IMPLEMENTATION OF A COMPUTATIONAL MODEL FOR RANDOM DIRECTIONAL SEAS AND UNDERWATER ACOUSTICS

by

Cihan Bayındır

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Civil Engineering

Fall 2009

IMPLEMENTATION OF A COMPUTATIONAL MODEL FOR RANDOM DIRECTIONAL SEAS AND UNDERWATER ACOUSTICS

by

Cihan Bayındır

Approved: _____
James T. Kirby, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: _____
Mohsen Badiey, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: _____
Harry W. Shenton, Ph.D.
Chair of the Department of Civil Engineering

Approved: _____
Michael J. Chajes, Ph.D.
Dean of the College of Engineering

Approved: _____
Debra Hess Norris, M.S.
Vice Provost for Graduate and Professional Education

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. James T. Kirby, for his support during last two years I spent at University of Delaware and giving me the opportunity. Without his guidance this thesis would have never been possible. It was an excellent opportunity for me to work under his supervision.

I also thank our colleagues in the College of Earth, Ocean and Environment, especially to Dr. Entin Karjadi, Dr. Aijun Song and to Dr. Mohsen Badiey for technical discussions about the high frequency acoustics and Bellhop program.

I will always gratefully remember efforts of two professors from University of Delaware mathematics department, Dr. David Bellamy and Dr. George Hsiao. I thank them for letting me attend their classes and their guidance.

I also thank my former advisor, Dr. Emre Otay for introducing me the field of coastal engineering and his support during bad times I had.

Structural engineer and my future collaborator Gökcan Karakuş and economist Hakan Bingöl were my best friends during the first two years I lived in US. I thank them for their regular phone calls and encouragement.

Most of all I want to thank my family, my father Ahmet Bayındır, my mother Göksel Bayındır and my sister Dilan Bayındır for their endless support. Their love is the most valuable thing I ever had. I hope this thesis accounts for the missing two years in our lives which we lived far from each other.

This work has been supported by the Office of Naval Research under grant N00013-07-1-0739.

TABLE OF CONTENTS

LIST OF FIGURES v
ABSTRACT vii

Chapter

1 INTRODUCTION 1
2 WAVE MODEL 4
2.1 Jonswap Spectrum and Construction of Initial Wave Field 4
Construction of Initial Wave Field in 2D 5
Construction of Initial Wave Field in 3D 8
2.2 Governing Equations for the Wave Model 11
2.3 Calculation of Surface-Normal Velocity 14
2.4 Numerical Method 16
Aliasing Errors 17
Amplification of Round-Off Errors 18
2.5 2D Linear Wave Model 19
2.6 3D Linear Wave Model 22
2.7 2D Nonlinear Wave Model 25
3 ACOUSTIC SCATTERING PROBLEM 31
3.1 Gaussian Beam Tracing and Bellhop Model 31
3.2 Doppler Shift Calculations 35
4 DISCUSSION OF RESULTS OF COUPLED MODEL 37
4.1 Experimental Data 37
4.2 Model Results 38
5 CONCLUSION 45

Appendix

A MATLAB AND FORTRAN PROGRAMS 47

REFERENCES 111

LIST OF FIGURES

Figure 2.1 A snapshot of spectrum of propagating linear waves in 2D with 1024 components 20

Figure 2.2 Comparisons of input and output spectrums 21

Figure 2.3 A snapshot of propagating linear waves with a principal propagation direction of 0° 22

Figure 2.4 A snapshot of propagating linear waves with a principal propagation direction of 30° 23

Figure 2.5 A snapshot of propagating linear waves with a principal propagation direction of 90° 24

Figure 2.6 A comparison of input and output spectrums in 3D linear model for principal propagation direction of 45° 25

Figure 2.7 Comparison of linear and nonlinear waves 26

Figure 2.8 Comparison of initial spectrum and final spectrums obtained after time evolution of the linear and nonlinear models 27

Figure 2.9 Comparison of probability distribution functions for linear and nonlinear waves 27

Figure 2.10 Wave with two sideband modes example 28

Figure 2.11 Time histories of normalized Fourier amplitudes of first , second and third harmonics 29

Figure 2.12 Time history of change in the energy level 30

Figure 3.1 Doppler effect of an moving object 35

Figure 3.2 Multiple bounce of an acoustic ray 36

Figure 4.1 Eigenrays with different arrival times 38

Figure 4.2 Time histories of arrival times and Doppler shifts for September 24th, 1997, 03.02 am. 39

Figure 4.3 Time histories of arrival times and Doppler shifts for September 24th, 1997, 00.00 am. 40

Figure 4.4 Time histories of arrival times and Doppler shifts obtained from linear and nonlinear simulations for September 24th, 1997, 00.00 am. 41

Figure 4.5 Effect of surface partitioning dx on standard deviation calculations 42

Figure 4.6 Effect of surface partitioning dx on Doppler shift calculations 43

ABSTRACT

Acoustic wave propagation in the ocean is an old and interesting problem. In the literature the problem of interaction of acoustic waves with the ocean surface has drawn the attention of many researchers. This interaction problem is a very complicated problem due to many physical processes involved. Some of the important factors on the ocean surface which affects underwater communications are presence of waves, turbulence generated by wind input, wave breaking, white capping, bubbles and density, salinity and temperature effects.

In this thesis, the effect of linear and nonlinear surface gravity waves on high frequency acoustic propagation is discussed. Realistic models of linear and nonlinear surface gravity waves, which solve the exact governing equations, are created, and these models are coupled with an acoustic Gaussian beam tracing program called Bellhop. Since Bellhop is not capable of accounting for out of plane scattering of acoustic rays, coupling is done only for two spatial dimensions, one horizontal and one vertical.

The wave model provides velocity components in the normal direction to the surface. These velocity components are used in the Doppler frequency shift calculations of acoustic rays generated by Bellhop.

Data from an acoustic experiment are taken as experimental results and coupled wave-acoustic model has been run with the same conditions and same geometric layout of experiments.

Comparisons between experimental results and coupled wave-acoustic model results are presented and the limits of model validity are discussed.

Chapter 1

INTRODUCTION

In the literature, the problem of modeling acoustic sound waves inside a water column has been studied by many researchers. Two widely accepted methods are present in the literature for calculation of intensity of acoustic scattering which applies to acoustic scattering from rough surfaces of different scale. These methods are the Rayleigh-Rice method (Rice, 1951) and Kirchhoff method (Eckart, 1953). The Rayleigh-Rice method is based on the small roughness perturbation approximation and the Kirchhoff method is based on the physical optics approximation (Thorsos, 1987). Since these methods are applicable to rough surfaces with different scales, various attempts have been made to combine these methods and apply combined models to multi-scaled ocean surfaces (Kuryanov, 1963).

A modified high-frequency Kirchhoff approximation was given by McDaniel, who combined Rayleigh-Rice and Kirchhoff methods and produced results which are independent of surface partitioning (McDaniel et al., 1983; McDaniel, 1986). A significant advance is again given by McDaniel who integrated an empirical sea surface model with her high-frequency Kirchhoff model (McDaniel, 1987). In her model surface wave heights and slopes are predicted by an empirical sea surface model as parameters for calculating the intensity of acoustic forward scatter.

Thorsos tested the accuracy and validity of the Rayleigh-Rice and Kirchhoff methods by combining an empirical surface wave model and an acoustic model based on Kirchhoff approximation (Thorsos, 1990). Instead of using surface

statistics such as wave heights and slopes, the Thorsos's model requires actual 1D-horizontal rough surface realizations as inputs.

Dahl conducted two experiments during which acoustic measurements as well as environmental measurements are recorded (Dahl, 2001). His experiments were designed to observe the time and angle spreading characteristics of high frequency sound which is forward scattered from the rough ocean surface. His experimental setup was able to examine high frequency sound waves with one surface interaction only. He also used a Kirchhoff method to interpret his results and obtained simple relations between geometry of acoustical scattering, surface wave conditions and time-angle spreading of the received signals (Dahl, 2001).

Recently, Heitsenrether (2004) coupled an empirical surface wave model with an acoustic Gaussian beam tracing model called Bellhop and compared his results with an experimental data set obtained in Delaware Bay in 1997. His surface realizations were uncorrelated since for each run of Bellhop he was generating different random realizations of the surface. As an extension to Heitsenrether's work, in this thesis we are using a realistic wave model following Dommermuth and Yue (1987) and integrate it with the same acoustic model Bellhop. This work is unique in the sense that there is no previous work present in the literature which couples a realistic wave model and a Gaussian beam tracer.

In Chapter 2 of this thesis construction of initial wave field by Jonswap spectrum and inverse Fourier transform method, governing equations of the wave model and their numerical solution technique will be explained. In two dimensions both linear and nonlinear simulations are presented whereas in three dimensions only linear simulations are presented.

In Chapter 3, acoustic scattering of sound waves from the ocean surface are discussed. The Gaussian beam approach, which is an extension to standard ray tracing theory, is explained. Also, the acoustic Gaussian beam tracing program Bellhop, which is used for modeling sound waves inside the water column, is described. Doppler shift calculations, which are not standard calculations in Bellhop, are explained as well.

In Chapter 4, details of the comparisons of the coupled linear wave-acoustic and coupled nonlinear wave-acoustic models with the available experimental data set are presented. Effect of spatial resolution in the wave model on results is discussed.

Chapter 2

WAVE MODEL

In this thesis, we follow the approach given by Dommermuth and Yue (1987) for modeling nonlinear surface gravity waves. The approach given by Dommermuth and Yue (1987) is a combination of canonical evolution equations and mode coupling idea and it is a direct numerical approach. Nonlinear interactions up to a specified order M in wave steepness can be successfully accounted for by the model provided that a Taylor series expansion of the velocity potential about the mean water level is valid (Dommermuth and Yue, 1987). $M = O(10)$ is adequate for waves up to 80% of Stokes limiting steepness $(ka \leq 0.35)$ (Dommermuth and Yue, 1987).

In order to simulate the experimental conditions we discuss in Section 4.1, we use Jonswap spectrum for construction of initial wave field since Jonswap spectrum provides a relationship between environmental variables, such as wind speed, fetch length, and frequency spectrum. After the construction of the initial wave field specified by the Jonswap spectrum, evolution equations are solved numerically in order to simulate linear and nonlinear waves.

2.1 Jonswap Spectrum and Construction of Initial Wave Field

A frequency spectrum represents the distribution of wave energy across a range of frequencies. In the literature there are numerous different frequency spectrum models. Jonswap is one of these frequency spectrum models and is empirically derived after a wave measurement program (Joint North Sea Wave Project) in the North Sea in 1968 and 1969 (Hasselman et al., 1973). Since experimental data that we use in

evaluating model performance which we will discuss in Chapter 4 is measured in a fetch limited area and we want to compare our results with Heitsenrether (2004), in our model Jonswap spectrum is the most appropriate one to be used when compared to other frequency spectrum models. This model provides frequency spectrum as a function of wind speed under fetch limited conditions (Hasselman et al., 1973) and is in the following form

$$S(\omega) = \alpha g^2 \omega^{-5} \exp\left[-\frac{5}{4}\left(\frac{\omega}{\omega_p}\right)^{-4}\right]\gamma^{\delta} \tag{2. 1}$$

where ω is the angular frequency, g is the gravitational acceleration, δ is a peak enhancement factor given by

$$\delta = \exp\left[-\frac{\left(\omega-\omega_p\right)^2}{2\sigma_o{}^2\omega_p{}^2}\right] \tag{2. 2}$$

and parameters γ and σ_o are given as $\gamma = 3.3$, $\sigma_o = 0.07$ for $\omega \le \omega_p$ and $\sigma_o = 0.09$ for $\omega > \omega_p$. Parameter α is a function of fetch length, X and wind speed, U,

$$\alpha = 0.076\left(\frac{gX}{U}\right)^{-0.22} \tag{2. 3}$$

and peak frequency is given by

$$\omega_p = 7\pi\left(\frac{g}{U}\right)\left(\frac{gX}{U^2}\right)^{-0.33} \tag{2. 4}$$

Construction of Initial Wave Field in 2D

Once the frequency spectrum is obtained by Equations (2.1)-(2.4), a wavenumber spectrum can be obtained from the frequency spectrum considering the energy equality under both curves which leads to

$$S(\omega)\frac{d\omega}{dk} = S_k(k) \tag{2. 5}$$

where S_k is the wavenumber spectrum. The relationship between ω and k is given by dispersion relationship

$$\omega^2 = gk\tanh(kh) \tag{2. 6}$$

from which the expression for group velocity can be derived

$$\frac{d\omega}{dk} = C_g = \frac{1}{2}\left(1+\frac{2kh}{\sinh(2kh)}\right)\frac{\omega}{k} \tag{2. 7}$$

Therefore, using Equations (2.5) and (2.7), wavenumber spectrum can be obtained.

Once the wavenumber spectrum is obtained, nodal amplitudes for each dk interval can be obtained by energy equality

$$\frac{1}{2}a_r{}^2 = S_k(k_r)dk \tag{2. 8}$$

where $r = 0,1,...,\frac{N}{2}$, $k_r = rdk$, $dk = \frac{2\pi}{L}$ and L is the periodic domain length. N is the total number of wave components and chosen to be a power of 2 in order to make use of Fast Fourier Transforms. Then two sided amplitude spectrum is constructed using one sided amplitude spectrum by the symmetry relation

$$a_s = a_{N-s} \tag{2. 9}$$

where $s = \frac{N}{2}+1,\frac{N}{2}+2,...,N-1$.

For $r = 0,1,...,\frac{N}{2}$ we generate uniformly distributed random phases Θ_r with values in the interval of $[0,2\pi]$. By using the symmetry relation given by

Equation (2.9) not for amplitudes but for random phases this time we obtain the symmetric random phases Θ_j for $j = -\frac{N}{2}+1, -\frac{N}{2}+2, ..., \frac{N}{2}$.

Complex amplitudes, A, are obtained by using real amplitudes obtained by Equations (2.8) and (2.9) according to

$$A_j = \frac{a_j \exp(i\Theta_j)}{2} \tag{2. 10}$$

where $j = -\frac{N}{2}+1, -\frac{N}{2}+2, ..., \frac{N}{2}$, $i = \sqrt{-1}$ and Θ_j denotes the symmetric uniformly distributed random phase shifts in the interval of $[0, 2\pi]$. Then, based on these complex amplitudes, initial water surface elevation can be obtained by

$$\eta_p = \sum_{j=-N/2+1}^{N/2} A_j \exp\left(ik_j x_p\right) \tag{2. 11}$$

where $p = 0, 1, ..., N-1$, $x_p = pdx$, $dx = \frac{L}{N}$ and L is the periodic domain length.

Similarly, for the complex amplitudes which we use in the construction of surface velocity potential, ϕ^s, we have

$$B_j = -\frac{ig}{\omega_j}\frac{a_j}{2}\exp(i\Theta_j) \tag{2. 12}$$

where ω_j are the angular frequencies obtained from Equation (2.6) for discrete wavenumbers, k_j, and g is the gravitational acceleration. Then initial surface velocity potential can be obtained by

$$\phi^s{}_p = \sum_{j=-N/2+1}^{N/2} B_j \exp\left(ik_j x_p\right) \tag{2. 13}$$

for $p = 0,1,...,N-1$. Note that in order to get real values for η and ϕ^s complex amplitudes, $A(k)$ and $B(k)$ have to be complex conjugates of $A(-k)$ and $B(-k)$. Computations of Equations (2.11) and (2.13) are done by making use of Inverse Fast Fourier Transforms in order to get η and ϕ^s.

Construction of Initial Wave Field in 3D

In 3D, since we are dealing with two horizontal dimensions, the wave field can not be described by frequency spectrum solely. We also need to consider directional spreading of waves and work in terms directional wave spectra. Directional wave spectra $S(k,\theta)$ can be described by

$$S(k,\theta) = S_k(k)G(\theta) \tag{2. 14}$$

where S_k is the wavenumber spectrum described by Equation (2.5) and $G(\theta)$ is the directional spreading function. Following Donelan et. al. (1985), we use the following form for the directional spreading function

$$G(\theta) = \frac{\beta}{2} sech^2\left(\beta\left(\theta - \bar{\theta}\right)\right) \tag{2. 15}$$

where $\bar{\theta}$ is the mean wave direction and

$$\beta = \begin{cases} 2.61\left(\dfrac{\omega}{\omega_p}\right)^{1.3}; & 0.56 < \dfrac{\omega}{\omega_p} < 0.95, \\ 2.28\left(\dfrac{\omega}{\omega_p}\right)^{-1.3}; & 0.95 < \dfrac{\omega}{\omega_p} < 1.6, \\ 1.24; & \textit{otherwise.} \end{cases} \tag{2. 16}$$

The directional wave spectra $S(k,\theta)$ can be obtained using Equations (2.14), (2.15) and (2.16). We will discuss derivation of component amplitudes shortly. In order to apply inverse Fourier transform method to these amplitudes to construct an initial water surface, we need to transform the directional wave spectrum, which is a function of wavenumber, k, and direction, θ, to a wavenumber spectrum as a function of k_x and k_y, $S_{k_xk_y}(k_x,k_y)$, where k_x and k_y are wavenumber components along x and y axes, respectively. Considering energy equality under surfaces of $S(k,\theta)$ and $S_{k_xk_y}(k_x,k_y)$, the transformation is done according to

$$S_{k_xk_y}(k_x,k_y)=\frac{S(k,\theta)}{|J|} \tag{2. 17}$$

where $|J|$ is the Jacobian of the transformation and is given by

$$|J|=\begin{vmatrix}\cos\theta & \sin\theta\\ -k\sin\theta & k\cos\theta\end{vmatrix}=k \tag{2. 18}$$

Once the wavenumber spectrum $S_{k_xk_y}(k_x,k_y)$ is obtained, component amplitudes for each dk_xdk_y interval can be obtained by energy equality

$$\frac{1}{2}a_{p,r}{}^2=S_{k_xk_y}({}_pk_x,{}_rk_y)dk_xdk_y \tag{2. 19}$$

where $p=0,1,...,\frac{N_x}{2}$, $r=0,1,...,\frac{N_y}{2}$, ${}_pk_x=pdk_x$ ${}_rk_y=rdk_y$, $dk_x=\frac{2\pi}{L_x}$ and $dk_y=\frac{2\pi}{L_y}$. N_x is the total number of wave components in x direction, N_y is the total number of wave components in y direction and L_x, L_y are lengths of periodic domain in x and y directions, respectively. Both N_x and N_y are chosen to be a power of 2 in order to make use of 2D Fast Fourier Transforms. Then two sided

amplitude spectrum is constructed using one sided amplitude spectrum by the symmetry relation

$$a_{s,u} = a_{N_x - s, N_y - u} \tag{2. 20}$$

where $s = \frac{N_x}{2} + 1, \frac{N_x}{2} + 2, ..., N_x - 1$ and $u = \frac{N_y}{2} + 1, \frac{N_y}{2} + 2, ..., N_y - 1$.

For $p = 0, 1, ..., \frac{N_x}{2}$ and $r = 0, 1, ..., \frac{N_y}{2}$ we generate uniformly distributed random phases $\Theta_{p,r}$ with values in the interval of $[0, 2\pi]$. By using the symmetry relation given by Equation (2.20) not for amplitudes but for random phases this time we obtain the symmetric random phases $\Theta_{j,n}$ for $j = -\frac{N_x}{2} + 1, -\frac{N_x}{2} + 1, ..., \frac{N_x}{2}$ and $n = -\frac{N_y}{2} + 1, -\frac{N_y}{2} + 1, ..., \frac{N_y}{2}$.

Complex amplitudes, A, are obtained by using real amplitudes obtained by Equations (2.19) and (2.20) and making use of a uniformly distributed random number generator with numbers generated in the interval of $[0, 2\pi]$ in order to get a random representation of the surface by using

$$A_{j,n} = \frac{a_{j,n}}{2} \exp(i\Theta_{j,n}) \tag{2. 21}$$

for $j = -\frac{N_x}{2} + 1, -\frac{N_x}{2} + 1, ..., \frac{N_x}{2}$, $n = -\frac{N_y}{2} + 1, -\frac{N_y}{2} + 1, ..., \frac{N_y}{2}$. Then, based on these complex amplitudes, initial water surface elevation can be obtained by

$$\eta_{v,q} = \sum_{j=-N_x/2+1}^{N_x/2} \sum_{n=-N_y/2+1}^{N_y/2} A_{j,n} \exp\left(ik_j x_v + ik_n y_q\right) \tag{2. 22}$$

where $v = 0, 1, ..., N_x - 1$, $q = 0, 1, ..., N_y - 1$, $x_v = vdx$, $y_q = qdy$, $dx = \frac{L_x}{N_x}$, $dy = \frac{L_y}{N_y}$.

Similarly for the complex amplitudes which we use in the construction of surface velocity potential, ϕ^s, we have

$$B_{j,n} = -\frac{ig}{\omega_{j,n}} \frac{a_{j,n}}{2} \exp(i\Theta_{j,n}) \tag{2. 23}$$

Then initial surface velocity potential can be obtained by

$$\phi^s{}_{v,q} = \sum_{j=-N_x/2+1}^{N_x/2} \sum_{n=-N_y/2+1}^{N_y/2} B_{j,n} \exp\left(ik_j x_v + ik_n y_q\right) \tag{2. 24}$$

where $v = 0,1,...,N_x - 1$ and $q = 0,1,...,N_y - 1$. Note that in order to get real values for η and ϕ^s complex amplitudes, $A(k_x, k_y)$ and $B(k_x, k_y)$, have to be complex conjugates of $A(-k_x, -k_y)$ and $B(-k_x, -k_y)$. Computations of Equations (2.22) and (2.24) are done by making use of 2D Inverse Fast Fourier Transforms in order to get η and ϕ^s.

2.2 Governing Equations for the Wave Model

We begin the derivation of the evolution equations with the classical boundary value problem given for water waves in constant depth h. We assume that flow is inviscid, incompressible and irrotational. Therefore a velocity potential $\phi(\vec{x}, z, t)$ exists such that ϕ satisfies the governing equation for flow which is Laplace's equation. $\vec{x} = (x, y)$ denotes the horizontal position vector. Following Zakharov (1968), we use the surface velocity potential

$$\phi^S(\vec{x}, t) = \phi(\vec{x}, \eta(\vec{x}, t), t) \tag{2. 25}$$

at $z = \eta(\vec{x}, t)$. η denotes the water surface fluctuation from the still water level. After chain rule differentiation, classical kinematic and dynamic boundary conditions can be expressed in terms of ϕ^S by

$$\eta_t + \nabla_h \phi^S . \nabla_h \eta - (1 + \nabla_h \eta . \nabla_h \eta) \phi_z(\vec{x}, \eta, t) = 0 \tag{2. 26}$$

$$\phi^S{}_t + g\eta + \frac{1}{2} \nabla_h \phi^S . \nabla_h \phi^S - \frac{1}{2} (1 + \nabla_h \eta . \nabla_h \eta) \phi_z{}^2(\vec{x}, \eta, t) = -\frac{P_a}{\rho} \tag{2. 27}$$

where $\nabla_h = \left(\partial / \partial x, \partial / \partial y \right)$ is the horizontal gradient, P_a is the atmospheric pressure and ρ is the density of the fluid.

