Evolutionary Optimization Algorithms*. Wiley, 2013. ISBN: 9781118659502. URL: https://books.google.com/books?id=gwUwIEPqk30C.

[93] NIST/SEMATECH e-Handbook of Statistical Methods. *Response Surface Designs*. URL: https://www.itl.nist.gov/div898/handbook/pri/section3/pri336.htm (visited on 07/29/2021).

[94] D.I. Stern. "Energy-GDP Relationship". In: *The New Palgrave Dictionary of Economics*. London: Palgrave Macmillan UK, 2018, pp. 3697–3714. ISBN: 978-1-349-95189-5. DOI: 10.1057/978-1-349-95189-5_3015. URL: https://doi.org/10.1057/978-1-349-95189-5_3015.

[95] G. Szeidl and L. Kiss. *Mechanical Vibrations: An Introduction*. June 2020. ISBN: 978-3-030-45073-1. DOI: 10.1007/978-3-030-45074-8.

[96] N. Tsafos. *Energy and Growth: Exploring a Nuanced Relationship*. 2018. URL: https://www.csis.org/analysis/energy-and-growth-exploring-nuanced-relationship (visited on 07/29/2021).

[97] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[98] *Wave Energy Density and Flux*. Feb. 2011. URL: https://wikiwaves.org/Wave_Energy_Density_and_Flux (visited on 07/29/2021).

[99] F.M. White. *Fluid Mechanics*. 7th Edition. New York: McGraw-Hill, Mar. 2011. ISBN: 0071333126.

[100] Wikipedia. *The free encyclopedia*. URL: https://en.wikipedia.org/wiki/Main_Page (visited on 08/12/2021).

[101] C. Windt, J. Tchoufag, and M.-R. Alam. "Numerical Investigation of Three-Dimensional Effects on Wave Excitation Forces on a Submerged Rigid Board". In: *The Second International Conference on Offshore Renewable Energy – CORE2016*. Glasgow, UK, Dec. 2016.

[102] X.-J. Yu and M. Gen. *Introduction to Evolutionary Algorithms*. Springer Publishing Company, Incorporated, 2012. ISBN: 144712569X.

[103] Z.B. Zabinsky. "Random Search Algorithms". In: *Wiley Encyclopedia of Operations Research and Management Science*. American Cancer Society, 2011. ISBN: 9780470400531. DOI: https://doi.org/10.1002/9780470400531.eorms0704. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470400531.eorms0704.

[104] Z. Zainuddin and P. Ong. "Function approximation using artificial neural networks". In: *WSEAS Transactions on Mathematics* 7 (June 2008).

Appendix A

Energy Flux

The energy flux $F(t)$ into the control volume Ω (see Figure 2.2b) is measured by the rate of change of energy $E(t)$. The energy $E(t)$ is in turn the integration of the energy density $e(t)$ over the whole control volume. The energy flux is this given by, according to the transport theorem:

$$\begin{aligned} F(t) = \frac{\mathrm{d}E(t)}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\iiint_{\Omega} e(t)\mathrm{d}V \\ &= \iiint_{\Omega} \frac{\partial e(t)}{\partial t}\mathrm{d}V + \iint_{\partial\Omega} e(t)U_n dS \\ &= \iiint_{\Omega} \frac{\partial e(t)}{\partial t}\mathrm{d}V \end{aligned} \tag{A.1}$$

where, U_n is the normal velocity of surface $\partial\Omega$ outwards of the control volume Ω. In our problem, the control volume Ω is fixed, $\partial\Omega$ is not moving, then $U_n = 0$. The energy flux $F(t)$ reduces to A.1.

The rate of change of energy density is given by:

$$\begin{aligned} \frac{\partial e(t)}{\partial t} &= \frac{\partial}{\partial t}\left(\frac{1}{2}\rho|\mathbf{v}^2| + \rho g h\right) \\ &= \frac{1}{2}\rho\frac{\partial}{\partial t}\left(\nabla\phi\cdot\nabla\phi\right) \\ &= \rho\nabla\cdot\left(\frac{\partial\phi}{\partial t}\nabla\phi\right) - \rho\frac{\partial\phi}{\partial t}\nabla^2\phi \\ &= \rho\nabla\cdot\left(\frac{\partial\phi}{\partial t}\nabla\phi\right) \end{aligned} \tag{A.2}$$

where, the mass continuity gives $\nabla^2\phi = 0$. The energy density rate of change reduces to A.2.