Expressing ϕ in terms of perturbation series, we get

$$\phi(\vec{x}, z, t) = \sum_{m=1}^{M} \phi^{(m)}(\vec{x}, z, t) \tag{2. 28}$$

where $\phi^{(m)}$ is a quantity in the order of ε^m and perturbation parameter, ε, is presumed to be small in wave steepness. Using Equation (2.25) and Equation (2.28) and carrying all $\phi^{(m)}$s to a known point $(z = 0)$ by Taylor series expansion we get

$$\phi^S(\vec{x}, t) = \phi(\vec{x}, \eta, t) = \sum_{m=1}^{M} \sum_{k=0}^{M-m} \frac{\eta^k}{k!} \frac{\partial^k}{\partial z^k} \phi^{(m)}(\vec{x}, 0, t) \tag{2. 29}$$

Expanding Equation (2.29) and collecting terms at each order, we obtain

$$\phi^{(m)}(\vec{x}, 0, t) = T^{(m)} \tag{2. 30}$$

$$T^{(1)} = \phi^S \tag{2. 31}$$

$$T^{(m)} = -\sum_{k=1}^{m-1} \frac{\eta^k}{k!} \frac{\partial^k}{\partial z^k} \phi^{(m-k)}(\vec{x}, 0, t) \tag{2. 32}$$

where $m = 2, ..., M$. M is the arbitrary order in wave steepness at which the expansion terminated and, following Dommemuth and Yue (1987), is specified to be 10. When the Laplace's equation as well as other boundary conditions, such as bottom boundary condition, is considered with Equations (2.30), (2.31) and (2.32) give a sequence of

boundary value problems. These boundary value problems, like any other perturbation expansion, can be solved successively with increasing m .

Additionally $\phi^{(m)}$ can be represented with eigenfunction expansion with coefficients which can vary with time. Thus we write;

$$\phi^{(m)}(\vec{x},z,t)=\sum_{n=1}^{\infty}\phi_n{}^{(m)}(t)\Psi_n(\vec{x},z) \qquad (2.\ 33)$$

where Ψ_n s are the eigenfunctions which have different forms in deep and shallow water and are given by Equations (2.40) and (2.41). If we substitute Equation (2.33) into Equation (2.29) we get

$$\phi_z(\vec{x},\eta,t)=\sum_{m=1}^{M}\sum_{k=0}^{M-m}\frac{\eta^k}{k!}\sum_{n=1}^{N}\phi_n{}^{(m)}(t)\frac{\partial^{k+1}}{\partial z^{k+1}}\Psi_n(\vec{x},0) \qquad (2.\ 34)$$

And finally if we substitute Equation (2.34) into Equations (2.26) and (2.27) we get

$$\eta_t+\nabla_h\phi^S.\nabla_h\eta-(1+\nabla_h\eta.\nabla_h\eta)\left[\sum_{m=1}^{M}\sum_{k=0}^{M-m}\frac{\eta^k}{k!}\sum_{n=1}^{N}\phi_n{}^{(m)}(t)\frac{\partial^{k+1}}{\partial z^{k+1}}\Psi_n(\vec{x},0)\right]=0 \quad (2.\ 35)$$

$$\phi^S{}_t+g\eta+\frac{1}{2}\nabla_h\phi^S.\nabla_h\phi^S-\frac{1}{2}(1+\nabla_h\eta.\nabla_h\eta)\left[\sum_{m=1}^{M}\sum_{k=0}^{M-m}\frac{\eta^k}{k!}\sum_{n=1}^{N}\phi_n{}^{(m)}(t)\frac{\partial^{k+1}}{\partial z^{k+1}}\Psi_n(\vec{x},0)\right]^2=-\frac{P_a}{\rho} \qquad (2.\ 36)$$

Equations (2.35) and (2.36) are the evolution equations which are solved numerically. Gage pressure, P_a, in Equation (2.36) is taken to be zero.

As an alternative, if we do not perform the Stoke's expansion and avoid Taylor series expansion, Ψ_n's are expressed at free surface and eigenfunction expansion of ϕ^S becomes

$$\phi(\vec{x},\eta,t)=\sum_{n=1}^{N}\phi_n(t)\Psi_n(\vec{x},\eta)=\phi^S(\vec{x},t) \tag{2. 37}$$

If we plug Equation (2.37) into Equations (2.26) and (2.27) we get

$$\eta_t+\nabla_h\phi^S.\nabla_h\eta-(1+\nabla_h\eta.\nabla_h\eta)\sum_{n=1}^{N}\phi_n(t)\frac{\partial}{\partial z}\Psi_n(\vec{x},\eta)=0 \tag{2. 38}$$

$$\phi^S{}_t+g\eta+\frac{1}{2}\nabla_h\phi^S.\nabla_h\phi^S-\frac{1}{2}(1+\nabla_h\eta.\nabla_h\eta)\left[\sum_{n=1}^{N}\phi_n(t)\frac{\partial}{\partial z}\Psi_n(\vec{x},\eta)\right]^2=-\frac{P_a}{\rho} \tag{2. 39}$$

Equations (2.38) and (2.39) are alternative evolution equations for the problem. But it is useful to note that, since eigenfunctions are evaluated at free surface, ordinary fast and inverse fast Fourier-transform technique can not be applied to Equations (2.38) and (2.39). Rienecker and Fenton (1981) and Bryant (1983) studied these equations in their papers.

For deep water, Equation (2.33) can be readily written as

$$\phi^{(m)}(\vec{x},z,t)=\sum_{n=0}^{\infty}\phi_n{}^{(m)}(t)\exp(\left|\overrightarrow{K_n}\right|z+i\overrightarrow{K_n}\bullet\vec{x}) \tag{2. 40}$$

and for arbitrary depth, Equation (2.33) becomes

$$\phi^{(m)}(\vec{x},z,t)=\sum_{n=0}^{\infty}\phi_n{}^{(m)}(t)\frac{\cosh\left[\left|\overrightarrow{K_n}\right|(z+h)\right]}{\cosh\left[\left|\overrightarrow{K_n}\right|(h)\right]}\exp(i\overrightarrow{K_n}\bullet\vec{x}) \tag{2. 41}$$

where $\overrightarrow{K_n}=(k_x,k_y)$ denotes the wavenumber vector. All calculations in the following sections are based on the evolution Equations (2.35) and (2.36).

2.3 Calculation of Surface-Normal Velocity

In Doppler shift calculations, which we discuss in Chapter 3, the velocity component in the direction normal to the surface turned out to be an important factor.

Using the wave model which we have already discussed we can obtain surface-normal velocity.

Knowing surface velocity potential ϕ^s as a function of spatial variables x and y at an instant of time, the velocity field at the surface in 3D is given by

$$\vec{u} = (\phi_x^s, \phi_y^s, \phi_z\big|_{z=\eta}) \tag{2. 42}$$

Path of a particle at the water surface can be described with the function

$$F = z - \eta \tag{2. 43}$$

Therefore outward normal to the water surface is given by

$$\vec{n} = \left(\frac{F_x, F_y, F_z}{\sqrt{F_x^{\,2} + F_y^{\,2} + F_z^{\,2}}} \right) = \left(\frac{-\eta_x, -\eta_y, 1}{\sqrt{\eta_x^{\,2} + \eta_y^{\,2} + 1}} \right) \tag{2. 44}$$

Then surface-normal velocity, v, can be specified as

$$v = \vec{u}.\vec{n} = (\phi_x^s, \phi_y^s, \phi_z\big|_{z=\eta}).\left(\frac{-\eta_x, -\eta_y, 1}{\sqrt{\eta_x^{\,2} + \eta_y^{\,2} + 1}} \right) = \left(\frac{-\eta_x \phi_x^s, -\eta_y \phi_y^s, \phi_z\big|_{z=\eta}}{\sqrt{\eta_x^{\,2} + \eta_y^{\,2} + 1}} \right) \tag{2. 45}$$

where $\phi_z\big|_{z=\eta}$ is calculated by using Equation (2.34), ϕ_x^s and ϕ_y^s are calculated by a pseudospectral approach by utilizing fast-Fourier and inverse fast-Fourier transforms between physical space and wavenumber space. In 2D Equation (2.45) reduces to

$$v = \vec{u}.\vec{n} = \left(\frac{-\eta_x \phi_x^s, \phi_z\big|_{z=\eta}}{\sqrt{\eta_x^{\,2} + 1}} \right) \tag{2. 46}$$

Equation (2.46) is used in wave model in order to get the surface-normal velocity which is an important factor in the Doppler shift calculations of the acoustic rays.

2.4 Numerical Method

Following Dommermuth and Yue (1987) we use a two-step procedure for solving evolution Equations (2.35) and (2.36).

First, all spatial derivatives in the evolution Equations (2.35), (2.36) and Equation (2.46) are evaluated by a pseudospectral approach by utilizing fast-Fourier and inverse fast-Fourier transforms between physical space and wavenumber space. All nonlinear products in the evolution Equations (2.35), (2.36) and Equation (2.46) are calculated in physical space.

Secondly, starting with given initial conditions for water surface fluctuation, η, and surface velocity potential, ϕ^S, time integration is done with a fourth order Runge-Kutta time integrator. Starting from initial conditions, this two-step procedure is repeated for every time step. For nonlinear simulations we give the flow field enough time to not to suffer from development of high-frequency standing waves due to linear initial conditions (Dommermuth, 1999).

Our numerical scheme has errors due to numerical time integration, truncation in the number of free wave modes, N, and truncation in the arbitrary order of wave steepness, M. Also, aliasing errors and amplification of the round-off errors are present in our numerical scheme which we discuss in detail in the following sections.

Aliasing Errors

In order to get alias free products we are using the technique of aliasing removal by padding (Canuto et al., 2006). Key of padding technique is to use T points for discrete transform rather than N points where $T \geq \frac{3N}{2}$.

Consider the product $s(x)$ of two functions $u(x)$ and $v(x)$ which have Fourier series representations:

$$s(x) = \sum_{n=-T/2+1}^{T/2} s_n \exp(inx) \tag{2. 47}$$

$$u(x) = \sum_{j=-T/2+1}^{T/2} u_j \exp(ijx) \tag{2. 48}$$

$$v(x) = \sum_{r=-T/2+1}^{T/2} v_r \exp(irx) \tag{2. 49}$$

Performing multiplication of $u(x)$ and $v(x)$ in physical space for $s(x)$ gives

$$s_n = \sum_{j+k=n} u_j v_k + \sum_{j+k=n\pm T} u_j v_k \tag{2. 50}$$

The second sum represents aliasing error terms. For $j = 0,1,...,T-1$ we define new coefficients

$$\bar{u}_j = \begin{cases} u_j, & -\frac{N}{2}+1 \leq j \leq \frac{N}{2} \\ 0, & otherwise \end{cases} \tag{2. 51}$$

and

$$\bar{v}_j = \begin{cases} v_j, & -\frac{N}{2}+1 \leq j \leq \frac{N}{2} \\ 0, & otherwise \end{cases} \tag{2. 52}$$

Multiplication of these new functions in physical space give new product , $\overline{s}(x)$,

$$\overline{s}(x) = \overline{u}(x)\overline{v}(x) \tag{2. 53}$$

So we obtain de-aliased coefficients by

$$s_n = \overline{s}_n \text{ for } n = -\frac{N}{2}+1,...,\frac{N}{2} \tag{2. 54}$$

from which the dealiased product, $s(x)$, can be constructed by Equation (2.47).

For products involving more than two terms, such as $s(x) = u(x)^p$, where p is an arbitrary power, the approach followed by Dommermuth and Yue (1987) is successive multiplication where each factor is made alias-free before multiplying by the next term. The anti-aliasing algorithm we use in evaluation of these terms is using T points for discrete transform rather than N points where $T \geq \frac{3pN}{4}$ points.

Performing multiplication $u(x)^p$ in physical space for $s(x)$ gives

$$s_n = \sum_{j+k+...+l=n} u_j u_k ... u_l + \sum_{j+k+...+l=n\pm T} u_j u_k ... u_l \tag{2. 55}$$

For $j = 0,1,...,T-1$ we define new coefficients by Equations (2.51) and we obtain de-aliased products by Equations (2.53), (2.54) and (2.47).

Amplification of Round-Off Errors

Consider a small random error δ_{mn} in the amplitude $\phi_n^{(m)}$ which leads to $\phi_n^{(m)} = \hat{\phi}_n^{(m)}(1+\delta_{mn})$ where $\hat{\phi}_n^{(m)}$ denotes the exact values (Dommermuth and Yue, 1987). Using Equations (2.32) and (2.33), we get

$$T^{(m)} - \hat{T}^{(m)} = -\sum_{k=1}^{m-1} \frac{\eta^k}{k!} \sum_{n=1}^{N} \hat{\phi}_n^{(m-k)} \delta_{(m-k),n} \frac{\partial^k}{\partial z^k} \psi_n \tag{2. 56}$$

In general $\left|\frac{\partial^k}{\partial z^k}\psi_n\right| \approx |K_n|^k$ where $|K_n| \approx n$ so for any m, error in the highest wavenumber modes is the most amplified. One way of eliminating such kind of instability is applying a smoothing function. Following Longuet-Higgins and Cokelet (1976) we apply a smoothing function to η and ϕ^S in wavenumber space in the form

$$\Lambda(K_n) = \frac{1}{8}\left[5 + 4\cos\left(\frac{\pi|K_n|}{|K_N|}\right) - \cos\left(\frac{2\pi|K_n|}{|K_N|}\right)\right] \tag{2. 57}$$

where $|K_N|$ is the wavenumber at the Nyquist frequency.

2.5 2D Linear Wave Model

If we linearize evolution Equations (2.35) and (2.36), we end up with linear evolution equations at $z = 0$ in the following form

$$\eta_t - \phi^S{}_z = 0 \tag{2. 58}$$

$$\phi^S{}_t + g\eta = 0 \tag{2. 59}$$

Starting with initial wave field in 2D, η and ϕ^S, linear Equations (2.58) and (2.59) are solved by the Runge-Kutta method discussed above.

An snapshot of propagating linear waves can be seen in Figure 2.1 where we plot water surface fluctuation, η, as a function of x.

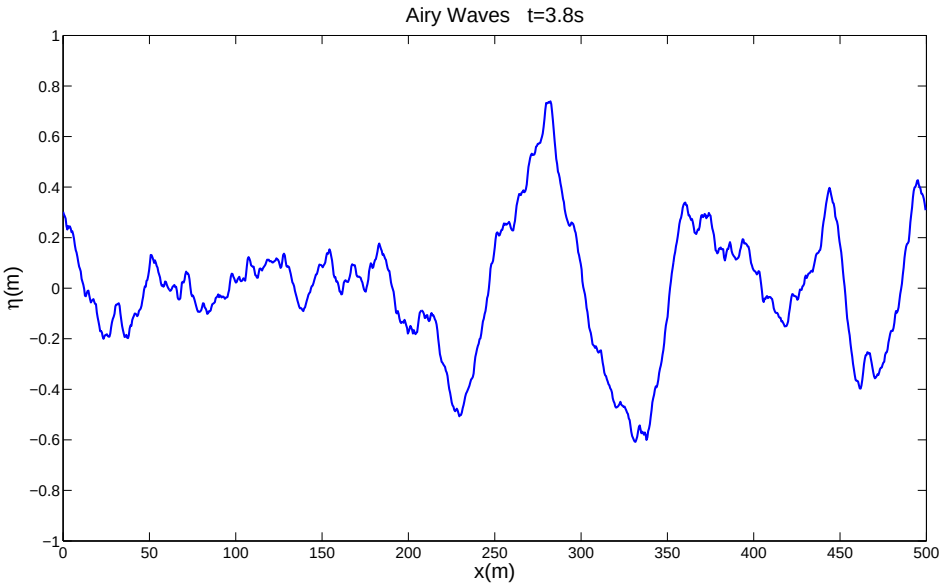

Figure 2. 1: A snapshot of propagating linear waves in 2D with 1024 components

In order to test the accuracy of the model we apply two accuracy checks. As a first check we compare the initial spectrum, which is used for construction of the initial wave field, with the spectrum we obtain by Fourier analysis after the model is marched in time. This comparison can be seen in Figure 2.2. Perfect agreement in Figure 2.2 indicates that linear model conserves the input energy. This is expected because linear evolution equations, Equations (2.58) and (2.59) do not suffer from amplification of round-off errors or aliasing errors. As a second check we compare the input and output variances. Initial variance is given by

$$\sigma^{2}_{initial} = \int_{0}^{\infty} S(\omega) d\omega \qquad (2.60)$$

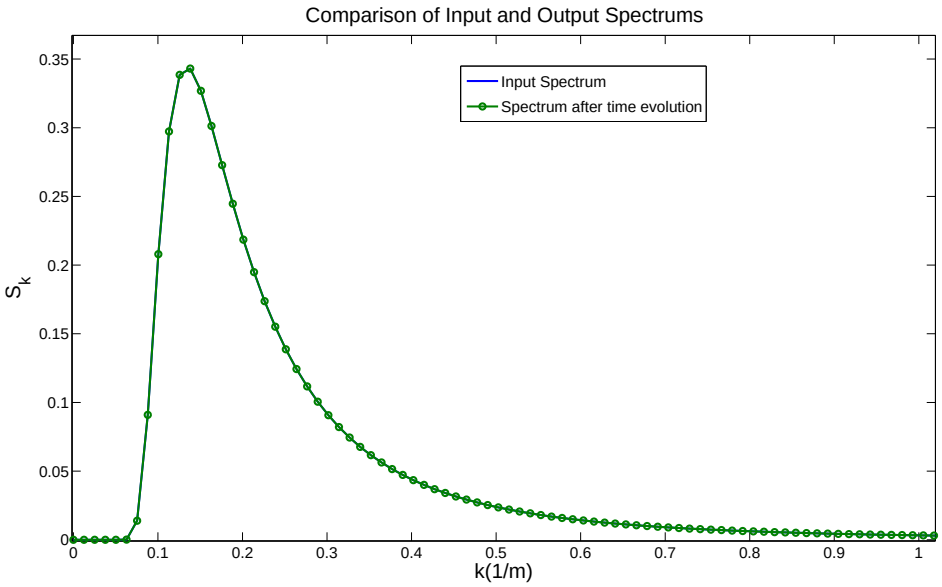

Figure 2. 2: A comparison of input and output spectrums

We calculate final variance of water surface fluctuation by

$$\sigma^{2}{}_{final} = \frac{1}{N}\sum_{j=1}^{N}(\eta_j - \overline{\eta})^2 \tag{2. 61}$$

where

$$\overline{\eta} = \frac{1}{N}\sum_{j=1}^{N}\eta_j \approx 0 \tag{2. 62}$$

For example for the frequency spectrum presented in the Figure 2.2, Equations (2.60) and (2.61) both lead to a variance value of $0.0624m^2$ which is another indicator of energy conservation in the model. This test is repeated for different frequency spectrums and same excellent agreement is obtained for every spectrum within an error in the order of $10^{-4}m^2$. As N gets bigger, the wave field is represented better; therefore we get closer agreement in the variances.

2.6 3D Linear Wave Model

In 3D linear wave model again linearized evolution equations, Equations (2.58) and (2.59), are solved starting with the initial water surface fluctuation and surface velocity potential, η and ϕ^S, described in Section 2.1. In the calculation of z derivatives of ϕ^S in Equation (2.58) 2D Fast Fourier Transforms are used.

Three snapshots of the 3D wave model are presented in Figures 2.3, 2.4 and 2.5 with mean wave directions of 0°, 45° and 90° defined from x axis in counter clockwise sense, respectively.

In our 3D wave model, the mean wave direction can be specified as arbitrarily.

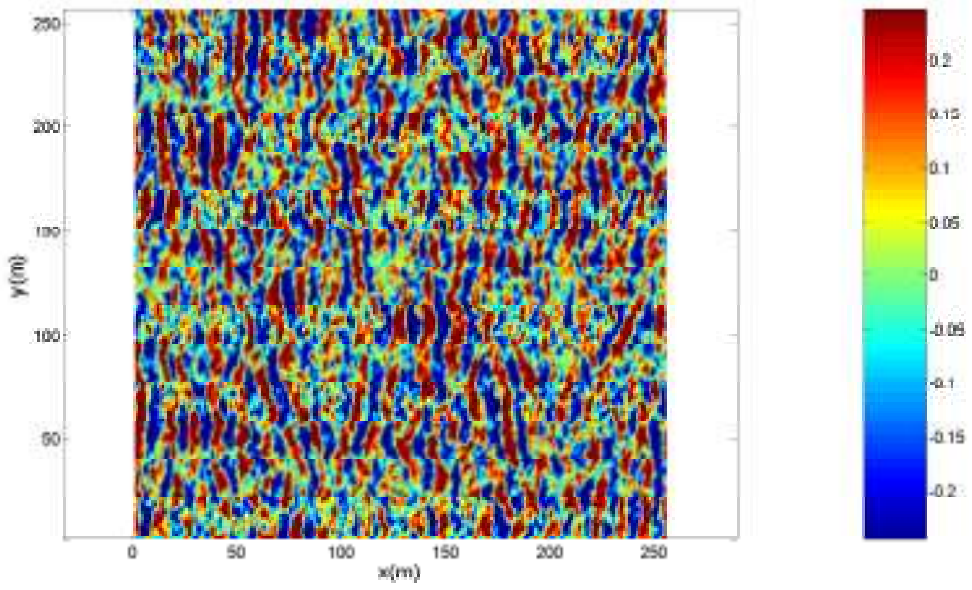

Figure 2. 3: A snapshot of propagating linear waves with a principal propagation direction of 0°

In the 3D model we are doing two accuracy checks. One of the accuracy checks is again comparison of input and output directional wavenumber spectrums. Input spectrum $S_{k_x k_y}(k_x, k_y)$ is specified by the Equation (2.17) whereas output

spectrum is obtained by 2D Fast Fourier Transforms after model has been run in time. Comparison of results is presented in Figure 2.6 and they agree well.

Second accuracy check is the comparison of initial and final variances. Initial variance under spectrum is given by

$$\sigma^2_{initial} = \int_{-\pi}^{\pi}\int_{0}^{\infty} S(\omega,\theta)d\omega d\theta = \int_{-\pi}^{\pi} G(\theta)d\theta \int_{0}^{\infty} S(\omega)d\omega \qquad (2.\ 63)$$

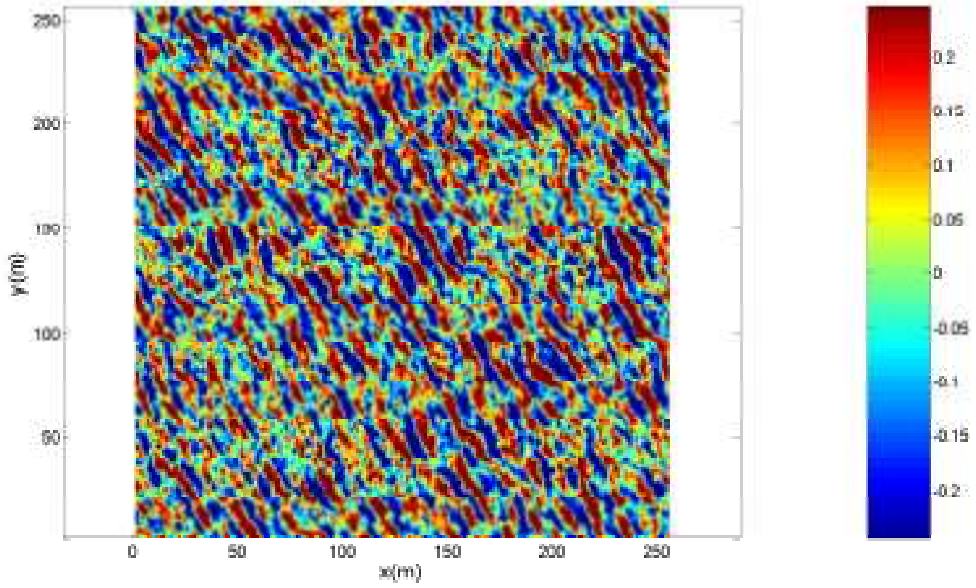

Figure 2. 4: A snapshot of propagating linear waves with a principal propagation direction of 30°

Since $\int_{-\pi}^{\pi} G(\theta)d\theta = 1$, Equation (2.63) reduces to

$$\sigma^2_{initial} = \int_{0}^{\infty} S(\omega)d\omega \qquad (2.\ 64)$$

And we are calculating variances of time series of η at different points on the rectangular grid by expression

$$\sigma^{2}_{final} = \frac{1}{T}\sum_{j=1}^{T}(\eta_j - \bar{\eta})^2 \qquad (2.\ 65)$$

where T denotes the length of time series data and

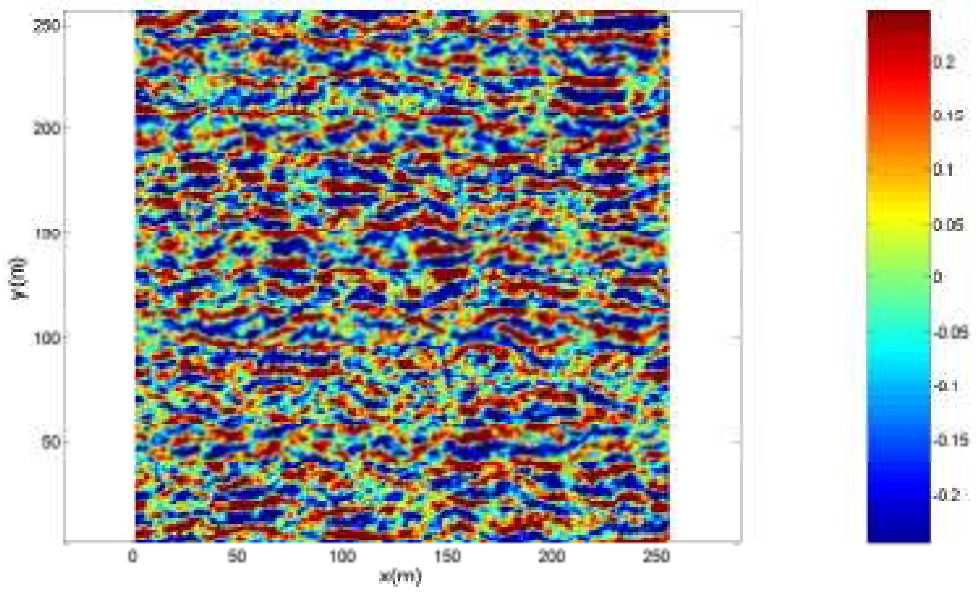

Figure 2. 5: A snapshot of propagating linear waves with a principal propagation direction of 90°

$$\bar{\eta} = \frac{1}{T}\sum_{j=1}^{T}\eta_j \approx 0 \qquad (2.\ 66)$$

For the spectrum given in Figure 2.6, Equation (2.64) leads to an initial variance of $0.0453m^2$. We calculate the final variances at four different points of the grid by Equation (2.65) and averaging those variances lead to $0.0434m^2$, $0.0445m^2$, $0.0447m^2$ and $0.0450m^2$ for simulation times $50T_p$, $100T_p$, $150T_p$ and $200T_p$, where T_p denotes the peak wave period. As simulation time gets bigger and statistics become more stable, variances converge to the value of initial variance as expected.

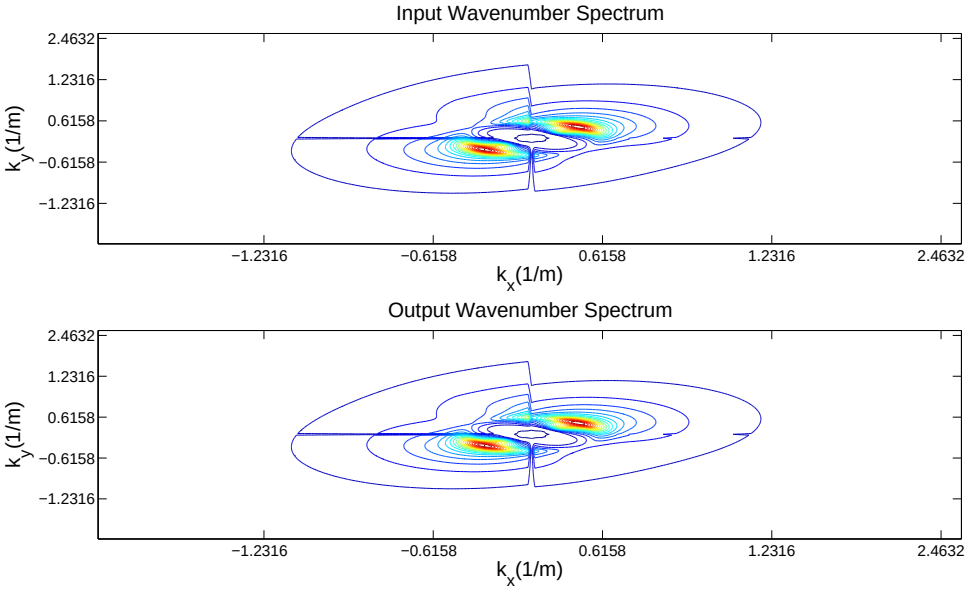

Figure 2. 6: A comparison of input and output spectrums in 3D linear model for principal propagation direction of 45°

2.7 2D Nonlinear Wave Model

In 2D nonlinear model we are solving evolution Equations (2.35) and (2.36). In this case horizontal gradient refers to gradient in the x direction only since it is the only horizontal dimension.

A comparison of linear and nonlinear waves with 1024 components is given in Figure 2.7. In Figure 2.7 nonlinear waves exhibits a peaked behavior in the crests and a flatter behavior in the troughs when compared to linear waves. This is a classical property of nonlinear systems.

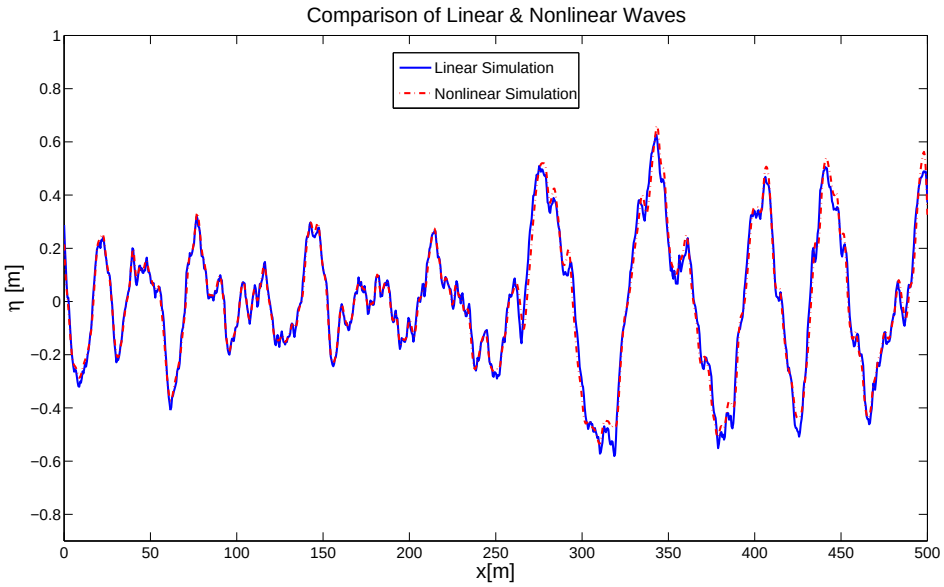

Figure 2. 7: Comparison of linear and nonlinear waves (continuous line refers to linear simulation; dash dot refers to nonlinear simulation)

In Figure 2.8 we compare initial spectrum and spectrums obtained after linear and nonlinear evolution of the simulations. As expectedly, agreement between the initial spectrum and spectrum obtained after linear evolution of the surface is excellent which was also shown in Figure 2.2. Spectrum obtained after nonlinear evolution of the surface differs from the initial spectrum due to nonlinear interaction and energy transfer between different wavenumber components.

In Figure 2.9 we compare the probability distributions obtained from linear and nonlinear water surface elevations with 1024 wave components. Both of these probability distributions obtained for a bin resolution of $0.05m$ and they show a Gaussian distribution behavior whereas probability distribution of nonlinear water surface elevation is a little shifted towards right indicating occurrence of sharper peaks and flatter troughs during nonlinear evolution.

In Figure 2.10 we present a sideband example where initial conditions for η and ϕ^s are specified by

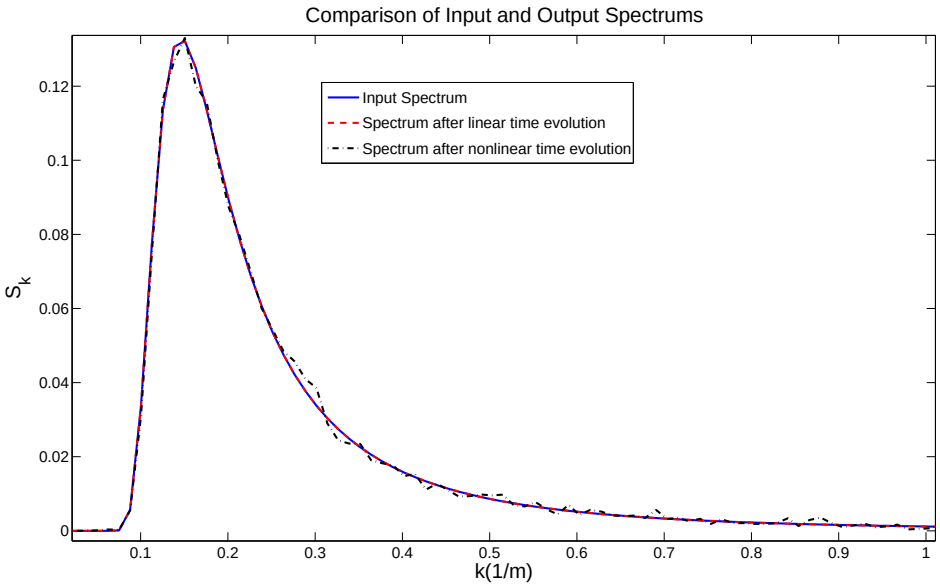

Figure 2. 8: Comparison of initial spectrum and final spectrums obtained after time evolution of the linear and nonlinear models

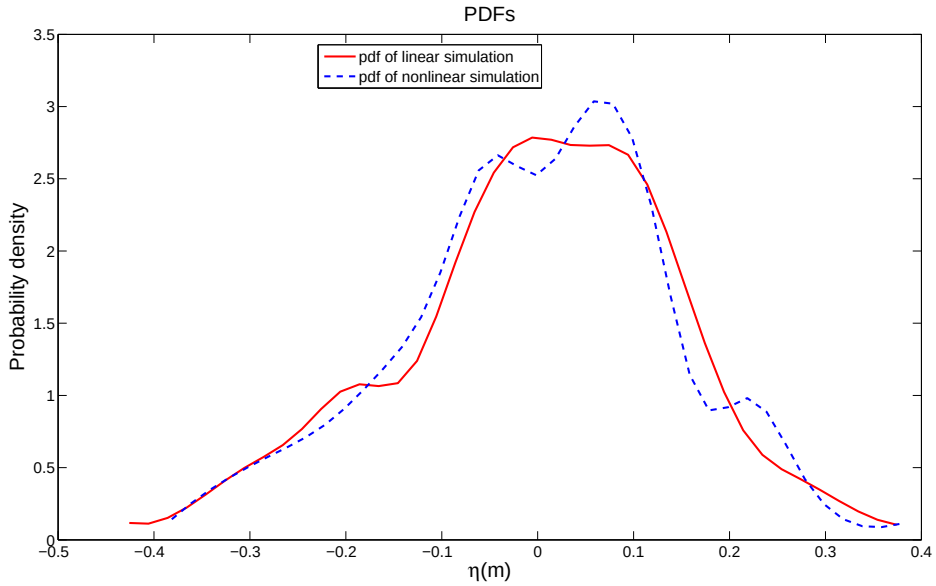

Figure 2. 9: Comparison of probability distribution functions for linear and nonlinear waves

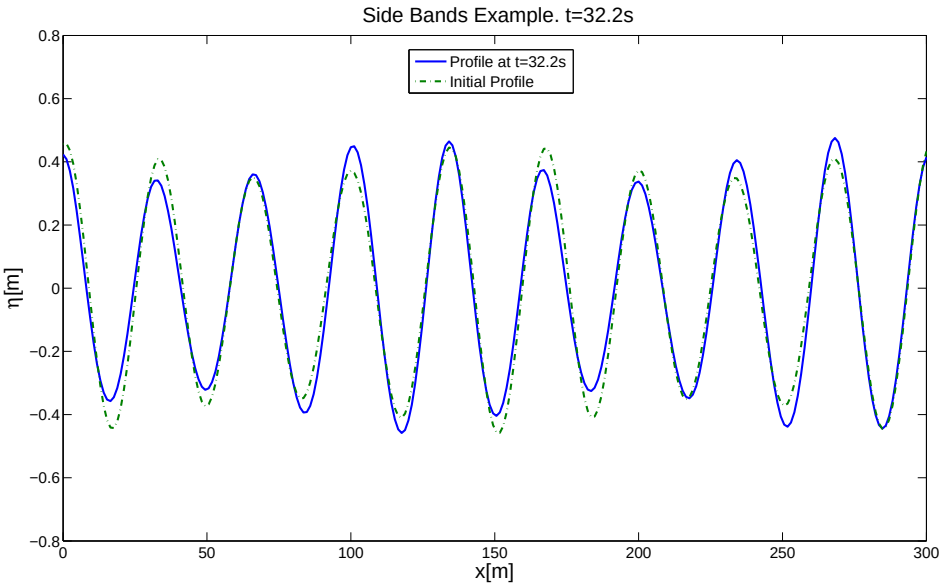

Figure 2. 10: Wave with two sideband modes example (continuous line refers to water surface fluctuation after evolution in time, dashed line refers to initial water surface fluctuation given by Equation (2.67))

$$\eta(x,0) = a\cos(9kx) + 0.1a\cos(7kx - \frac{\pi}{4}) + 0.1a\cos(11kx - \frac{\pi}{4}) \qquad (2.67)$$

$$\phi^s(x,0) = \frac{ag}{\sqrt{9kg\tanh(9kh)}}\cos(9kx)\exp(9k\eta) + 0.1\frac{ag}{\sqrt{7kg\tanh(7kh)}}\cos(7kx - \frac{\pi}{4})\exp(7k\eta)$$

$$+0.1\frac{ag}{\sqrt{11kg\tanh(11kh)}}\cos(11kx - \frac{\pi}{4})\exp(11k\eta) \qquad (2.68)$$

where a is the amplitude, h is the water depth, $k = 2\pi / L$ and L is the periodic domain length. Equations (2.67) and (2.68) are the initial conditions for sideband modes example with fundamental wavenumber, $9k$, and sideband wavenumbers $7k$ and $11k$. Integer numbers are chosen so that sideband modes can be fitted into computational domain (Dommermuth and Yue, 1987).

We check the accuracy of the nonlinear model by examining Fourier coefficients. In Figure 2.11 we present the time histories of Fourier coefficients obtained from monochromatic wave example. As it can be seen in the Figure 2.11,

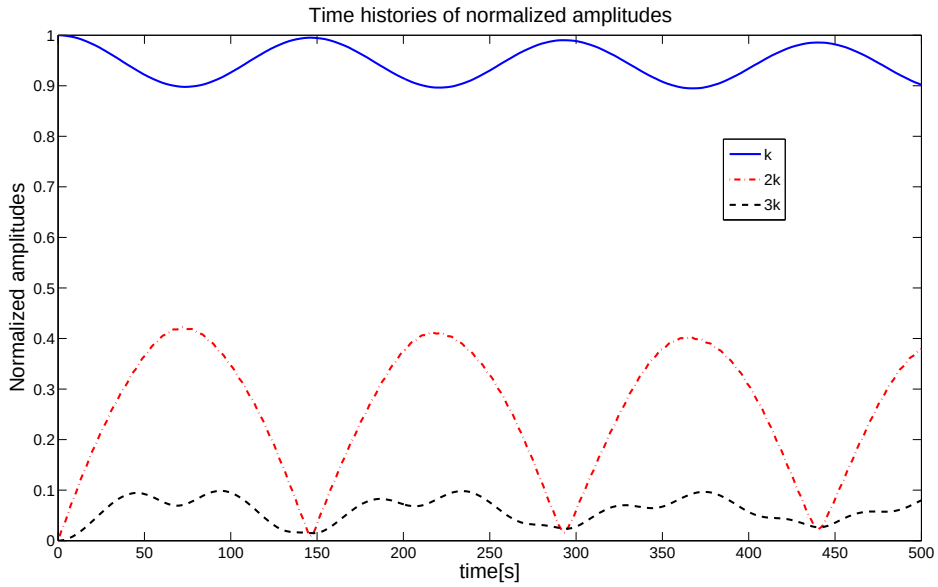

Figure 2. 11: Time histories of normalized Fourier amplitudes of first (straight line), second (dash-dotted line) and third (dashed line) harmonics

energy is moving between different wavenumber components and this energy transfer shows an oscillatory behavior. When the fundamental component loses some amount of its energy about $t = 75s$, the energy is transferred to higher harmonics. This is physically meaningful since conservation of energy law must be satisfied. Also it is useful to mention that, energy is not conserved perfectly in the system due to finite truncation in the number of free modes, N, and application of smoothing filter.

Effect of application of smoothing filter given by Equation (2.57) can be seen in Figure 2.12. Smoothing filter is applied at every 5 time step of the model and applied in wavenumber space in order to prevent saw-tooth instability. This filter results in an energy loss which is less than 10% for a simulation time of $125T_p \approx 500s$ as can be seen in Figure 2.12. This loss is reasonable since the maximum simulation time for coupled acoustic-wave model is $10T_p \approx 40s$ which is the duration of signal transmission used in the experiments.

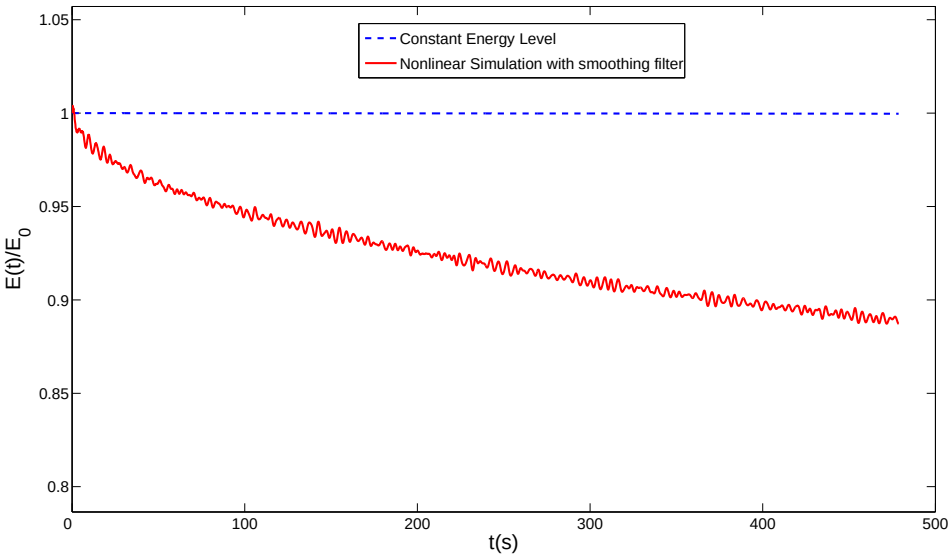

Figure 2. 12: Time history of change in the energy level, (dashed line refers to input energy level, continuous line refers to energy obtained from nonlinear surface elevation with smoothing filter applied)

Chapter 3

ACOUSTIC SCATTERING PROBLEM

In the literature full-wave acoustic models such as the normal mode and parabolic equation methods are extensively used. Also, ray tracing is one of the oldest methods which has been used to model acoustic propagation. The ray tracing method basically consists of approximating a given source by a fan of beams and tracing the propagation of these beams through the medium (Porter and Bucker, 1987).

Although ray models are preferred for their computational efficiency, there are some drawbacks of these methods such as occurrence of shadow zones and caustics (Porter and Bucker, 1987). There have been numerous efforts to modify the ray theory in order to get more accurate results while keeping the computations efficient.

The Gaussian beam approach that we use in our coupled wave-acoustic model can be viewed as a rigorous alternative to conventional ray tracing. We discuss the details of the Gaussian beam approach in the next section of this thesis.