Plugging A.2 into A.1 and applying the divergence theorem gives:

$$\begin{aligned} F(t) &= \iiint_{\Omega} \rho \nabla \cdot \left(\frac{\partial \phi}{\partial t} \nabla \phi \right) \mathrm{d}V \\ &= \rho \iint_{\partial\Omega} \frac{\partial \phi}{\partial t} \nabla \phi \cdot \mathbf{n} \mathrm{d}S \\ &= \rho \iint_{\partial\Omega} \frac{\partial \phi}{\partial t} \phi_n \mathrm{d}S \end{aligned} \tag{A.3}$$

where, $\mathbf{n}$ is the normal unit pointing inward the control boundary, and $\nabla\phi \cdot \mathbf{n} = \dfrac{\partial \phi}{\partial n} = \phi_n$. In our problem $\mathbf{n}$ takes the x-axis direction.

Note that the equation A.3 gives the energy influx into the control volume. The energy outflux has the opposite sign, or the sign can be represented by the normal unit $\mathbf{n}$ [98, 88].

Appendix B

Wave Carpet Area

The wave carpet shape is represented by the Fourier series subject to the constraints in the Equations 4.1–4.8. It is necessary to derive the equations for carpet area and half area to impose the constant area constraint. Starting from the differential area in polar coordinates:

$$\mathrm{d}a = \frac{1}{2} r \mathrm{d}l = \frac{1}{2} r^2 \mathrm{d}\phi$$

Full Area

$$\begin{aligned}
A = \int_0^{2\pi} \frac{1}{2} r^2 \mathrm{d}\theta &= \int_0^{2\pi} \frac{1}{2} \left(r_0 + \sum_{n=1}^{N_F} a_n \sin(n\theta + \phi_n) \right)^2 \mathrm{d}\theta \\
&= \int_0^{2\pi} \frac{1}{2} \Bigg(r_0^2 + \sum_{n=1}^{N_F} a_n^2 \sin^2(n\theta + \phi_n) + 2 \sum_{n=1}^{N_F} r_0 a_n \sin(n\theta + \phi_n) \\
&\qquad + \sum_{\substack{m=1,n=1 \\ m \neq n}}^{N_F} a_m a_n \sin(m\theta + \phi_m) \sin(n\theta + \phi_n) \Bigg) \mathrm{d}\theta \\
&= \frac{1}{2} r_0^2 \cdot 2\pi + \frac{1}{2} \sum_{n=1}^{N_F} a_n^2 \int_0^{2\pi} \sin^2(n\theta + \phi_n) \mathrm{d}\theta + \sum_{n=1}^{N_F} r_0 a_n \int_0^{2\pi} \sin(n\theta + \phi_n) \mathrm{d}\theta \\
&\qquad + \frac{1}{2} \sum_{\substack{m=1,n=1 \\ m \neq n}}^{N_F} a_m a_n \int_0^{2\pi} \sin(m\theta + \phi_m) \sin(n\theta + \phi_n) \mathrm{d}\theta
\end{aligned}$$

The integrals are evaluated to:

$$\int_0^{2\pi} \sin^2(n\theta+\phi_n)\mathrm{d}\theta = \int_0^{2\pi} \frac{1-\cos(2n\theta+2\phi_n)}{2}\mathrm{d}\theta = \pi$$

$$\int_0^{2\pi} \sin(n\theta+\phi_n)\mathrm{d}\theta = 0$$

$$\int_0^{2\pi} \sin(m\theta+\phi_m)\sin(n\theta+\phi_n)\mathrm{d}\theta$$

$$= \int_0^{2\pi} \frac{1}{2}\left(\cos\left(\theta(m-n)+\phi_m-\phi_n\right)-\cos\left(\theta(m+n)+\phi_m+\phi_n\right)\right)\mathrm{d}\theta$$

$$= 0$$

The full area of wave carpet is given by:

$$A = \pi\left(r_0^2 + \frac{1}{2}\sum_{n=1}^{N_F} a_n^2\right) \tag{B.1}$$

Half Area

$$A_h = \int_0^{\pi} \frac{1}{2}r^2\mathrm{d}\theta = \frac{1}{2}r_0^2\cdot\pi + \frac{1}{2}\sum_{n=1}^{N_F} a_n^2 \int_0^{\pi} \sin^2(n\theta+\phi_n)\mathrm{d}\theta + \sum_{n=1}^{N_F} r_0 a_n \int_0^{\pi} \sin(n\theta+\phi_n)\mathrm{d}\theta$$
$$+ \frac{1}{2}\sum_{\substack{m=1,n=1\\ m\neq n}}^{N_F} a_m a_n \int_0^{\pi} \sin(m\theta+\phi_m)\sin(n\theta+\phi_n)\mathrm{d}\theta$$

Evaluating the integrals separately gives:

$$\int_0^\pi \sin^2(n\theta+\phi_n)\mathrm{d}\theta = \int_0^\pi \frac{1-\cos(2n\theta+2\phi_n)}{2}\mathrm{d}\theta = \frac{\pi}{2}$$

$$\int_0^\pi \sin(n\theta+\phi_n)\mathrm{d}\theta = \begin{cases} 0 & \text{if } n \text{ even} \\ \dfrac{2\cos\phi_n}{n} & \text{if } n \text{ odd} \end{cases}$$

$$\int_0^\pi \sin(m\theta+\phi_m)\sin(n\theta+\phi_n)\mathrm{d}\theta$$

$$= \frac{1}{2}\int_0^\pi \cos\left(\theta(m-n)+\phi_m-\phi_n\right)\mathrm{d}\theta - \frac{1}{2}\int_0^\pi \cos\left(\theta(m+n)+\phi_m+\phi_n\right)\mathrm{d}\theta$$

$$= 0$$

$$\frac{1}{2}\int_0^\pi \cos\left(\theta(m-n)+\phi_m-\phi_n\right)\mathrm{d}\theta = \begin{cases} 0 & \text{if } m-n \text{ even} \\ -\dfrac{\sin(\phi_m-\phi_n)}{m-n} & \text{if } m-n \text{ odd} \end{cases}$$

$$\frac{1}{2}\int_0^\pi \cos\left(\theta(m+n)+\phi_m+\phi_n\right)\mathrm{d}\theta = \begin{cases} 0 & \text{if } m+n \text{ even} \\ -\dfrac{\sin(\phi_m+\phi_n)}{m+n} & \text{if } m+n \text{ odd} \end{cases}$$

The half area of wave carpet is given by:

$$A_h = \frac{\pi}{2}r_0^2 + \frac{\pi}{4}\sum_{n=1}^{N_F} a_n^2 + 2r_0 \sum_{\substack{n=1 \\ n \text{ odd}}}^{N_F} \frac{a_n}{n}\cos\phi_n$$
$$+ \frac{1}{2}\sum_{\substack{m=1,n=1 \\ (m+n) \text{ odd}}}^{N_F} a_m a_n \left(\frac{\sin(\phi_m+\phi_n)}{m+n} - \frac{\sin(\phi_m-\phi_n)}{m-n}\right) \tag{B.2}$$

Appendix C

Optimal Shape Parameters

C.1 Linear Waves

Random Search

Table C.1: Random Search-based Optimization Results

Wave Case	r_0	a_1	a_2	a_3	a_4	a_5	p
		ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	β
1F 1D	0.5	-0.0422	-0.2	-0.1240	0.1112	-0.1112	0.5
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
1F MD	0.5	0.0807	-0.1856	-0.0407	0.0144	-0.1860	0.5065
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
MF MD	0.5	-0.2	0.2	-0.2	0.0838	0.0242	2
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
A MF MD	0.5	0.1948	0.1940	0.0872	0.1208	0.0509	1.7072
		0.9627	0.7931	1.4250	0.398	2.0680	2.6361

Neural Network-based Optimization

Table C.2: Neural Network-based Optimization

Wave Case	r_0	a_1	a_2	a_3	a_4	a_5	p
		ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	β
1F 1D	0.5	0.0053	-0.2	-0.2	0.1573	0.2	0.5
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
1F MD	0.5	-0.1921	-0.2	0.2	0.1951	-0.1296	0.5004
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
MF MD	0.5	-0.2	0.2	-0.0798	0.18178	-0.2	1.5
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
A MF MD	0.5	-0.2	-0.1974	0.1869	-0.2	-0.2	2
		1.5112	4.8001	5.2510	5.7102	2.2270	5.7784

C.2 Nonlinear Waves

Cross-Entropy Method

Table C.3: Cross-Entropy Method-based Optimization Results

Wave Case	r_0	a_1	a_2	a_3	a_4	a_5	p
		ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	β
1F 1D	0.5	0.0856	-0.2	-0.2	0.2	0.2	0.5
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
1F MD	0.5	-0.1673	-0.1546	0.2	0.2	-0.1843	0.5048
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
MF 1D	0.5	-0.1893	-0.2	0.2	0.0084	-0.2	0.5
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0

Genetic Algorithm-based Optimization Results

Table C.4: Genetic Algorithm-based Optimization

Wave Case	r_0	a_1	a_2	a_3	a_4	a_5	p
		ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	β
1F 1D	0.5	0.0053	-0.2	-0.2	0.1548	0.2	0.5
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
1F MD	0.5	-0.1924	-0.2	0.2	0.1994	-0.1289	0.5
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0
MF 1D	0.5	-0.1995	-0.2	0.2	0.0998	-0.2	0.5003
		$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	0