3.1 Gaussian Beam Tracing and Bellhop Model

Consider a cylindrical coordinate system with r is the horizontal range and z is the vertical coordinate. Ray equations as a first-order system can be written as

$$\frac{dr}{ds} = c\rho(s) \tag{3. 1}$$

$$\frac{d\rho}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial r} \tag{3. 2}$$

$$\frac{dz}{ds} = c\xi(s) \tag{3. 3}$$

$$\frac{d\xi}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial z} \tag{3. 4}$$

where $[r(s), z(s)]$ are ray coordinates as a function of the arc length s, and $c(r,z)$ is the sound speed. Auxiliary angular variables $[\rho(s), \xi(s)]$ are quantities which are proportional to the local tangent vector (Porter and Bucker, 1987). Initial conditions are given as

$$[r(0), z(0)] = (r_s, z_s) \tag{3. 5}$$

$$[\rho(0), \xi(0)] = (\cos\alpha, \sin\alpha)/c(0) \tag{3. 6}$$

where (r_s, z_s) denotes the source coordinates and α is takeoff angle.

The beam curvature and width are calculated using two quantities $p(s)$ and $q(s)$, which are obtained by integrating two ordinary differential equations of the following form

$$\frac{dq}{ds} = c(s)p(s) \tag{3. 7}$$

$$\frac{dp}{ds} = -\frac{c_{nn}}{c^2(s)}q(s) \tag{3. 8}$$

where c_{nn} denotes the second normal derivative of the sound speed $c(r,z)$ which can be computed by

$$c_{nn} = c_{rr}\left(\frac{dr}{dn}\right)^2 + 2c_{rz}\left(\frac{dr}{dn}\right)\left(\frac{dz}{dn}\right) + c_{zz}\left(\frac{dz}{dn}\right)^2 \tag{3. 9}$$

Simply presenting the result, beam field can be defined by

$$u(s,n) = A\sqrt{\frac{c(s)}{rq(s)}}\exp\left(-i\omega\left\{\tau(s) + 0.5\frac{p(s)}{q(s)}n^2\right\}\right) \tag{3. 10}$$

where A is an arbitrary constant, ω is the angular frequency of the source, n is the normal distance from central ray. The term $\tau(s)$ in Equation (3.10) is the travel time of the ray which satisfies

$$\frac{d\tau}{ds} = \frac{1}{c(s)} \tag{3. 11}$$

Matching the exact solution for a point source in three dimensions A can be expressed as

$$A(\alpha) = \sum \delta\alpha\left(\frac{1}{c_o}\right)\exp\left(\frac{i\pi}{4}\right)\sqrt{\frac{q(0)\omega\cos\alpha}{2\pi}} \tag{3. 12}$$

where $\delta\alpha$ is the angle between adjacent rays, c_o is the constant sound speed in a homogeneous medium used for derivations. Combining Equations (3.10) and Equations (3.12), one can get the final result

$$u(s,n) = \sum \delta\alpha\left(\frac{1}{c_o}\right)\exp\left(\frac{i\pi}{4}\right)\sqrt{\frac{q(0)\omega\cos\alpha}{2\pi}}\sqrt{\frac{c(s)}{rq(s)}}\exp\left(-i\omega\left\{\tau(s) + 0.5\frac{p(s)}{q(s)}n^2\right\}\right) \tag{3. 13}$$

For a detailed discussion of Gaussian beam technique, the reader may refer to Porter and Bucker (1987).

Bellhop is a tool which uses the Gaussian beam tracing method. Bellhop numerically integrates the ray equations to keep track of acoustic rays inside the water column. This program has been tested with full wave acoustic models which are computationally intensive and shows excellent agreement with those methods (Porter

and Bucker, 1987). The method does not experience numerical artifacts which are present in standard ray models and is still computationally efficient like other ray based approaches (Porter and Bucker, 1987).

In order to run Bellhop, an input file containing environmental parameters such as water depth, source and receiver locations, top and bottom boundary conditions and sound speed profile should be created. This file is also known as the environmental file. Also, properties of beams such as number of beams to be used, their launching angles and type of beams should be specified. Following Heitsenrether (2004), we are using an initial fan of 1000 beams lying in an angular interval of 0° to 14°. All our calculations are based on eigenrays generated by Bellhop, which are the rays that are generated at the source and reach the receiver, which are reflected from surface only and do not experience any other scatter from bottom or surface. However during this surface reflection ray can be bounced from surface more than once.

After execution of Bellhop an output file is generated. This file contains information about arrival time, arrival angle, pressure amplitude, number of surface and number of bottom interactions. This output file is used to get the results of the model.

It is well known that when a sound wave is scattered from a moving object the frequency of the sound wave changes. This phenomenon is called Doppler shift. In this thesis we are calculating Doppler shifts experienced by acoustic rays at the moving surface. For this purpose we need to know the geometry of the eigenrays. Therefore at each time step of the wave model we are running Bellhop two times. One run is with the arrival file option in order to get arrival times of rays. The second run is with the eigenray option in order to plot the rays using a Matlab program called plotray. The Matlab program plotray which is standard in Bellhop package is modified in order to account for Doppler effect.

3.2 Doppler Shift Calculations

In the literature it is well known that motion of source, receiver or a scattering object causes a change in the frequency of the scattered signal. This is known as the Doppler effect. Depending on the geometry and direction of the motion the Doppler shift can be positive or negative.

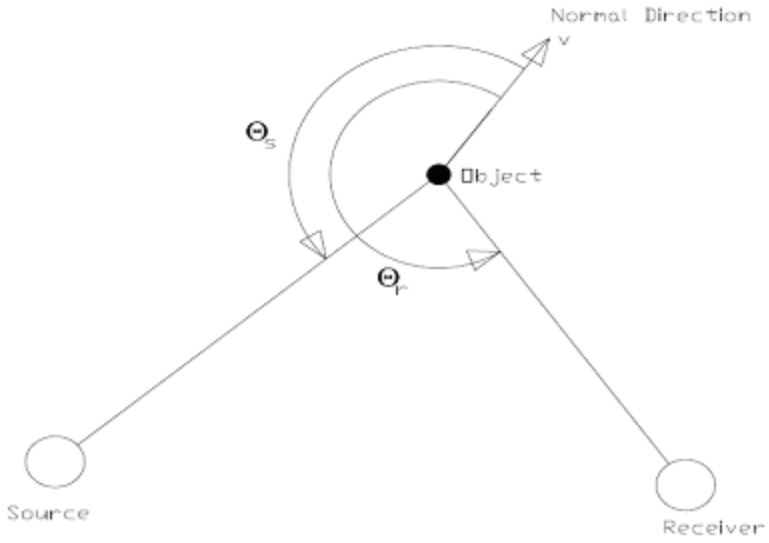

Figure 3. 1: Doppler effect of an moving object

Based on the geometry given in Figure 3.1, Doppler effect can be calculated using

$$f_r = \frac{f_s(c + v\cos\theta_s)}{(c - v\cos\theta_r)} \tag{3. 14}$$

where f_s is the source radiating frequency, f_r is the frequency at the receiver and c is the sound speed. v is the surface-normal velocity which we discussed in Section 2.3.

In Doppler shift calculations a source radiating frequency of 12 *kHz* is used which is the same value used in experiments we describe in Section 4.1.

In the surface scattering problem, the water surface serves as the scatterer at the point where the acoustic ray hits the surface. Once rays are generated by Bellhop and these intersection points are specified, Doppler shifts are calculated at these intersection points with the velocity of the surface specified by wave model.

When a ray experiences more then one surface bounce, Doppler shift calculations are applied successively at each bounce point. An example of multiple surface bounce is given in Figure 3.2. Application of Doppler shift calculations at each bounce point may not be the most reasonable approach since continous refraction of sound waves affects the geometry of bouncing of the acoustic ray and thus Doppler shifts calculations. Detailed study of Doppler shift calculations on the surface is recommended as a future work.

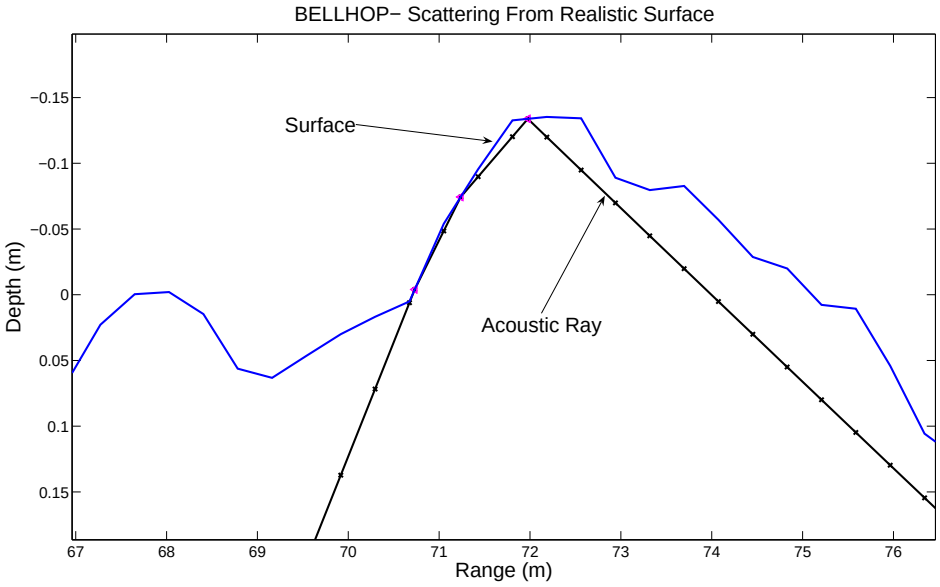

Figure 3.2: Multiple bounce of an acoustic ray

Chapter 4

DISCUSSION OF RESULTS OF COUPLED MODEL

4.1 Experimental Data

The experimental data used in this thesis is obtained from an experiment conducted in shallow water region of the Delaware Bay during September 22 through 29, 1997 (HFA 97). Main scope of this experiment was to measure the effect of the ocean environment on acoustic propagation.

During the experiment two stationary tripods were deployed on the sea floor, in 15m depth of water and separated horizontally by 387 m. For every 10 minutes during the whole experiment broad-band (0.6-18.0 kHz) chirp signals were transmitted every 0.345 s (pings) for 5 s and for 40 s intervals.

Among several different types of received signals observed in HFA 97, data is obtained for the acoustic rays traveling from source of one receiver to hydrophone receivers of the other tripod which is located 387 meters away. Time evolution of the individual ray paths which examines only one surface bounce are considered here. Statistics of these single bounce rays provides the data set to which model comparison are performed in Chapter 4 of this thesis.

During HFA 97 not only acoustic but also environmental measurements were performed. Tide height, current profiles, sound speed profiles were simultaneously measured. Air temperature, wind speed and wind direction measurements were collected at a lighthouse in the vicinity of the experimental field. Environmental variability was significant during the five day period of HFA 97 experiment. Wind speed ranged from 1 m/s to 15 m/s during the experiment. For a

much more detailed discussion of HFA 97, the reader may refer to Heitsenrether (2004).

4.2 Model Results

The main parameter we use in studying the model performance is the arrival time of the eigenrays. Arrival time is the time elapsed between the transmission of the eigenray at the source and its detection at the receiver. In Figure 4.1 we present

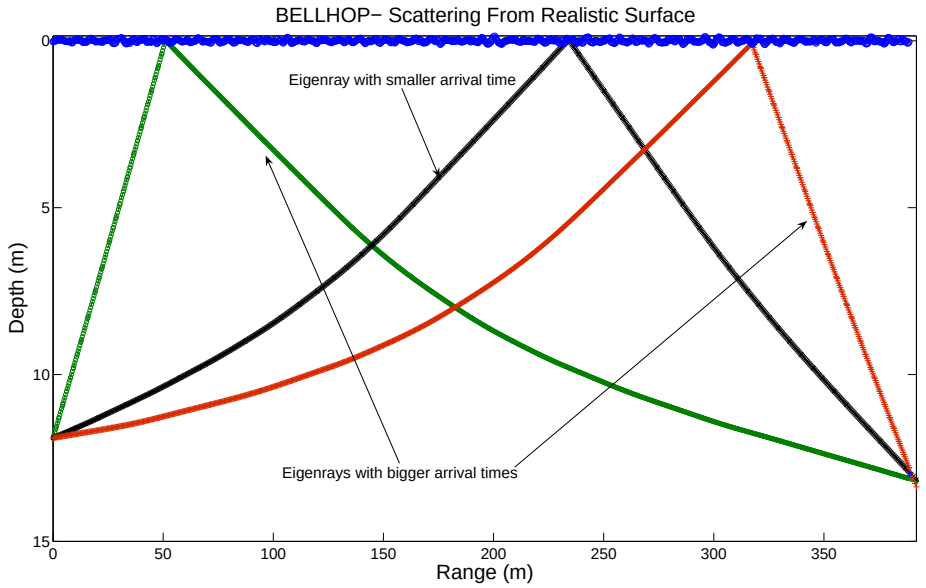

Figure 4. 1: Eigenrays with different arrival times

three eigenrays with different paths and thus different arrival times. Eigenray reflected from the middle region of the surface has the shortest path and thus smallest arrival time. Rays which are reflected from outer regions of the water surface have longer paths and thus bigger arrival times.

In the Figure 4.2 we present the time histories of arrival time of eigenrays and Doppler shifts associated with those eigenrays obtained from linear model for

September 24th, 1997, 03.02 am. Wind speed is $13.9 m/s$ for this simulation which is obtained from experimental data set. The most important factor affecting the results

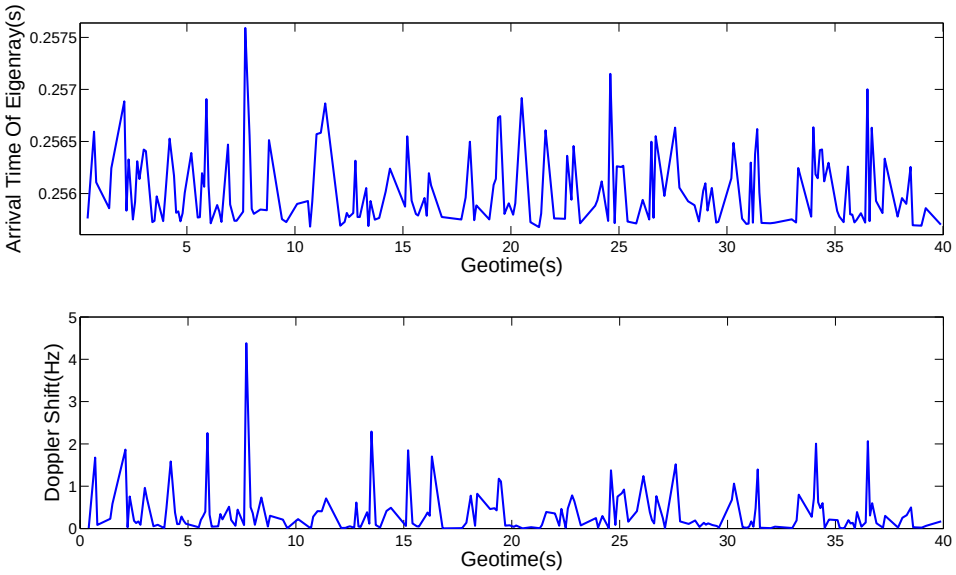

Figure 4. 2: Time histories of arrival times and Doppler shifts for September 24th, 1997, 03.02 am.

turned out to be the horizontal location of the bounce point of the acoustic ray. Rays which are reflected from the middle region of the surface have smaller arrival times. Arrival times of these rays lead to a smaller standard deviation of arrival times and agree well with the experimental results. Arrival times of rays which are reflected from outer regions of the surface lead to peaks in Figure 4.2, to a bigger standard deviation of arrival times and do not agree well with the experimental results. Using an acoustic model different than the ray based models can lead to better results since fineness of the initial ray fan will not be an important factor for those models.

In Figure 4.2, arrival times of eigenrays and associated Doppler shifts appear to be in phase. This is also expected because when an acoustic ray trapped inside a crest on the water surface its travel path is longer which makes arrival time

bigger as well as bigger surface-normal velocity on the crest leads to bigger Doppler shift of the signal. Also when the ray is trapped inside a crest it possibly experiences

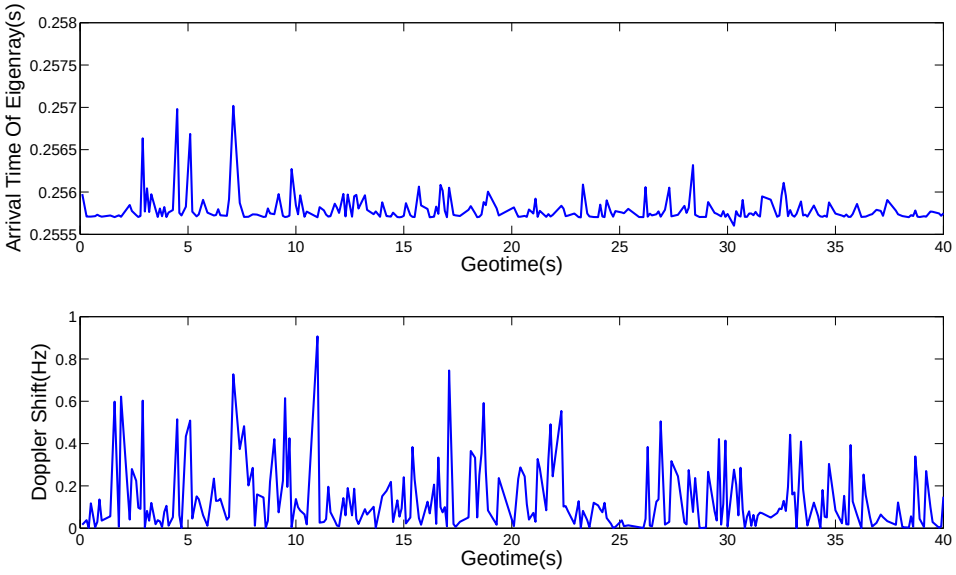

Figure 4. 3: Time histories of arrival times and Doppler shifts for September 24th, 1997, 00.00 am.

several surface bounces which may lead to successive Doppler shifts therefore to a bigger total Doppler shift.

In Figure 4.3, time histories of arrival times and Doppler shifts for September 24th, 1997, 00.00 am, obtained from a linear simulation are presented. In this simulation, wind speed is $1m/s$, and therefore surface roughness and surface-normal velocities are small. Therefore, arrival times of eigenrays and magnitude of Doppler shifts are smaller when compared to Figure 4.2. Also time series of arrival time and Doppler shifts in Figure 4.3 do not clearly appear to be in phase since surface elevation is very small and there are no well developed crests present for the wind speed of $1m/s$.

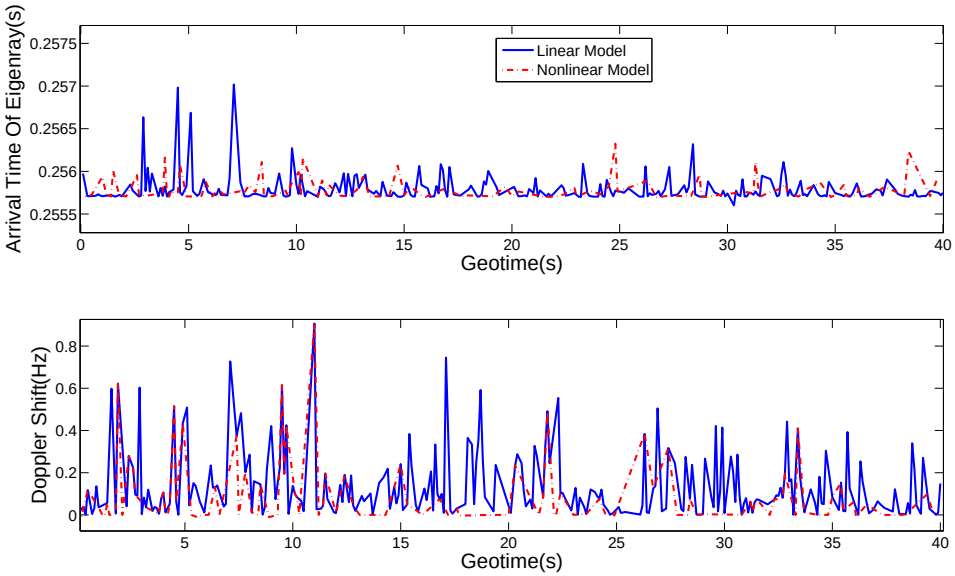

Figure 4. 4: Time histories of arrival times and Doppler shifts obtained from linear and nonlinear simulations for September 24th, 1997, 00.00 am. (continuous line refers to linear simulation, dashed line refers to nonlinear simulation)

In Figure 4.4, time histories of arrival times and Doppler shifts for September 24th, 1997, 00.00 am, obtained from linear and nonlinear simulations, are presented. Since the horizontal location of bounce point has a significant effect on results it is hard to observe the effect of nonlinearity. When eigenrays have same horizontal location of bouncing points both for linear and nonlinear simulations, nonlinearity leads to a bigger arrival time and a bigger Doppler shift since nonlinear surface profile is more peaked than linear surface profile and surface-normal velocities are bigger for nonlinear surface profile. This can rarely be observed by the coincident peaks of linear and nonlinear simulations such as on the second plot of Figure 4.3 for $t = 11s$ and $t = 34s$ where result of nonlinear simulation is at the top of result of linear simulation.

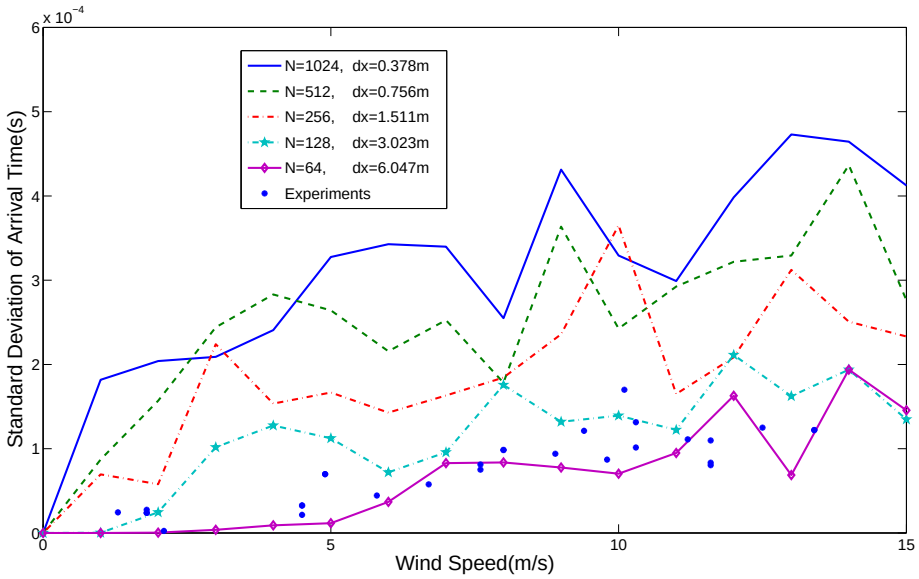

Figure 4. 5: Effect of surface partitioning dx on standard deviation calculations

In Figures 4.5 and 4.6 we present standard deviation of arrival times and average Doppler shifts in the frequency as functions of wind speed. These statistics are obtained using 200 successive runs for each $1m/s$ increment of wind speed. General tendency of the curves presented in Figures 4.5 and 4.6 show an expected behavior. When wind speed and therefore surface roughness is small, standard deviation of arrival times and Doppler shifts in the frequency of the acoustic rays converge to zero since surface becomes flat and surface-normal velocities become zero.

One of the major critics that Heitsenrether (2004) was subjected to is the selection of surface partitioning length, dx. Figures 4.5 and 4.6 show that this parameter of the wave model has a significant effect on the results. The number of

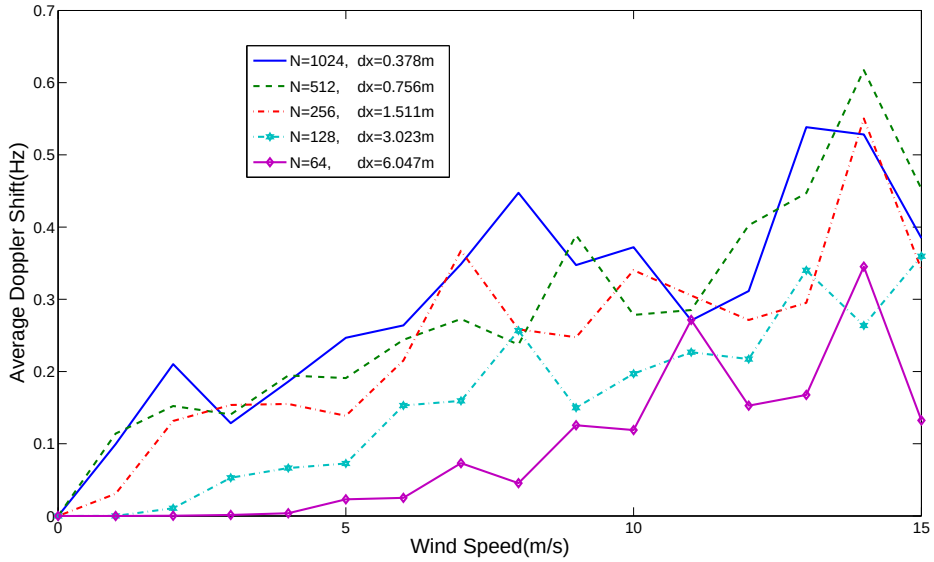

Figure 4. 6: Effect of surface partitioning dx on Doppler shift calculations

wave components used by Heitsenrether (2004) was $N = 64$ which agree well with experimental results since acoustic rays are mainly reflected from the points in the neighborhood of the flat surface specular point. The larger dx leads to smaller scatter since the surface is less faceted. From a convergence point of view, this is wrong, as a more resolved answer is by definition better. We have since shown that the problem lies in having a good estimate of the minimum travel time ray, which implies reliably finding the eigenray reflecting closest to the flat surface specular point. The problem becomes less severe at large dx because the surface is flatter in the neighborhood of the original specular point, and the ray there is usually found.

Using a bigger number of wave components and keeping the dx small enough is the correct approach of creating a spatially well represented surface profile. Based on this surface profile a ray based acoustic model has to be initiated using a very fine fan of beams to ensure that rays are mainly reflected from the points in the

neighborhood of the flat surface specular point. Another possible solution is to use a different acoustic model such as parabolic equation model.

Chapter 5

CONCLUSION

In this thesis, the effect of linear and nonlinear surface gravity waves on high frequency acoustic propagation has been discussed. Spectral linear and nonlinear wave models which solve governing equations are created, and these wave models are coupled with an acoustic Gaussian beam model called Bellhop. Coupled wave-acoustic models have been run with the same conditions of an experiment which was conducted in Delaware Bay in 1997. Results obtained by coupled wave-acoustic model are presented as functions of environmental parameters and are compared to experimental results. Limits of model applicability and validity are discussed.

As an extension to the wave model adopted from literature, the velocity component in the normal direction to the surface is calculated. The surface-normal velocity is used in the Doppler shift calculations.

For future work, the primary question to be addressed is the understanding of the effect of spatial resolution of surface waves on the acoustic ray method. Simulations with different types of acoustic models such parabolic equation model are needed in order to discuss the validity of the ray based Gaussian beam approach.

A 3D nonlinear wave model should be created. A 3D acoustic program which is capable of accounting for out of plane scattering should be used with 3D linear and 3D nonlinear wave models. 3D coupling will provide more realistic simulation of the underwater acoustic communication.

Moving source and receiver have effects on the geometry of the acoustic ray tracing model as well as Doppler shift calculations. These effects should be included in the calculations.

Since the presence of bubbles is another important factor which affects underwater acoustics, a much more complete hydrodynamic model should be created which can account for bubbles in the water column. Also, refractive and dissipative effect of bubbles should be included in the acoustic model as well.

We also know that as sound waves approach surface they are subjected to continuous refraction process. Therefore the approach we adopt may not be the most realistic approach for Doppler shift calculations. Literature about the whispering gallery concept should be reviewed and the effect of continuous interaction between rays and a curved surface at glancing incidence should be implemented in the model.

Appendix A

MATLAB AND FORTRAN PROGRAMS

1. Matlab-2D Linear Wave Model

```
%  Article: Douglas G.DOMMERMUTH and Dick K.P.YUE
%   A High-Order Spectral Method for the study of   nonlinear gravity waves
%  Cambridge, MA, USA 1986
% Two step procedure; pseudospectral method for determining nodal amplitudes
% and fourth order Runge-Kutta time integrator
% James KIRBY, Cihan BAYINDIR, University of Delaware, 02/04/2008
% Linear Version-Spectrum of Waves; Working in metric units
    clear all
    close all
    h=15;          % Water Depth, Bob Heitsenrether used h=15 meters
    g=9.81;        % Gravitational acceleration
    dt=0.1;        % Time step
    tmax=25;       % Maximum Time Of Scheme Evaluation
    t=0:dt:tmax;   % Time array
    Time=length(t);  % Length of Time Array
    N=2^10;          % Number of Fourier points;
                     % a power of 2 for fast computation
    xmin=0;          % Minimum x value
    xmax=500;       % Pick a domain length; with Hs=1.0 and Ts=10
                % energy spectrum drops to zero at L~500
                % choose domain length accordingly
    dx=(xmax-xmin)/N;   % Step size in x
    j=0:1:N-1;
    x=dx*j;             % Space array, Periodic domain
    Lmax=xmax-xmin;      % Length of periodic domain=max wave length

    k1=2*pi/Lmax*(0:1:N/2);      % First portion of Wavenumber array
    k2=2*pi/Lmax*(-N/2+1:1:-1);  % Second portion of Wavenumber array
    k=[k1,k2];                   % Combine first and second Portions of
                                 % Wave Number Array
```

```
W=sqrt(g*k1.*tanh(k1*h));  % wave angular frequency;
                    % using linear dispersion relationship
Cg=0.5.*(1+2.*k1*h./sinh(2.*k1*h)).*W./k1; %Group velocity
Cg(1)=0;
%-------------------------------------------------------
% 1st Formulation of Frequency Spectrum
% Frequency spectrum as a function of frequency
% Formulation for spectrum is taken from 'Random Seas and
% Design of Maritime Structures', Goda et.al. , second edition page 28
% equation 2.10
% To activate uncomment following seven lines
%-------------------------------------------------------------
Hs=1;          % Significant wave height
Ts=10;         % Significant wave period
fr=W/(2*pi);
Sf=0.257*Hs^2*Ts^-4.*(fr.^-5).*exp(-1.03.*(Ts.*fr).^-4);
Sf(1)=0;
Sk=Cg.*Sf./(2*pi);     % Frequency Spectrum ==> Wavenumber Spectrum
Sk(1)=0;

%-------------------------------------------------------------
% 2nd Formulation of Frequency Spectrum
% Frequency spectrum as a function of frequency
% Taken from master's thesis, Bob Heitsenrether, UD marine studies
% uses the Jonswap Spectrum given by Shore Protection Manual
%-------------------------------------------------------------
% X=18520;   % Equals 10 nautical miles, 1nmi=1852 meters
% U=13;      % Wind Speed in meters per second; Bob used values between 5-13
% gam=3.3; % Parameter for Jonswap Spectrum
% Wp=7*pi*g/U*(g*X/U^2)^-0.33; % Peak angular frequency
% alpha=0.076*(g*X/U)^-0.22;   % constant of equation for S(W)
% [dum,in]=find(W<=Wp);
% mm=max(in);
% SW1=alpha*g^2.*W(in).^-5.*exp(-5/4*(W(in)/Wp).^-4).*gam.^exp(-(W(in)-
Wp).^2/(2*0.07^2*Wp^2));
% SW2=alpha*g^2.*W(mm+1:N/2+1).^-5.*exp(-5/4*(W(mm+1:N/2+1)/Wp).^-
4).*gam.^exp(-(W(mm+1:N/2+1)-Wp).^2/(2*0.09^2*Wp^2));
% Sw=[SW1,SW2];
% Sw(1)=0;
% Sk=Cg.*Sw;     % Frequency Spectrum ==> Wavenumber Spectrum
% Sk(1)=0;
```

```
% Following line is not necessary but it gives an idea about selection of
% xmax, Horizontal length scale
%fp=Wp/(2*pi);   Tp=1/fp;
% According to deep water formula L~1.56 T^2
% Make sure that xmax you choose is about at least 25 times of this value
% since it represents the spectrum without much error

a2=sqrt(2*pi*Sk*2/Lmax);   % Nodal amplitudes obtained from spectrum

F2(1:N/2+1)=2*pi*rand(1,N/2+1);   % Uniformly Distributed Random Number
 % generator for phase angles on the interval [0,2*pi]

kTANHkh=abs(k).*tanh(abs(k).*h);    % In order to avoid calculation at every time
                                    % step product assigned to a variable

W(1)=1; % Since a(1)=0 W(1) will drop out. No problem to change it.

% Initial values for progressive wave; (Initial values for standing wave
% are; n is the same and Qs=0 BRUTE FORCE METHOD
% for j=1:1:N;
% n(j)=sum(a(1:N/2+1).*cos(k(1:N/2+1).*x(j)+F(1:N/2+1)));        % Initial
Values for Wat. Sur. Fluc.
% Qs(j)=sum(a(1:N/2+1).*g./W(1:N/2+1).*sin(k(1:N/2+1).*x(j)+F(1:N/2+1)));  %
Initial Values for Sur. Vel. Pot.
% %Qs(1:N)=0; % Initial Value for Sur. Vel. Pot. is 0 [for standing waves]
% end


%INVERSE FOURIER TRANSFORM METHOD
A2=a2(1:N/2+1).*exp(i*F2(1:N/2+1))/2;
A(1:N/2+1)=A2;  A(N/2+2:1:N)=conj(A2(N/2:-1:2));
A(N/2+1)=abs(A(N/2+1));
B(1:N/2+1)=-i*g./W.*A2;  B(N/2+2:1:N)=conj(-i*g./W(N/2:-1:2).*A2(N/2:-1:2));
B(N/2+1)=abs(B(N/2+1));
n=N*ifft(A);
Qs=N*ifft(B);

ni=n;
Qsi=Qs;
% Check the spectrum if initial wsf is given correctly
% figure(1)
```

```
    % B=fft(ni)/N
    % b=2*B(1:N/2+1)./exp(i*F(1:N/2+1));     S_k_2=b.^2*Lmax/(4*pi)
    % plot(k(1:N/2+1),Sk,k(1:N/2+1),S_k_2(1:N/2+1),'o')


for j=1:1:Time;    %Time Indice

 %% Runge-Kutta Time Integration with Pseudospectral Method %%

    % First Slope for Qs(surface vel.pot)
   d1=-g.*n;
   % First Slope for n(water surface fluct.)
   m1=ifft(fft(Qs).*kTANHkh);

   % Second Slope for Qs(surface vel.pot)
   d2=-g.*(n+0.5*dt.*m1);
   % Second Slope for n(water surface fluct.)
   m2=ifft(fft(Qs+0.5*dt.*d1).*kTANHkh);


   % Third Slope for Qs(surface vel.pot)
   d3=-g.*(n+0.5*dt.*m2);
   % Third Slope for n(water surface fluct.)
   m3=ifft(fft(Qs+0.5*dt.*d2).*kTANHkh);


   % Fourth Slope for Qs(surface vel.pot)
   d4=-g.*(n+dt*m3);
   % Fourth Slope for n(water surface fluct.)
   m4=ifft(fft(Qs+dt*d3).*kTANHkh);


   %Evaluate n(water surface fluc.) and Qs(surface velocity pot.) for next
   %time step
   n=n+dt/6.*(m1+2.*m2+2.*m3+m4);
   Qs=Qs+dt/6.*(d1+2.*d2+2.*d3+d4);

    %     nA(j)=n(3);
    %     nB(j)=n(45);
    %     nC(j)=n(76);
    %     nD(j)=n(108);
    %     nE(j)=n(157);
```

```
%     nF(j)=n(200);
%     nG(j)=n(244);
%     nH(j)=n(510);

 % Following lines are for calculating velocity component in the normal
 % direction which is going to be important in the Doppler shift
 % calculations of the Acoustic rays reflected from ocean surface

    Qsz=ifft(fft(Qs).*kTANHkh);  % Vertical velocity
    FTQs=i*k.*fft(Qs); FTQs(N/2+1)=abs(FTQs(N/2+1)); % At Nyquist freq.
    Qsx=ifft(FTQs);  % Horizontal velocity

    FTn=i*k.*fft(n); FTn(N/2+1)=abs(FTn(N/2+1));
    nx=ifft(FTn);    % Slope of water surface fluctuation
    udotn=(-Qsx.*nx+Qsz)./sqrt(nx.^2+1); % Velocity in the normal direction

    plot(x,n),title(['Airy Waves   t=',num2str((j-1)*dt),'s'])
    xlim([xmin xmax])
    ylim([-2 2])
   % subplot(2,1,2), plot(x,udotn); title('Velocity In Outward Normal Direction')
    pause(0.1)
    % j
end

 ax=0:dt:dt*(Time-1);

Varianceof_initial_spectrum= 0.0624*Hs^2  % take the integral under spectrum
(analytically)
Varianceof_initial_wsf=var(ni)
Varianceof_final_wsf=var(n)

Hs_result_from_initialWsf=4.004*sqrt(mean((ni.^2)))
Hs_result_from_finalWsf=4.004*sqrt(mean((n.^2)))
```

2. <u>Fortran-2D Linear Wave Model</u>

```
! Article: Douglas G.Dommermuth and Dick K.P.Yue
!         A High-Order Spectral Method for the study of
!         Nonlinear gravity waves     Cambridge, MA, USA 1986
!        Two Step Procedure; pseudospectral method for determining nodal
```

```
!       amplitudes and fourth order Runge-Kutta time integrator
!       Cihan BAYINDIR, James T. Kirby  University of Delaware 01/18/2009
!
!       Work In Metric Units
!       Linear, 1-D, Spectrum of  Waves

      program RandomSea
      real h,g, dt,tmax,xmin,xmax,L,pi,Hs,Ts
      integer sizet,j
!   p is the power of 2, has to be equal to power of 2 in the
!   coming line
      parameter (p=11)
!   INPUT N, has to be a power of 2 for fast computation
!   power of 2 has to be equal to p
      parameter (N=2**p)
      real x(N), F2(N/2+1), k(N), W(N/2+1), fr(N/2+1)
      real  kTANHkh(N), Cg(N/2+1)
      real Sf(N/2+1),Sk(N/2+1),a2(N/2+1)
      complex i, A_com_2(N/2+1), B_com_2(N/2+1)
      complex A_com(N),B_com(N)
      complex  Wsf(N),Qs(N)
      complex m1(N),m2(N),m3(N),m4(N),k1(N),k2(N),k3(N),k4(N)
      complex Qrhs1(N),Qrhs2(N),Qrhs3(N),Qrhs4(N)
      complex Qsz(N), Qsx(N),Wsfx(N),udotn(N)
      real I_var,sum,F_var
      real aux
CCCCCCCCCCCCCCC  INPUTS   CCCCCCCCCCCCCCCCCCCCCCC
!  Imaginary Number
      i=(0.0,1.0)
!   h is the water depth
      h=50
!   g is the gravitational acceleration
      g=9.81
!  dt is the time step
      dt=0.1
!  tmax is the maximum time of scheme evaluation
      tmax=10
! xmax is the maximum  value
      xmax=3000 ! Enter approximately 10 times of biggest wavelength
!  xmin is the minimum x value
      xmin=0
! Hs is the significant wave height
```

```
      Hs=1
! Ts is the significant wave period
      Ts=15
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
! Calculate size of t array
      sizet=tmax/dt
! Calculate pi
      pi=3.141592653589793
! pi=4*atan(1.0) ! Alternative expression for pi
!  Calculate step size in x
      dx=(xmax-xmin)/N
!  Calculate wave length
      L=xmax-xmin
! Create Space Array
      x(1)=xmin
      do 10 j=1,N-1
      x(j+1)=x(j)+dx
 10   continue


! Calculate Wave Number Vector For Spectral Code
      do 11 j=1,N/2+1
           k(j)=2*pi/L*(j-1)
 11   continue
      do 12 j=N/2+2,N
           k(j)=2*pi/L*(j-N-1)
 12  continue

! kTANHkh is needed in the time stepping in order to calculate at every time step
! calculate it here.
       do 13 j=1,N
         kTANHkh(j)=abs(k(j))*tanh(abs(k(j))*h)
 13    continue


! Calculate Angular Wave Frequency, Frequency, kTANHkh
! Remember these are half of the spectrum up to
! and including Nyquist frequency
      do 14 j=1,N/2+1
        W(j)=sqrt(g*k(j)*tanh(k(j)*h))
        fr(j)=W(j)/(2*pi)
        Cg(j)=0.5*(1+2*k(j)*h/sinh(2*k(j)*h))*W(j)/k(j)
```

```
 14   continue
      W(1)=1   ! W(1) will be denominator in coming lines
             ! avoid dividing by zero. This terms drops out since
             ! a(1)=0
      Cg(1)=0  ! k(1)=0 so dividing by zero creates NaN, avoid it
             ! this term will drop out since a(1)=0


! In order to calculate the nodal amplitudes we need frequency spectrum
! Formulation of the spectrum is taken from Random Seas and Design of Maritime
! Structures, Goda et. al. second edition page 28 equation 2.10

!  S(f)= Sf=frequency spectrum
!  S(k)=Sk= wavenumber spectrum calculated from frequency spectrum
!  a2 is the half of the real amplitude array
!  a  is the real amplitude array which is calculated by using symmetricity
!  of a, from a2

      Sf(1)=0
      Sk(1)=0
      a2(1)=0    ! k(1)=0 so dividing by zero creates NaN, avoid it
             ! these terms will drop out since a(1)=0
      do 15 j=2,N/2+1
      Sf(j)=0.257*(Hs**2)*(Ts**-4)*(fr(j)**-5)*exp(-1.03*(Ts*fr(j))**-4)
      Sk(j)=Cg(j)*Sf(j)/(2*pi)
      a2(j)=sqrt(2*pi*Sk(j)*2/L)
 15   continue


 !     open (unit=10,file="Sf.txt",ACCESS='APPEND')
 !     write(10,2) Sf
 !2     format (f8.5)
 !     close(10)

      call urand(F2,N/2+1)  ! Call Random Number Generator
                        ! for N/2+1 uniformly dist.
                        ! random number


! Initial Water Surface fluctuation and Surface Velocity Profile
! is being constructed using inverse Fourier Transform Method
```

```
! A2 half of the complex amplitude matrix
! B2 half of the complex water surface fluctuation matrix
      do 16 j=1,N/2+1
      A_com_2(j)=a2(j)*exp(i*F2(j))/2
      B_com_2(j)=-i*g/W(j)*A_com_2(j)
 16   continue

      do 17 j=1,N/2+1
        A_com(j)=A_com_2(j)
        B_com(j)=B_com_2(j)
 17   continue
      do 18 j=N/2+2,N
        A_com(j)=conjg(A_com_2(N-j+2))
        B_com(j)=conjg(B_com_2(N-j+2))
 18   continue
        A_com(N/2+1)=abs(A_com(N/2+1))
        B_com(N/2+1)=abs(B_com(N/2+1))

! Initial Values for water surface fluctuation Wsf
! and surface velocity potential Qs
       call ifft(A_com,p,N)
       Wsf=A_com
       call ifft(B_com,p,N)
       Qs=B_com

      open (unit=10,file="N_and_dt_dx.txt",ACCESS='APPEND')
      write(10,*) N , dt, dx
      close(10)

! Here Begins The Time Stepping. Time Integrator Is Fourth Order Runge-Kutta
      do 24  j=1,sizet

            !  First Slopes For Qs(surface velocity potential) and n
            !  water surface fluctuation are denoted as k1 and m1
            Qrhs1=Qs
            ! Calculate RHS of evaluation equation for 1st equation
            ! After operations Qrhs1 is changing!!
             call fft(Qrhs1,p,N)
             do 25 v=1,N
             Qrhs1(v)=Qrhs1(v)*kTANHkh(z)
 25           continue
             call ifft(Qrhs1,p,N)
```

```
      do 26 v=1,N
       k1(v)=-g*Wsf(v)
       m1(v)=Qrhs1(v)
       Qrhs2(v)=Qs(v)+0.5*dt*k1(v)
26     continue

      ! Second Slopes For Qs(surface velocity potential) and n
      ! water surface fluctuation are denotes as k2 and m2
      ! Calculate RHS of evaluation equation for 2nd equation
      ! After operations Qrhs2 is changing!!
      call fft(Qrhs2,p,N)
      do 27 v=1,N
      Qrhs2(v)=Qrhs2(v)*kTANHkh(v)
27     continue
      call ifft(Qrhs2,p,N)
      do 28 v=1,N
        k2(v)=-g*(Wsf(v)+0.5*dt*m1(v))
        m2(v)=Qrhs2(v)
        Qrhs3(v)=Qs(v)+0.5*dt*k2(v)
28     continue

      ! Third Slopes For Qs(surface velocity potential) and n
      ! water surface fluctuation are denotes as k3 and m3
      ! Calculate RHS of evaluation equation for 2nd equation
      ! After operations Qrhs2 is changing!!
      call fft(Qrhs3,p,N)
      do 29 v=1,N
       Qrhs3(v)=Qrhs3(v)*kTANHkh(v)
29    continue
      call ifft(Qrhs3,p,N)
      do 30 v=1,N
        k3(v)=-g*(Wsf(v)+0.5*dt*m2(v))
        m3(v)=Qrhs3(v)
        Qrhs4(v)=Qs(v)+dt*k3(v)
30     continue


      ! Fourth  Slopes For Qs(surface velocity potential) and n
      ! water surface fluctuation are denotes as k4 and m4
      ! Calculate RHS of evaluation equation for 2nd equation
      ! After operations Qrhs2 is changing!!
      call fft(Qrhs4,p,N)
```

```
          do 31 v=1,N
           Qrhs4(v)=Qrhs4(v)*kTANHkh(v)
31          continue
          call ifft(Qrhs4,p,N)
          do 32 v=1,N
            k4(v)=-g*(Wsf(v)+dt*m3(v))
            m4(v)=Qrhs4(v)
            Wsf(v)=Wsf(v)+dt/6*(m1(v)+2*m2(v)+2*m3(v)+m4(v))
            Qs(v)=Qs(v)+dt/6*(k1(v)+2*k2(v)+2*k3(v)+k4(v))
32          continue


!         Write Water Surface Fluctuation To File
      open (unit=9,file="Water Surface Fluctuation.txt",ACCESS='APPEND')
      do 33 z=1,N
       write(9,1) Wsf(z)
1      format (f8.5,f8.5)
33    continue
      close(9)

!-------------------------------------------------------------------------------
          ! Following lines are for calculating velocity component in the normal
          ! direction which is going to be important in the Doppler shift
          ! calculations of the acoustic rays reflected from the ocean surface

          ! Qsz is the z derivative of Qs; vertical velocity calculate from spectral
          ! formulation
          Qsz=Qs
          call fft(Qsz,p,N)
          do 34 v=1,N
          Qsz(v)=Qsz(v)*kTANHkh(v)
34         continue
          call ifft(Qsz,p,N)
          Qsz=real(Qsz) ! Avoid roundoff  errors in the imaginary part


          ! Qsx is the x derivative of Qs, horizontal velocity
          Qsx=Qs
          call fft(Qsx,p,N)
          do 35 v=1,N
          Qsx(v)=Qsx(v)*k(v)*i
35         continue
```

```
        Qsx(N/2+1)=abs(Qsx(N/2+1))
        call ifft(Qsx,p,N)
        Qsx=real(Qsx)  ! Avoid roundoff errors in the imaginary part

        ! Wsfx is the x derivative of water surface fluc. Wsf
        Wsfx=Wsf
        call fft(Wsfx,p,N)
        do 36 v=1,N
        Wsfx(v)=Wsfx(v)*k(v)*i
36       continue
        Wsfx(N/2+1)=abs(Wsfx(N/2+1))
        call ifft(Wsfx,p,N)
        Wsfx= real(Wsfx) ! Avoid roundoff errors in the imaginary part


       ! udotn is the dot product of velocity u with normal gives the velocity
!component
        ! in the normal direction
        do 37 v=1,N
        udotn(v)=(-Qsx(v)*Wsfx(v)+Qsz(v))/sqrt(Wsfx(v)**2+1)
37       continue
!-------------------------------------------------------------------------

24      continue


    ! Check if the energy is conserved
    ! I_var is the initial variance take the integral of Sf from 0 to inf
    ! Gives you initial variance
    I_var=0.0624*(Hs**2)


    ! F-var is the Final variance of water surface fluctuation
    sum=0
     do 40 j=1,N
      sum=sum+(Wsf(j))**2
40    continue
    F_var=sum/N


    print *, 'Initial Var=', I_var,'Final Var=',F_var
    print *, ' udotn', udotn
```

```
      stop
      end

!!!!!!!!!!!  SUBROUTINES !!!!!!!!!!!!!!!

!    Simple random number generator
      subroutine urand(F,N)
      integer  N,j, seed
      real pi,old,c,d
      real F(N)
      PI=3.141592653589793
!     pi=4*atan(1.0) ! Alternative expression for pi
      seed=11
      old=seed
      j=1
      do 1 j=1,N-1,2
        c=mod ((57*old+1),2*pi)
        d=mod ((57*c+1),2*pi)
        old=d
        F(j)=c
        F(j+1)=d
 1    continue
      return
      end
! End of simple random number generator



! 1-D Fast Fourier Transfrom
! N=2^p, A is the complex array to be fourier transformed
      subroutine fft(A,p,N)
      complex A(N),U,W,T
!  Divide all elements by N
      do 1 J=1,N
 1    A(J)=A(J)/N
!  Reorder Sequence According to fig 12.8 of Newland
      NBD2=N/2
      NBM1=N-1
      J=1
      do 4 L=1,NBM1
      if(L.ge.J) go to 2
      T=A(J)
```

```
      A(J)=A(L)
      A(L)=T
 2    K=NBD2
 3    if(K.ge.J) go to 4
      J=J-K
      K=K/2
      go to 3
 4    J=J+K
!   Calculate FFT acoording to fig 12.5
      PI=3.141592653589793
      do 6 M=1,p
      U=(1.0,0.0)
      ME=2**M
      K=ME/2
      W=CMPLX(COS(PI/K),-SIN(PI/K))
      do 6 J=1,K
      do 5 L=J,N,ME
      LPK=L+K
      T=A(LPK)*U
      A(LPK)=A(L)-T
 5    A(L)=A(L)+T
 6    U=U*W
      return
      end
! End of 1-D Fast Fourier Transform



! 1-D Inverse Fast Fourier Transform
! Taken from www.algarcia.org/nummeth/Fortran/ifft.f
! And normalization has been changed by multiplying with N rather than dividing by N
! Due to definition of Fourier Pair
      subroutine ifft(A,p,N)
      integer N
      complex A(N)
!     Routine to compute inverse Fourier transform using FFT algorithm
      integer j
      do j=1,N
        A(j)=conjg(A(j))
      enddo

      call fft(A,p,N)
```

```
    do j=1,N
      A(j)=conjg(A(j))*N ! Compute conjugates and Normalize
    enddo
    return
    end
! End of 1-D Inverse Fourier Transform
```

3. Matlab-3D Linear Wave Model

```
%   Article: Douglas G.DOMMERMUTH and Dick K.P.YUE
%   A High-Order Spectral Method for the study of
%    nonlinear gravity waves  Cambridge, MA, USA 1986
%   Two step procedure; pseudospectral method for determining nodal amplitudes
%    and fourth order Runge-Kutta time integrator
% James KIRBY, Cihan BAYINDIR, University of Delaware, 1/20/2009

% 2-D, Linear Version-Spectrum of Waves; Working in metric units
clear all
close all
h=15;          % Water Depth
g=9.81;        % Gravitational acceleration
dt=0.25;       % Time step
tmax=50;       % Maximum Time Of Scheme Evaluation
t=0:dt:tmax;   % Time array
Time=length(t); % Length of Time Array

N=2^9;          % Number of Fourier points in x direction;
              % a power of 2 for fast computation

M=2^9;          % Number of Fourier points in y direction

xmin=0;         % Minimum x value
ymin=0;         % Minimum y value
xmax=780;       % Pick a domain length in x direction;

ymax=500;       % Pick a domain length in y direction; with Hs=1.0
              % and Ts=15  L~350 So choose a length which is on the order
                                % of 10 times this value
```

```
dx=(xmax-xmin)/N;   % Step size in x
j=0:1:N-1;
x=dx*j;             % x array, Periodic domain
Lx=xmax-xmin;    % Length of periodic domain

dy=(ymax-ymin)/M;
j=0:1:M-1;
y=dy*j;          % y array, Periodic domain
Ly=ymax-ymin;  % Length of periodic in y dir

dkx=2*pi/Lx;
kx1=dkx*(0:1:N/2);      % First portion of kx array
kx2=dkx*(-N/2+1:1:-1);  % Second portion of kx array
kx=[kx1,kx2];           % Combine first and second Portions of
                        % kx Array

dky=2*pi/Ly;
ky1=dky*(0:1:M/2);      % First portion of ky array
ky2=dky*(-M/2+1:1:-1);  % Second portion of ky array
ky=[ky1,ky2];           % Combine first and second Portions of
                        % ky Array

%Create wavenumber matrix
[KX,KY]=meshgrid(kx,ky);
k=sqrt(KX.*KX+KY.*KY);
k=k';

W=sqrt(g*k.*tanh(k*h));  % Angular frequency by linear dispersion relationship


W(1,1)=0.01;  % To avoid dividing by zeros need these modifications, these drop out
k(1,1)=0.01;  % To avoid dividing by zeros need these modifications, these drop out

Cg=0.5.*(1+2.*k*h./sinh(2.*k*h)).*W./k;    %Group velocity
Cg(1,1)=0;



kTANHkh=abs(k).*tanh(abs(k).*h);    % In order to avoid calculation at every time
                        % step product assigned to a variable
                        % Will be needed inside the time stepping
kTANHkh=kTANHkh';
```

```
% ------------------------------------------------------------
% ------------------Direction Matrix-------------------------
TetaPr=0;  % Principal direction of wave propogation
              % Pick one of the following values
               % 0, 45, 90 degrees or arbitrarily
% TetaRelative=0;  % Relative direction of acoustic communication path
%               % with respect to principal direction of wave propogation
%               % Pick one of the following values
%               % For TetaPr=0;  Select 0 or 45 or 90
%               % For TetaPr=45; Select 0, 45
%               % For TetaPr=90; Select 0, -45

%Create Theta(direction) matrix
Teta(1,1)=0;       % kx=0 so avoid dividing by zero, theta is not defined at this point,
%set any number
Teta(2:N/2+1,1)=0;  % kx=0 so avoid dividing by zero
Teta(N/2+2:N,1)=180;
for j=2:1:M/2+1;
Teta(1:N,j)=180/pi*(atan(ky(j)./kx(1:N)));
end
Teta(N/2+2:N,2:M/2+1)=180+Teta(N/2+2:N,2:M/2+1);
Teta(1,2:M/2+1)=90;
Teta(1,M/2+2:M)=-90;
 for p=2:1:N/2+1;
    Teta(p,M/2+2:M)=180/pi*(atan(ky(M/2+2:M)./kx(p)));
 end
for p=N/2+2:1:N;
   Teta(p,M/2+2:M)=-180+180/pi*(atan(ky(M/2+2:M)./kx(p)));
end
%------------------------------------------------------------



%-----------------------------------------------------------
% 1st Formulation of Frequency Spectrum
% Frequency spectrum as a function of frequency
% Formulation for spectrum is taken from 'Random Seas and
% Design of Maritime Structures', Goda et.al. , second edition page 28
% equation 2.10
% To activate uncomment following seven lines
```

```
%-----------------------------------------------------------
%  Hs=1;          % Significant wave height
%  Ts=15;         % Significant wave period
%  fr=W/(2*pi);
%  Sf=0.257*Hs^2*Ts^-4.*(fr.^-5).*exp(-1.03.*(Ts.*fr).^-4);
%  Sf(1,1)=0;
%  Sk=Cg.*Sf./(2*pi);     % Frequency Spectrum ==> Wavenumber Spectrum
%  Sk(1,1)=0;

%-----------------------------------------------------------
% 2nd Formulation of Frequency Spectrum
% Frequency spectrum as a function of frequency
% Taken from master's thesis, Bob Heitsenrether, UD marine studies
% uses the Jonswap Spectrum given by Shore Protection Manual
%-----------------------------------------------------------
 X=18520;    % Equals 10 nautical miles, 1nmi=1852 meters
 U=15;       % Wind Speed in meters per second; Bob used values between 5-13
 gam=3.3;    % Parameter for Jonswap Spectrum
 Wp=7*pi*g/U*(g*X/U^2)^-0.33;  % Peak angular frequency
 alpha=0.076*(g*X/U)^-0.22;    % constant of equation for S(W)
cons=alpha*g^2;

 for p=1:1:N;
    for j=1:1:M;
      if W(p,j)<=Wp;
      Sw(p,j)=cons.*(W(p,j).^-5).*(exp(-5/4*(W(p,j)/Wp).^-4)).*(gam.^exp(-(W(p,j)-
Wp).^2/(2*0.07^2*Wp^2)));
      elseif  W(p,j)>Wp;
      Sw(p,j)=cons.*(W(p,j).^-5).*(exp(-5/4*(W(p,j)/Wp).^-4)).*(gam.^exp(-(W(p,j)-
Wp).^2/(2*0.09^2*Wp^2)));
      end
    end
 end
 Sw(1,1)=0;
 Sk=Cg.*Sw;     % Frequency Spectrum ==> Wavenumber Spectrum
 Sk(1,1)=0;

%% Numerical Integral Under Jonswap Spectrum=Initial Variance
%  Need to evaluate to understand if the scheme is conserving energy

F1 = @(W)alpha*g^2.*W.^-5.*exp(-5/4*(W/Wp).^-4).*gam.^exp(-(W-
Wp).^2/(2*0.07^2*Wp^2));
```

```
Initial_variance1 = quad(F1,0,Wp);

F2 =  @(W)alpha*g^2.*W.^-5.*exp(-5/4*(W/Wp).^-4).*gam.^exp(-(W-
Wp).^2/(2*0.09^2*Wp^2));

Initial_variance2 = quad(F2,Wp,10);
Initial_variance =Initial_variance1 +Initial_variance2


%Sech^2 type directional spreading function. Reference: "Directional
%Spectra of wind generated waves", Donelan M.A., Hamilton J., and Hui, W.H.
Wnormal=W/Wp;
for p=1:1:N;
   for j=1:1:M;
     if (Wnormal(p,j)<=0.56 | Wnormal(p,j)>=1.6);
         Beta(p,j)=1.24;

     elseif (0.56< Wnormal(p,j)< 0.95);
       Beta(p,j)=2.61*(Wnormal(p,j))^1.3;

     elseif  (0.95 < Wnormal(p,j)< 1.6);
       Beta(p,j)=2.28*(Wnormal(p,j))^-1.3;
      end
   end
end
 Gt=Beta/2.*(sech(Beta.*(pi/180*(Teta-TetaPr)))).^2;
  SkGt=Sk.*Gt;

% Nodal amplitudes obtained from directional spectrum

a=sqrt(SkGt*dkx*dky*2./k); %Use Jacobian of Transformation to convert S(k,teta) to
%S(kx,ky) % {J}=k
a(1,1)=0;
%a2=a(1:N,1:M/2+1); % Half of the nodal amplitude matrix

%% Use uniformly distributed random number generator for phase angles
 F=2*pi*rand(N,M); % Uniformly Distributed Random Number generator
                   % for phase angles on the
                   % interval [0,2*pi]

% Initial Conditions For Water Surface Fluctuation and Velocity Potential
```

```
% First define complex amplitudes and then by inverse Fourier Transform
% construct the water surface

Aorg=a.*exp(i*F)/2;   Borg=-i*g./W.*Aorg;
A=Aorg;               B=Borg;

A(2:N,2:M/2)=Aorg(2:N,2:M/2)+ conj(Aorg(N:-1:2,M:-1:M/2+2));
A(2:N,M/2+2:M)=Aorg(2:N,M/2+2:M)+conj(Aorg(N:-1:2,M/2:-1:2));

A(1,M/2+2:M)=conj(A(1,M/2:-1:2));
A(N/2+2:N,1)=conj(A(N/2:-1:2,1));


B(2:N,2:M/2)=Borg(2:N,2:M/2)+ conj(Borg(N:-1:2,M:-1:M/2+2));
B(2:N,M/2+2:M)=Borg(2:N,M/2+2:M)+conj(Borg(N:-1:2,M/2:-1:2));

B(1,M/2+2:M)=conj(B(1,M/2:-1:2));
B(N/2+2:N,1)=conj(B(N/2:-1:2,1));

A(N/2+1,:)=abs(A(N/2+1,:)); % At Nyquist frequency need real numbers
A(:,M/2+1)=abs(A(:,M/2+1)); % At Nyquist frequency need real numbers
A(N/2+2:N,M/2+1)=A(N/2:-1:2,M/2+1); % At Nyquist frequency real numbers has to
% be symmetric

B(N/2+1,:)=abs(B(N/2+1,:)); % At Nyquist frequency need real numbers
B(:,M/2+1)=abs(B(:,M/2+1)); % At Nyquist frequency need real numbers
B(N/2+2:N,M/2+1)=B(N/2:-1:2,M/2+1); % At Nyquist frequency real numbers has to
%be symmetric

n=N*M*ifft2(A);
Qs=N*M*ifft2(B);

n=n';
Qs=Qs';


figure
for j=1:1:Time;    %Time Indice
   % Runge-Kutta Time Integration with Pseudospectral Method %%

   % First Slope for Qs(surface vel.pot)
   d1=-g.*n;
```

```
  % First Slope for n(water surface fluct.)
  m1=ifft2(fft2(Qs).*kTANHkh);

  % Second Slope for Qs(surface vel.pot)
  d2=-g.*(n+0.5*dt.*m1);

  % Second Slope for n(water surface fluct.)
  m2=ifft2(fft2(Qs+0.5*dt.*d1).*kTANHkh);

  % Third Slope for Qs(surface vel.pot)
  d3=-g.*(n+0.5*dt.*m2);

   % Third Slope for n(water surface fluct.)
   m3=ifft2(fft2(Qs+0.5*dt.*d2).*kTANHkh);

   % Fourth Slope for Qs(surface vel.pot)
   d4=-g.*(n+dt*m3);

   % Fourth Slope for n(water surface fluct.)
   m4=ifft2(fft2(Qs+dt*d3).*kTANHkh);

   %Evaluate n(water surface fluc.) and Qs(surface velocity pot.) for next
   %time step using fourth order Runge Kutta time integrator
   n=n+dt/6.*(m1+2.*m2+2.*m3+m4);
   Qs=Qs+dt/6.*(d1+2.*d2+2.*d3+d4);

% Store time series at some points
%    nA(j)=n(3,8);
%    nB(j)=n(60,60);
%    nC(j)=n(90,3);
%    nD(j)=n(110,235);
%    nE(j)=n(145,82);
%    nF(j)=n(21,4);
%    nG(j)=n(12,250);
%    nH(j)=n(200,14);



% Following lines are added in order to calculate velocity components in
% the normal direction
%   FT2Qsx=i*KX.*fft2(Qs); FT2nx=i*KX.*fft2(n);
```

```
%    FT2Qsy=i*KY.*fft2(Qs); FT2ny=i*KY.*fft2(n);
%
%    FT2Qsx(N/2+1,:)=abs(FT2Qsx(N/2+1,:));
%    FT2Qsx(:,M/2+1)=abs(FT2Qsx(:,M/2+1));
%    FT2nx(N/2+1,:)=abs(FT2nx(N/2+1,:));
%    FT2nx(:,M/2+1)=abs(FT2nx(:,M/2+1));
%
%    FT2Qsy(N/2+1,:)=abs(FT2Qsy(N/2+1,:));
%    FT2Qsy(:,M/2+1)=abs(FT2Qsy(:,M/2+1));
%    FT2ny(N/2+1,:)=abs(FT2ny(N/2+1,:));
%    FT2ny(:,M/2+1)=abs(FT2ny(:,M/2+1));
%
%    Qsz=ifft2(kTANHkh.*fft2(Qs));  % Vertical Velocity at the surface
 %    Qsx=ifft2(FT2Qsx);   % Horizontal Velocity in the x dir at the surface
 %    Qsy=ifft2(FT2Qsy);   % Horizontal Velocity in the y dir at the surface
 %    nx=ifft2(FT2nx);     % x derivative of water surface fluc.
 %    ny=ifft2(FT2ny);     % y derivative of the water surface fluc.
 %    udotn=(-Qsx.*nx-Qsy.*ny+Qsz)./sqrt(nx.^2+ny.^2+1); % Velocity in the normal
 %direction

 nalongpath=n(1,:);

 subplot(2,1,1),
 pcolor(real(n)), shading interp, axis('equal')
 % hold on
 caxis([-0.25 0.25])    % Set limits for color map to prevent sudden darkening and
                         %lightening
%view([0,-2,10])
title(['Airy Waves   t=',num2str(j*dt),'s'])
%subplot(2,1,2),quiver(1:N,1:M,Qsx,Qsy), axis equal
subplot(2,1,2),plot(x,nalongpath),ylim([-1  1])
pause(0.05)
%    j
end
```

4. Fortran-3D Linear Wave Model

```
!       Article: Douglas G.Dommermuth and Dick K.P.Yue
!       A High-Order Spectral Method for the study of
!       Nonlinear gravity waves     Cambridge, MA, USA 1986
!
```

```
!       Two Step Procedure; pseudospectral method for determining nodal
!       amplitudes and fourth order Runge-Kutta time integrator
!       Cihan BAYINDIR, James T. Kirby  University of Delaware 01/22/2009
!
!       Work In Metric Units
!       Linear, 2-D, Spectrum of  Waves

      program RandomSea
      real h,g, dt,tmax,xmin,xmax,L,pi,Hs,Ts
      integer sizet,j
!     Enter p1, an integer , is going to be the power of  2
      parameter (p1=8)
!     Enter p2, an integer , is going to be the power of  2
      parameter (p2=8)
!     N has to be a  power of 2 for Fast Fourier Trans.
      parameter (N=2**p1)
!     M has to be a power of 2 for Fast Fourier Trans.
      parameter (M=2**p2)
      real dkx, dky
      real x(N),y(N), F(N,M),  fr(N,M)
      real kx(N), ky(N), k(N,M), W(N,M)
      real KXa(N,1), KYa(M,1)
      real Cg(N,M),  kTANHkh(N,M), Beta(N,M)
      real Sf(N,M),Sw(N,M),Sk(N,M), Wnormal(N,M)
      real wa,wb,step1, waxis1(40001),func1(40001), Sumodd1, Sumeven1
      real step2, func2(40001), waxis2(40001), Sumodd2, Sumeven2
      real Fetch, U, gam, Wp, alpha, cons
      real TetaPr, Go , Teta(N,M)
      real Gt(N,M), SkGt(N,M), a(N,M)
      double precision s, arg1, arg2
      complex i
      complex A_com_org(N,M),B_com_org(N,M)
      complex A_com(N,M), B_com(N,M)
      complex Wsf(N,M),Qs(N,M)
      complex Wsf_tr(N,M), Qs_tr(N,M)
      complex m1(N,M),m2(N,M),m3(N,M),m4(N,M),k1(N,M),k2(N,M)
      complex k3(N,M),k4(N,M)
      complex Qrhs1(N,M),Qrhs2(N,M),Qrhs3(N,M),Qrhs4(N,M)
      complex Qsz(N,M),Qsx(N,M),Qsy(N,M),Wsfx(N,M),Wsfy(N,M)
      complex udotn(N,M)
      real I_var,sum1,sum2,sum3,sum4,sum5,sum6,sum7,sum8
      real  var1,var2,var3,var4,var5,var6,var7,var8
```

```
!-----------------------------------------------------------
! ---------------------  INPUTS  --------------------------
! _____________________________________________
!  Imaginary Number
      i=(0.0,1.0)
!   h is the water depth
      h=15
!   g is he gravitational acceleration
      g=9.81
!  dt is the time step
      dt=0.1
!  tmax is the maximum time of scheme evaluation
      tmax=1500
!  xmax is the maximum x  value
      xmax=500 ! Enter approximately 10 times of biggest wavelength
!  xmin is the minimum x value
      xmin=0
!  ymax is the maximum y value
      ymax=500 ! Enter approximately 10 times of biggest wavelength
!  ymin is the minimum y value
      ymin=0

! Principal Direction Of Wave Propogation in Degrees
      TetaPr=45
! There are some other inputs to this code. For example fetch length and wind speed
! in specified for Jonswap spectrum as inputs at further lines.
! ------------------------------------------------------------------
! ------------------------------------------------------------------
! Calculate size of t array
      sizet=tmax/dt
! Calculate pi
      pi=3.141592653589793

! pi=4*atan(1.0) ! Alternative expression for pi
!  Calculate step size in x
      dx=(xmax-xmin)/N
!  Calculate wave length
      Lx=xmax-xmin
! Create Space Array
      x(1)=xmin
      do 10 j=1,N-1
      x(j+1)=x(j)+dx
```

```
 10   continue

!  Calculate step size in y
     dy=(ymax-ymin)/M
     Ly=ymax-ymin
! Create Space Array
     y(1)=ymin
     do 11 j=1,N-1
     y(j+1)=y(j)+dx
 11   continue

! Calculate fundamental wave number component in x and y directions
     dkx=2*pi/Lx
     dky=2*pi/Ly

! Calculate Wave Number Vector in x Direction
! Calculate Wave Number Vector in y Direction
     do 12 j=1,N/2+1
          kx(j)=dkx*(j-1)
          ky(j)=dky*(j-1)
          KXa(j,1)=kx(j)
          KYa(j,1)=ky(j)
 12    continue
     do 13 j=N/2+2,N
           kx(j)=dkx*(j-N-1)
           ky(j)=dky*(j-N-1)
           KXa(j,1)=kx(j)
           KYa(j,1)=ky(j)
 13    continue


! ------------------------------------------------
! -------------- Create Theta Matrix ---------------
! ------------------------------------------------
! ------------------------------------------------
      Teta(1,1)=0
      do 14 j=2,N/2+1
           Teta(j,1)=0
 14    continue
      do 15 j=N/2+2,N
           Teta(j,1)=180
 15    continue
```

```
      do 16 z=2,M/2+1
        do 17 j=1,N
          Teta(j,z)=180/pi*atan(ky(z)/kx(j))
17      continue
16    continue


     do 18 j=N/2+2,N
         do 19 z=2,M/2+1
           Teta(j,z)=180+Teta(j,z)
           Teta(1,z)=90
19       continue
18    continue

     do 20 j=2,N/2+1
        do 21 z=M/2+2,M
           Teta(j,z)=180/pi*atan(ky(z)/kx(j))
           Teta(1,z)=-90
21      continue
20    continue

     do 22 j=N/2+2,N
       do 23 z=M/2+2,M
          Teta(j,z)=-180+180/pi*atan(ky(z)/kx(j))
23      continue
22    continue
!-----------------------------------------------------------------




!!!! In order to activate Goda type spreading function uncomment the following 3
     blocks.
!!!! and comment out the next three blocks.
!!!! In this type of formulation directional spreading only depends on angle not
     frequency.

c$$$!-------------------------------------------------------------------------
c$$$! Calculate Wave Number Matrix k  using k^2=kx^2+ky^2
c$$$! Calculate Angular Frequency Matrix W using Linear dispersion relationship
c$$$! Calculate Group Velocity Cg
```

```
c$$$! Calculate Frequency Matrix
c$$$! Calculate kTANHkh here  in order to prevent calculation every time step
c$$$! Calculate Sf, frequency Spectrum, Goda et.al. 'Random Seas and Design of
c$$$! Maritime Structures' Second edition page 28
c$$$! Calculate wavenumber spectrum from frequency  Spectrum
c$$$! Hs is the significant wave height
c$$$      Hs=1
c$$$! Ts is the significant wave period
c$$$      Ts=15
c$$$       do 24 j=1,N
c$$$          do 25 z=1,M
c$$$             k(j,z)=sqrt(kx(j)**2+ky(z)**2)
c$$$             W(j,z)=sqrt(g*k(j,z)*tanh(k(j,z)*h))
c$$$             Cg(j,z)=0.5*(1+2*k(j,z)*h/sinh(2*k(j,z)*h))*W(j,z)/k(j,z)
c$$$             fr(j,z)=W(j,z)/(2*pi)
c$$$             kTANHkh(j,z)=abs(k(j,z))*tanh(abs(k(j,z))*h)
c$$$             Sf(j,z)=0.257*(Hs**2)*(Ts**-4)
c$$$     &            *(fr(j,z)**-5)*exp(-1.03*(Ts*fr(j,z))**-4)
c$$$             Sk(j,z)=Cg(j,z)*Sf(j,z)/(2*pi)
c$$$ 25       continue
c$$$ 24    continue
c$$$
c$$$! ------------------------------------------------------
c$$$
c$$$! In order to avoid dividing by zeros and getting NaN s set arbitrary numbers for
     elements
c$$$! (1,1). These will all drop out since corresponding amplitude a(1,1)=0
c$$$       k(1,1)=1
c$$$       W(1,1)=1
c$$$       Cg(1,1)=0
c$$$       kTANHkh(1,1)=1
c$$$       Sw(1,1)=0
c$$$       Sk(1,1)=0
c$$$!-------------------------------------------------------------------------
c$$$! Mitsuyau Type Directional Spectrum as a function of azimuth(teta)
c$$$! Formulation for the spectrum is taken From Random Seas and Design of
c$$$! Maritime Structures, Goda et.al. second edition page 32 eq. 2.21 and
c$$$! eq. 2.23
c$$$      s=10
c$$$      call gamma(s+1,arg1)
c$$$      call gamma(2*s+1,arg2)
c$$$      Go=1/pi*(2**(2*s-1))*(arg1)**2/arg2
```

```
c$$$      call urand2D(F,N,M)
c$$$! Gt is the directional Spreading Function
c$$$! S(k,Teta)=S(k)G(Teta)=SkGt is the directional frequency spectrum
c$$$! a is the real amplitude matrix. Jacobian of transformation(k) is used
c$$$! to convert S(k,Teta) to S(dkx,dky)
c$$$! A_com is the complex amplitude matrix
c$$$! B_com is the complex water surface fluctuation coeff. matrix
c$$$! A_com_org is the complex amplitude matrix derived directly from a
c$$$! and A_com will be created by taking the tranpose of the A_com_org and
c$$$! adding it to A_com_org.  Same thing for B_com.
c$$$! Also at Nyquist Limit components of the complex amplitude matrix has to be
     !! real
c$$$! and symmetric
c$$$      do 26 j=1,N
c$$$        do 27 z=1,M
c$$$          Gt(j,z)=Go*(cos((pi/180*(Teta(j,z)-TetaPr))/2))**(2*s)
c$$$          SkGt(j,z)=Sk(j,z)*Gt(j,z)
c$$$          a(j,z)=sqrt(SkGt(j,z)*dkx*dky*2/k(j,z))
c$$$          A_com_org(j,z)=a(j,z)*exp(i*F(j,z))/2
c$$$          B_com_org(j,z)=-i*g/W(j,z)*A_com_org(j,z)
c$$$ 27    continue
c$$$ 26   continue
c$$$
c$$$! ------------------------------------------------------------------------

!!! Use Jonswap Spectrum and sech^2 type spreading function given by Donelan,
     Hamilton and Hui

!---------------------------------------------------------------

! Calculate Wave Number Matrix k  using k^2=kx^2+ky^2
! Calculate Angular Frequency Matrix W using Linear dispersion relationship
! Calculate Group Velocity Cg
! Calculate Frequency Matrix
! Calculate kTANHkh here  in order to prevent calculation every time step
! Calculate Sw, angular frequency spectrum, using Jonswap spectra. Taken from
! Bob Heitsenrether's thesis.
! Calculate wavenumber spectrum from angular frequency Spectrum

      Fetch=18520     ! Fetch Length taken= 10 nautical miles
      U=15            ! Wind speed in meters/ second
      gam=3.3         ! constant for jonswap spectrum
```

```
      Wp=7*pi*g/U*((g*Fetch/(U**2))**-0.33)  ! Peak angular frequency
      alpha=0.076*((g*Fetch/U)**-0.22)
      cons=alpha*(g**2)

      do 24 j=1,N
        do 25 z=1,M
          k(j,z)=sqrt(kx(j)**2+ky(z)**2)
          W(j,z)=sqrt(g*k(j,z)*tanh(k(j,z)*h))
          Cg(j,z)=0.5*(1+2*k(j,z)*h/sinh(2*k(j,z)*h))*W(j,z)/k(j,z)
          fr(j,z)=W(j,z)/(2*pi)
          kTANHkh(j,z)=abs(k(j,z))*tanh(abs(k(j,z))*h)
          if (W(j,z).le.Wp) then
          Sw(j,z)=cons*(W(j,z)**-5)*exp(-1.25*(W(j,z)/Wp)**-4)*
     &          (gam**exp(-((W(j,z)-Wp)**2)/(2*(0.07*Wp)**2)))
          elseif (W(j,z).gt.Wp) then
           Sw(j,z)=cons*(W(j,z)**-5)*exp(-1.25*(W(j,z)/Wp)**-4)*
     &          (gam**exp(-((W(j,z)-Wp)**2)/(2*(0.09*Wp)**2)))
          endif
          Sk(j,z)=Cg(j,z)*Sw(j,z)
 25      continue
 24    continue
!-------------------------------------------------------------------------------

! In order to avoid dividing by zeros and getting NaN s set arbitrary numbers for
      elements
! (1,1). These will all drop out since corresponding amplitude a(1,1)=0
       k(1,1)=1
       W(1,1)=1
       Cg(1,1)=0
       kTANHkh(1,1)=1
       Sw(1,1)=0
       Sk(1,1)=0
! -------------------------------------------------------------------------------

! sech^2 type directional spreading function. Reference " Directional Spectra of wind
 ! generated waves", Donelan M.A., Hamilton J., and Hui, W.H.
      Wnormal=W/Wp
      do 26 j=1,N
       do 27 z=1,M
        if ((Wnormal(j,z).le.0.56).or.(Wnormal(j,z).ge.1.6)) then
           Beta(j,z)=1.24
         elseif ((Wnormal(j,z).gt.0.56).or.(Wnormal(j,z).lt.0.95)) then
```

```
          Beta(j,z)=2.61*(Wnormal(j,z)**1.3)
        elseif ((Wnormal(j,z).gt.0.95).or.(Wnormal(j,z).lt.1.6)) then
          Beta(j,z)=2.28*(Wnormal(j,z)**-1.3)
        endif
 27     continue
 26   continue

      call urand2D(F,N,M)
! Gt is the directional Spreading Function
! S(k,Teta)=S(k)G(Teta)=SkGt is the directional frequency spectrum
! a is the real amplitude matrix. Jacobian of transformation(k) is used
! to convert S(k,Teta) to S(dkx,dky)
! A_com is the complex amplitude matrix
! B_com is the complex water surface fluctuation coeff. matrix
! A_com_org is the complex amplitude matrix derived directly from a
! and A_com will be created by taking the tranpose of the A_com_org and
! adding it to A_com_org.  Same thing for B_com.
! Also at Nyquist Limit components of the complex amplitude matrix has to be real
! and symmetric
      do 28 j=1,N
        do 29 z=1,M
          Gt(j,z)=Beta(j,z)/2*
     &          (cosh(Beta(j,z)*(pi/180*(Teta(j,z)-TetaPr))))**-2
          SkGt(j,z)=Sk(j,z)*Gt(j,z)
          a(j,z)=sqrt(SkGt(j,z)*dkx*dky*2/k(j,z))
          A_com_org(j,z)=a(j,z)*exp(i*F(j,z))/2
          B_com_org(j,z)=-i*g/W(j,z)*A_com_org(j,z)
 29     continue
 28   continue

! ---------------------------------------------------------------------


! Calculate real and then complex amplitudes from spectrum, force Nyquist limits
! to be real numbers
!----------------------------------------------------------------------
      a(1,1)=0
      A_com_org(1,1)=0
      B_com_org(1,1)=0

      A_com=A_com_org
      B_com=B_com_org
```

```
    do 30 j=2,N
      do 31 z=2,M/2
        A_com(j,z)=A_com_org(j,z)+conjg(A_com_org(N-j+2,M-z+2))
        B_com(j,z)=B_com_org(j,z)+conjg(B_com_org(N-j+2,M-z+2))
31      continue
      do 32 z=M/2+2,M
        A_com(j,z)=A_com_org(j,z)+conjg(A_com_org(N-j+2,M-z+2))
        B_com(j,z)=B_com_org(j,z)+conjg(B_com_org(N-j+2,M-z+2))
32    continue
30  continue

    do 33  z=M/2+2,M
       A_com(1,z)=conjg(A_com(1,M-z+2))
       B_com(1,z)=conjg(B_com(1,M-z+2))
33   continue

    do 34 j=N/2+2,N
       A_com(j,1)=conjg(A_com(N-j+2,1))
       B_com(j,1)=conjg(B_com(N-j+2,1))
34  continue


    do 35 z=1,M
      A_com(N/2+1,z)=abs(A_com(N/2+1,z))
      B_com(N/2+1,z)=abs(B_com(N/2+1,z))
35  continue

    do 36 j=1,N
      A_com(j,M/2+1)=abs(A_com(j,M/2+1))
      B_com(j,M/2+1)=abs(B_com(j,M/2+1))
36  continue

    do 37 j=N/2+2,N
     A_com(j,M/2+1)=conjg(A_com(N-j+2,M/2+1))
     B_com(j,M/2+1)=conjg(B_com(N-j+2,M/2+1))
37  continue
!---------------------------------------------------------------------
```

```
!   Construct Initial Water Surface Fluctuation and
!   Initial Surface Velocity Potential
!-----------------------------------
      call ifft2(A_com,p1,p2,N,M)
      Wsf_tr=A_com
      call ifft2(B_com,p1,p2,N,M)
      Qs_tr=B_com
!-----------------------------------


!   Take transpozes of the Water Surface Fluctuation and the Surface Velocity Potential
!-------------------------------------------------------------------------------------
      do 90 j=1,N
        do 91 z=1,M
        Wsf(j,z)=Wsf_tr(z,j)
        Qs(j,z)=Qs_tr(z,j)
 91     continue
 90    continue
!-------------------------------------------------

!    Need initial variance to examine if the code is working properly
!    Area under the spectrum is total variance. Use Simpsons Rule.
! ----------------------------------------
      wa=0    ! Lower boundary for integration
      wb=10   ! Upper boundary for integration
      step1=(Wp-wa)/40000
      step2=(wb-Wp)/40000
      Sumodd1=0
      Sumeven1=0
      Sumodd2=0
      Sumeven2=0
      do 92 j=1,40001
       waxis1(j)=step1*(j-1)
       waxis2(j)=Wp+step2*(j-1)
       func1(j)=cons*(waxis1(j)**-5)*exp(-1.25*(waxis1(j)/Wp)**-4)*
     &          (gam**exp(-((waxis1(j)-Wp)**2)/(2*(0.07*Wp)**2)))
       func2(j)=cons*(waxis2(j)**-5)*exp(-1.25*(waxis2(j)/Wp)**-4)*
     &          (gam**exp(-((waxis2(j)-Wp)**2)/(2*(0.09*Wp)**2)))

 92   continue
      func1(1)=0
      do 93 j=1,39999
```

```
      Sumeven1=(1-mod(j,2))*2*step1/3*func1(j)+Sumeven1
      Sumodd1=mod(j,2)*4*step1/3*func1(j)+Sumodd1
      Sumeven2=(1-mod(j,2))*2*step2/3*func2(j)+Sumeven2
      Sumodd2=mod(j,2)*4*step2/3*func2(j)+Sumodd2

 93   continue
      I_var=step1/3*(func1(40001)+func1(1))+Sumodd1+Sumeven1+
     &       step2/3*(func2(40001)+func2(1))+Sumodd2+Sumeven2
!-------------------------------------------------------

!    These sums are going to be used in calculation of the variances
      sum1=0
      sum2=0
      sum3=0
      sum4=0
      sum5=0
      sum6=0
      sum7=0
      sum8=0

! Here Begins The Time Stepping. Time Integrator Is Fourth Order Runge-Kutta
      do 38  e=1,sizet

           !  First Slopes For Qs(surface velocity potential) and n
           !  water surface fluctuation are denoted as k1 and m1
           Qrhs1=Qs
           ! Calculate RHS of evaluation equation for 1st equation
           ! After operations Qrhs1 is changing!!
           call fft2(Qrhs1,p1,p2,N,M)
           do 39 j=1,N
             do 40 z=1,M
               Qrhs1(j,z)=Qrhs1(j,z)*kTANHkh(j,z)
 40          continue
 39          continue
           call ifft2(Qrhs1,p1,p2,N,M)
           m1=Qrhs1
           do 41 j=1,N
              do 42 z=1,M
                k1(j,z)=-g*Wsf(j,z)
                Qrhs2(j,z)=Qs(j,z)+0.5*dt*k1(j,z)
 42             continue
 41        continue
```

```
      ! Second Slopes For Qs(surface velocity potential) and n
      ! water surface fluctuation are denoted as k2 and m2
      ! Calculate RHS of evaluation equation for 2nd equation
      ! After operations Qrhs2 is changing!!
      call fft2(Qrhs2,p1,p2,N,M)
      do 45 j=1,N
        do 46 z=1,M
         Qrhs2(j,z)=Qrhs2(j,z)*kTANHkh(j,z)
46        continue
45      continue
      call ifft2(Qrhs2,p1,p2,N,M)
      m2=Qrhs2
      do 47 j=1,N
        do 48 z=1,M
        k2(j,z)=-g*(Wsf(j,z)+0.5*dt*m1(j,z))
        Qrhs3(j,z)=Qs(j,z)+0.5*dt*k2(j,z)
48        continue
47      continue

      ! Third Slopes For Qs(surface velocity potential) and n
      ! water surface fluctuation are denotes as k3 and m3
      ! Calculate RHS of evaluation equation for 2nd equation
      ! After operations Qrhs2 is changing!!
      call fft2(Qrhs3,p1,p2,N,M)
      do 51 j=1,N
       do 52 z=1,M
       Qrhs3(j,z)=Qrhs3(j,z)*kTANHkh(j,z)
52        continue
51      continue
      call ifft2(Qrhs3,p1,p2,N,M)
      m3=Qrhs3
      do 53  j=1,N
        do 54 z=1,M
        k3(j,z)=-g*(Wsf(j,z)+0.5*dt*m2(j,z))
        Qrhs4(j,z)=Qs(j,z)+dt*k3(j,z)
54        continue
53     continue

      ! Fourth  Slopes For Qs(surface velocity potential) and n
      ! water surface fluctuation are denoted as k4 and m4
      ! Calculate RHS of evaluation equation for 2nd equation
```

```
          ! After operations Qrhs2 is changing!!
          call fft2(Qrhs4,p1,p2,N,M)
          do 57 j=1,N
            do 58 z=1,M
            Qrhs4(j,z)=Qrhs4(j,z)*kTANHkh(j,z)
 58          continue
 57         continue
          call ifft2(Qrhs4,p1,p2,N,M)
!    Calculate last slopes and water surface fluctuation and
!    surface velocity potential for next time step inside same loop
!    Also store water surface fluctuation into a file named WSf2D
           m4=Qrhs4
 !     open (unit=9,file="WSF2D.txt",ACCESS='APPEND')

          do 59 j=1,N
            do 60 z=1,M
             k4(j,z)=-g*(Wsf(j,z)+dt*m3(j,z))
         Wsf(j,z)=Wsf(j,z)+dt/6*(m1(j,z)+2*m2(j,z)+2*m3(j,z)+m4(j,z))
         Qs(j,z)=Qs(j,z)+dt/6*(k1(j,z)+2*k2(j,z)+2*k3(j,z)+k4(j,z))
!         write(9,1) Wsf(j,z)
!  1       format (f8.5,f8.5)
 60            continue
 59         continue

 !     close(9)


c$$$!--------------------------------------------------------------------------------
c$$$          ! Following lines are for calculating velocity component in the normal
c$$$          ! direction which is going to be important in the Doppler shift
c$$$          ! calculations of the acoustic rays reflected from the ocean surface
c$$$
c$$$          ! Qsz is the z derivative of Qs; vertical velocity calculate from spectral
c$$$          ! formulation
c$$$          ! Qsx is the x derivative of Qs, horizontal velocity
c$$$          ! Qsy is the y derivative of Qs, horizontal velocity
c$$$          ! Wsfx is the x derivative of water surface fluc. Wsf
c$$$          ! Wsfy is the y derivative of water surface fluc. Wsf
c$$$          Qsz=Qs
c$$$          Qsx=Qs
c$$$          Qsy=Qs
c$$$          Wsfx=Wsf
```

```
c$$$          Wsfy=Wsf
c$$$          call fft2(Qsz,p1,p2,N,M)
c$$$          call fft2(Qsx,p1,p2,N,M)
c$$$          call fft2(Qsy,p1,p2,N,M)
c$$$          call fft2(Wsfx,p1,p2,N,M)
c$$$          call fft2(Wsfy,p1,p2,N,M)
c$$$          do 61 j=1,N
c$$$            do 62 z=1,M
c$$$            Qsz(j,z)=Qsz(j,z)*kTANHkh(j,z)
c$$$            Qsx(j,z)=Qsx(j,1)*KXa(j,1)*i
c$$$            Qsy(j,z)=Qsy(j,z)*KYa(j,1)*i
c$$$            Wsfx(j,z)=Wsfx(j,z)*KXa(j,1)*i
c$$$            Wsfy(j,z)=Wsfy(j,z)*KYa(j,1)*i
c$$$ 62           continue
c$$$ 61        continue
c$$$
c$$$         do 63 z=1,M
c$$$         Qsx(N/2+1,z)=abs(Qsx(N/2+1,z))
c$$$         Qsy(N/2+1,z)=abs(Qsy(N/2+1,z))
c$$$         Wsfx(N/2+1,z)=abs(Wsfx(N/2+1,z))
c$$$         Wsfy(N/2+1,z)=abs(Wsfy(N/2+1,z))
c$$$ 63      continue
c$$$        do 64 j=1,N
c$$$         Qsx(j,M/2+1)=abs(Qsx(j,M/2+1))
c$$$         Qsy(j,M/2+1)=abs(Qsy(j,M/2+1))
c$$$         Wsfx(j,M/2+1)=abs(Wsfx(j,M/2+1))
c$$$         Wsfy(j,M/2+1)=abs(Wsfy(j,M/2+1))
c$$$ 64      continue
c$$$
c$$$          call ifft2(Qsz,p1,p2,N,M)
c$$$          call ifft2(Qsx,p1,p2,N,M)
c$$$          call ifft2(Qsy,p1,p2,N,M)
c$$$          call ifft2(Wsfx,p1,p2,N,M)
c$$$          call ifft2(Wsfy,p1,p2,N,M)
c$$$           Qsz=real(Qsz)  ! Avoid roundoff  errors in the imaginary part
c$$$           Qsx=real(Qsx)  ! Avoid roundoff errors in the imaginary part
c$$$           Qsy=real(Qsy)  ! Avoid roundoff errors in the imaginary part
c$$$           Wsfx= real(Wsfx) ! Avoid roundoff errors in the imaginary part
c$$$           Wsfy= real(Wsfy) ! Avoid roundoff errors in the imaginary part
c$$$
c$$$ ! udotn is the dot product of velocity u and outward normal gives the velocity
     component
```

```
c$$$ ! in the normal direction
c$$$          do 65 j=1,N
c$$$            do 66 z=1,M
c$$$          udotn(j,z)=(-Qsx(j,z)*Wsfx(j,z)-Qsy(j,z)*Wsfy(j,z)+Qsz(j,z))
c$$$     &               /sqrt(Wsfx(j,z)**2+Wsfy(j,z)**2+1)
c$$$ 66         continue
c$$$ 65       continue
c$$$!-------------------------------------------------------------------------------

! Calculate Variances at few different points
!---------------------------------------
      sum1=sum1+Wsf(3,8)**2
      sum2=sum2+Wsf(60,60)**2
      sum3=sum3+Wsf(90,3)**2
      sum4=sum4+Wsf(110,235)**2
      sum5=sum5+Wsf(145,82)**2
      sum6=sum6+Wsf(21,4)**2
      sum7=sum7+Wsf(12,250)**2
      sum8=sum8+Wsf(200,14)**2
!---------------------------------------
      print *,'it.=',e
 38   continue


     open (unit=10,file="N_M_dt_dx_dy.txt",ACCESS='APPEND')
     write(10,*) N ,M, dt, dx,dy
     close(10)


     ! Var1 is the Final variance of water surface fluctuation
     var1=sum1/sizet
     var2=sum2/sizet
     var3=sum3/sizet
     var4=sum4/sizet
     var5=sum5/sizet
     var6=sum6/sizet
     var7=sum7/sizet
     var8=sum8/sizet

     print *, 'Initial Var=', I_var
     print *, 'Var 1=',var1
     print *, 'Var 2=',var2
```

```
      print *, 'Var 3=',var3
      print *, 'Var 4=',var4
      print *, 'Var 5=',var5
      print *, 'Var 6=',var6
      print *, 'Var 7=',Var7
      print *, 'Var 8=',Var8
      stop
      end

! ----------------------------------------
! -------------  SUBROUTINES --------------
!----------------------------------------

!----- Simple random number generator------
      subroutine urand2D(F,N,M)
      integer  N,j, seed
      real pi,old,c,d
      real F(N,M)
      PI=3.141592653589793
!     pi=4*atan(1.0) ! Alternative expression for pi
      seed=11
      old=seed
      j=1
      do 1 j=1,N
        do 2 z=1,M-1,2
         c=mod ((57*old+1),2*pi)
         d=mod ((57*c+1),2*pi)
         old=d
         F(j,z)=c
         F(j,z+1)=d
 2      continue
 1    continue
      return
      end
!--- End of simple random number generator -----

!-------- 1-D Fast Fourier Transfrom----------
! N=2^p, A is the complex array to be fourier transformed
      subroutine fft(A,p,N)
      complex A(N),U,W,T
!  Divide all elements by N
      do 1 J=1,N
```

```
 1    A(J)=A(J)/N
!  Reorder Sequence According to fig 12.8 of Newland
      NBD2=N/2
      NBM1=N-1
      J=1
      do 4 L=1,NBM1
      if(L.ge.J) go to 2
      T=A(J)
      A(J)=A(L)
      A(L)=T
 2    K=NBD2
 3    if(K.ge.J) go to 4
      J=J-K
      K=K/2
      go to 3
 4    J=J+K
!   Calculate FFT acoording to fig 12.5
      PI=3.141592653589793
      do 6 M=1,p
      U=(1.0,0.0)
      ME=2**M
      K=ME/2
      W=CMPLX(COS(PI/K),-SIN(PI/K))
      do 6 J=1,K
      do 5 L=J,N,ME
      LPK=L+K
      T=A(LPK)*U
      A(LPK)=A(L)-T
 5    A(L)=A(L)+T
 6    U=U*W
      return
      end
! ----- End of 1-D Fast Fourier Transform---------


!-------- 1-D Inverse Fast Fourier Transform --------
! Taken from www.algarcia.org/nummeth/Fortran/ifft.f
! And normalization has been changed by multiplying with N rather than dividing by N
! Due to definition of Fourier Pair
      subroutine ifft(A,p,N)
      complex A(N)
!    Routine to compute inverse Fourier transform using FFT algorithm
```

```
      integer j
      do j=1,N
        A(j)=conjg(A(j))
      enddo

      call fft(A,p,N)

      do j=1,N
        A(j)=conjg(A(j))*N ! Compute conjugates and Normalize
      enddo
      return
      end
!-------- End of 1-D Inverse Fourier Transform --------

!-------- 2-D Fast Fourier Transfrom-------------
! N=2^p, C is the complex matrix to be fourier transformed
      subroutine fft2(C,p1,p2,N,M)
      complex A(N), B(M),C(N,M)
      do 2 K=1,N
      do 1 J=1,M
 1    B(J)=C(K,J)
      call fft(B,p2,M)
      do 2 J=1,M
 2    C(K,J)=B(J)
      do 4 K=1,M
      do 3 J=1,N
 3    A(J)=C(J,K)
      call fft(A,p1,N)
      do 4 J=1,N
 4    C(J,K)=A(J)
      return
      end
!---------- End of 2-D Fast Fourier Transform----------


!-------- 2-D Inverse Fast Fourier Transform ----------
! And normalization has been changed by multiplying with N rather than dividing by N
! Due to definition of Fourier Pair
      subroutine ifft2(C,p1,p2,N,M)
      integer N,M
      complex C(N,M)
!    Routine to compute inverse Fourier transform using FFT algorithm
```

```
      integer j
      do j=1,N
        do z=1,M
        C(j,z)=conjg(C(j,z))
        enddo
      enddo

      call fft2(C,p1,p2,N,M)

      do j=1,N
        do z=1,M
        C(j,z)=conjg(C(j,z))*N*M ! Compute conjugates and Normalize
        enddo
      enddo
      return
      end
!-------- End of 2-D Inverse Fourier Transform --------

!------------------ Gamma function ------------------
!       Taken from http://jin.ece.uiuc.edu/routines/routines.html
!       This program computes the gamma function
      subroutine gamma(s,GA)
      implicit double precision (A-H,O-Z)
      dimension G(26)
      PI=3.141592653589793D0
      if (s.EQ.INT(s)) then
        if (s.GT.0.0D0) then
          GA=1.0D0
          M1=s-1
          do 10 K=2,M1
10            GA=GA*K
        else
          GA=1.0D+300
        endif
      else
        if (DABS(s).GT.1.0D0) then
          Z=DABS(s)
          M=INT(Z)
          R=1.0D0
          do 15 K=1,M
15            R=R*(Z-K)
          Z=Z-M
```

```
      else
        Z=s
      endif
      DATA G/1.0D0,0.5772156649015329D0,
     &      -0.6558780715202538D0, -0.420026350340952D-1,
     &      0.166538613822915D0,-.421977345555443D-1,
     &      -.96219715278770D-2, .72189432466630D-2,
     &      -.11651675918591D-2, -.2152416741149D-3,
     &      .1280502823882D-3, -.201348547807D-4,
     &      -.12504934821D-5, .11330272320D-5,
     &      -.2056338417D-6, .61160950D-8,
     &      .50020075D-8, -.11812746D-8,
     &      .1043427D-9, .77823D-11,
     &      -.36968D-11, .51D-12,
     &      -.206D-13, -.54D-14, .14D-14, .1D-15/
      GR=G(26)
      do 20 K=25,1,-1
20       GR=GR*Z+G(K)
      GA=1.0D0/(GR*Z)
      if (DABS(s).GT.1.0D0) then
        GA=GA*R
        if (s.LT.0.0D0) GA=-PI/(s*GA*DSIN(PI*s))
      endif
    endif
    return
    end
!------------ End of Gamma Function ----------------
```

5. Matlab-2D Nonlinear Wave Model

```
%  Reference Article: Douglas G.DOMMERMUTH and Dick K.P.YUE
%  A High-Order Spectral Method for the study of
%   nonlinear gravity waves  Cambridge, MA, USA 1986
% Two step procedure; pseudospectral method for determining nodal amplitudes
% and fourth order Runge-Kutta time integrator
% James KIRBY, Cihan BAYINDIR, University of Delaware, 18/August/2008

% Nonlinear Version-single wave train; Working in metric units

clear all
```

```
close all
h=200;          % Water Depth
g=9.81;         % Gravitational acceleration
dt=0.1;         % Time step
tmax=100;       % Maximum Time Of Scheme Evaluation
t=0:dt:tmax;    % Time array
Time=length(t);   % Length of Time Array

xmin=0;         % Minimum x value
N=2^8;          % Number of Fourier points;
                % a power of 2 for fast computation
M=10;           % Arbitrary Order in Wave Steepness

xmax=100;       % Wave length in meters
dx=xmax/(N-1);    % Step size in x
x=xmin:dx:xmax;   % Space array
L=xmax-xmin+dx;   % Length of periodic domain=wave length

k=2*pi/L;             % Wave Number
W=sqrt(g*k*tanh(k*h));  % Wave Frequency
                % using linear dispersion relationship

kTANHkh=k*tanh(k*h);    % In order to avoid calculation at every time step
                % product assigned to a variable
filter1=k*(0:1:N/2);
filter2=k*(-N/2+1:1:-1);
filter=[filter1,filter2];

a=0.1;    % Amplitude of single wave
Im=i;   % imaginary number


% Initial values for progressive wave; (Initial values for standing wave  are: n is the
same and Qs=0)
n(1:N)=a*cos(9*k*x)+0.5*a*cos(7*k*x-pi/4)+0.5*a*cos(11*k*x-pi/4);        % Initial
%Values for water surface fluctuation [Side bands example]
Qs(1:N)=a*g/sqrt(g*9*k*tanh(9*k*h)).*exp(9*k*n).*sin(9*k*x)+0.5*a*g/sqrt(g*7*k
*tanh(7*k*h)).*exp(7*k*n).*sin(7*k*x-
pi/4)+0.5*a*g/sqrt(g*11*k*tanh(11*k*h)).*exp(11*k*n).*sin(11*k*x-pi/4);
 % Initial Values for surface velocity potential
%Qs(1,1:N)=0; %Initial Values for standing wave
```

```
for i=1:1:Time;

   %% Runge-Kutta Time Integration with Pseudospectral Method %%

   X=Term(Qs,n,M,filter,h);       % Calculates the double-summed big term in the
evolution equations
   Qsx=ifft(fft(Qs).*Im.*filter); % x derivative of Qs
   nx=ifft(fft(n).*Im.*filter);   % x derivative of n

   % First Slope for Qs(surface vel.pot)
   k1=-g.*n-0.5*dealias(Qsx,Qsx)+0.5*(1+dealias(nx,nx)).*dealias(X,X);
   % First Slope for n(water surface fluct.)
   m1=-dealias(Qsx,nx)+(1+dealias(nx,nx)).*X;

   X=Term(Qs+0.5*dt.*k1,n+0.5*dt.*m1,M,filter,h); % Calculates the double-
  %summed big term in the evolution equations
   Qsx=ifft(fft(Qs+0.5*dt.*k1).*Im.*filter);   % x derivative of Qs
   nx=ifft(fft(n+0.5*dt.*m1).*Im.*filter);     % x derivative of n

   % Second Slope for Qs(surface vel.pot)
   k2=-g.*(n+0.5*dt.*m1)-
   0.5*dealias(Qsx,Qsx)+0.5*(1+dealias(nx,nx)).*dealias(X,X);
   % Second Slope for n(water surface fluct.)
   m2=-dealias(Qsx,nx)+(1+dealias(nx,nx)).*X;


   X=Term(Qs+0.5*dt.*k2,n+0.5*dt.*m2,M,filter,h); % Calculates the double-
   % summed big term in the evolution equations
   Qsx=ifft(fft(Qs+0.5*dt.*k2).*Im.*filter);    % x derivative of Qs
   nx=ifft(fft(n+0.5*dt.*m2).*Im.*filter);      % x derivative of n

   % Third Slope for Qs(surface vel.pot)
   k3=-g.*(n+0.5*dt.*m2)-
    0.5*dealias(Qsx,Qsx)+0.5*(1+dealias(nx,nx)).*dealias(X,X);
   % Third Slope for n(water surface fluct.)
   m3=-dealias(Qsx,nx)+(1+dealias(nx,nx)).*X;


   X=Term(Qs+dt.*k3,n+dt.*m3,M,filter,h);
   Qsx=ifft(fft(Qs+dt.*k3).*Im.*filter);   % x derivative of Qs
   nx=ifft(fft(n+dt.*m3).*Im.*filter);     % x derivative of n
```

```
   % Fourth Slope for Qs(surface vel.pot)
   k4=-g.*(n+dt*m3)-0.5*dealias(Qsx,Qsx)+0.5*(1+dealias(nx,nx)).*dealias(X,X);
   % Fourth Slope for n(water surface fluct.)
   m4= -dealias(Qsx,nx)+(1+dealias(nx,nx)).*X;


   % In order to examine energy conservation keep track of the Fourier modes with
%given side band values
   FC=fft(n)/N;
   FourierCoef1(i,1)=2*abs(FC(8))/a;
   FourierCoef2(i,1)=2*abs(FC(10))/a;
   FourierCoef3(i,1)=2*abs(FC(12))/a;

%    %Smoothing filter
%      if mod(i,4)==0;
%      n=ifft(fft(n).*(5+4.*cos(pi.*abs(filter)./abs(filter(N/2+1)))-
cos(2*pi.*abs(filter)./abs(filter(N/2+1))))/8);
%      Qs=ifft(fft(Qs).*(5+4.*cos(pi.*abs(filter)./abs(filter(N/2+1)))-
cos(2*pi.*abs(filter)./abs(filter(N/2+1))))/8);
%      end

   %Evaluate n(water surface fluc.) and Qs(surface velocity pot.) for next time step
   n=n+dt/6.*(m1+2*m2+2*m3+m4);
   Qs=Qs+dt/6.*(k1+2*k2+2*k3+k4);

   % Plot the water surface fluctuation
   plot(x,n),title(['t=',num2str((i-1)*dt),'s']);
   xlim([xmin xmax])
   ylim([-3*a 3*a])
   xlabel('x[m]'),ylabel('Water Surface Fluctuation[m]')
   pause(0.05)
end

  % Plot the time histories of Fourier Coefficients
  figure(2)

plot(dt:dt:length(FourierCoef1)*dt,FourierCoef1,dt:dt:length(FourierCoef1)*dt,Fourier
Coef2,'red-.',dt:dt:length(FourierCoef1)*dt,FourierCoef3,'black--'),title('Time histories
of amplitudes'),legend('7k','9k','11k')

Initial_variance=0.0624*Hs^2
```

```
Final_Variance=var(n)
```

5.1 Function: Term.m

```
% This function calculates the Double Sum Term in the evolution equations
% 2.8 given in the page of 270 in D.G.Dommermuth and D.K.P.Yue's article
% James Kirby, Cihan Bayindir September 2008
function [X] = Term(Qs,n,M,filter,h)

% Inputs are Qs: surface velocity potential
%           n: water surface fluctuation
%           M: arbitrary order in wave steepness
%       filter: wavenumber values fitting into domain(filter)
%           h: water depth

   N=length(Qs);
   R(1,1:N)=Qs;
   Qm(1,1:N)=R(1,1:N);
   for m=2:1:M;
       R(m,1:N)=0;
      for j=1:1:(m-1);
         R(m,1:N)=R(m,1:N)-dealiasP(n,j)/factorial(j).*ifft(fft(Qm(m-j,1:N)).*((1-
               mod(j,2)).*(abs(filter)).^j+mod(j,2).*(abs(filter)).^j.*tanh(abs(filter)*h)));
      end
      Qm(m,1:N)=R(m,1:N);
   end

   % Calculate the long term at the most RHS of the evolution equations

   X(1:N)=0;
   for m=1:1:M;
      for j=0:1:M-m;
         X= X+dealiasP(n,j)/factorial(j).*ifft(fft(Qm(m,1:N)).*((1-
               mod(j,2)).*(abs(filter)).^(j+1).*tanh(abs(filter)*h)+mod(j,2).*(abs(filter)).
               ^(j+1)));
      end
   end
```

5.2 Function: dealias.m

```
function [w] = dealias(u,v)
% Function dealias.m
```

```
% This function computes the alias-free product of two series.
% Input is the two arrays to be multiplied, output is the alias-free
% product
% Anti-aliasing is based on the algorithm given by Canuto, Hussaini,
% Quarteroni, Zang 1988

N=length(u); % Which should be equal to the length of v
M=3*N/2;

uk=fft(u)/N; % Fourier Coefficients of u
UK(1:N/2+1)=uk(1:N/2+1);  UK(N/2+2:N+1)=0;  UK(N+2:M)=uk(N/2+2:N); % For
            region between N/2 and M/2 pad with zeros
U=ifft(UK)*M; %Get new U whose size is twice of initial input

vk=fft(v)/N; %Fourier Coefficients of v
VK(1:N/2+1)=vk(1:N/2+1);  VK(N/2+2:N+1)=0;  VK(N+2:M)=vk(N/2+2:N);  % For
            region between N/2 and M/2 pad with zeros
V=ifft(VK)*M; %Get new V whose size is twice of initial input
W=U.*V;

WK=fft(W)/M;
wk(1:N/2+1)=WK(1:N/2+1);    wk(N/2+2:N)=WK(N+2:M);  % Truncate back to
            original length
w=ifft(wk)*N;
```

5.3 Function: dealiasP.m

```
function [w] = dealiasP(u,Pow)
% Function dealiasP.m
% This function computes the alias-free power of a series.
% First input is the array to be taken the power and second input is a
% number which is power
% Output is the anti-aliased power.
%
% Anti-aliasing is based on the algorithm given by Canuto, Hussaini,
% Quarteroni, Zang 1988

N=length(u);
M=Pow*3*N/4;

if Pow==0;
   w(1:N)=1;
```

```
  elseif Pow==1;
   w=u;
  else

uk=fft(u)/N; % Fourier Coefficients of u
UK(1:N/2+1)=uk(1:N/2+1);  UK(N/2+2:M-N/2+1)=0;  UK(M-
N/2+2:M)=uk(N/2+2:N); % For region between N/2 and M/2 pad with zeros
  U=ifft(UK)*M; %Get new U whose size is twice of initial input

  W=U.^Pow;

  WK=fft(W)/M;
  wk(1:N/2+1)=WK(1:N/2+1);    wk(N/2+2:N)=WK(M-N/2+2:M);  % Truncate back
                                                          % to original length
  w=ifft(wk)*N;
 end
```

6. Linear and Nonlinear Coupled Wave-Acoustic Models

```
% Article: Douglas G.DOMMERMUTH and Dick K.P.YUE
% A High-Order Spectral Method for the study of
%  nonlinear gravity waves  Cambridge, MA, USA 1986
% Two step procedure; pseudospectral method for determining nodal amplitudes
% and fourth order Runge-Kutta time integrator
% Cihan BAYINDIR, James Kirby, University of Delaware, 02/04/2008

% Linear Version-Spectrum of Waves; Working in metric units
clear all
close all

% Load HFA sound data
%--------------------
 load sound; % location of matrix 'ss'; HFA 97 sound speed data
 % t is the hours, ss is the sounddata, hy?
 hours=t;
 sounddata=ss;
 hy=hy;
 clear t;% clear ss; clear hy; % Clear variables coming up from Bobs Thesis
 %--------------------
```

```
%Load HFA weather data
load Nweather2.dat
weather=Nweather2;

%--------------------
 read_init_conditions1;
 clear n; % Clear some variables coming up from Bobs codes
 read_init_conditions2;
 clear n2; % Clear some variables coming up from Bobs codes
%--------------------

% Load HFA tide data
load tide_vs_geotime;
AV_TIDE=(TIDE_A+TIDE_C)/2;
[dumind,tidein]=find(24<=TIDE_GEOTIME & TIDE_GEOTIME <=48); % Tide
%measurements started at 23 september 97 at 00
                    % We are interested in the 1 day duration
                    % between 24 September and 25 September
                    % therefore seek corresponding indices
tidehours=TIDE_GEOTIME(tidein)-24;
Avtide=AV_TIDE(tidein);

for q=1:1:29;   %1:1:length(hours);   % 24 hour loop for acoustic calculations

   [dummind,matchin]=find(abs(tidehours-hours(q))<0.05);
   h=Avtide(matchin);   % Water Depth, Bob Heitsenrether used h=15 meters
   h=15;

   g=9.81;                % Gravitational acceleration
   dtExperiments=0.345;    %0.345    %The value of timestepping used in HFA
%experiments
   dt=0.1725;             %0.1725;  % Time step. Enter a number which exactly divides
%dtExperiments
   multiple=dtExperiments/dt;
   tmax=3.45;       % Maximum Time Of Scheme Evaluation
               % 40s used for experiments
   t=0:dt:tmax;    % Time array
   Time=length(t);  % Length of Time Array
   N=2^10;         % Number of Fourier points;
               % a power of 2 for fast computation
   xmin=0;         % Minimum x value
```

```
   xmax=1000*receiver_rng; % Pick a domain length; the value used in HFA
%experiments is 389m
              % variable receiver_rng is coming from
              % read_init_conditions2 file in kms convert it to meters
   dx=(xmax-xmin)/N;    % Step size in x
   j=0:1:N-1;
   x=dx*j;             % Space array, Periodic domain
   Lmax=xmax-xmin;      % Length of periodic domain=max wave length
   k1=2*pi/Lmax*(0:1:N/2);      % First portion of Wavenumber array
   k2=2*pi/Lmax*(-N/2+1:1:-1);  % Second portion of Wavenumber array
   k=[k1,k2];                  % Combine first and second Portions of
                         % Wave Number Array
   filter=k;
   M=10; % Arbitrary order in wave steepness for nonlinear simulations

   W=sqrt(g*k1.*tanh(k1*h));  % wave angular frequency;
                     % using linear dispersion relationship
   Cg=0.5.*(1+2.*k1*h./sinh(2.*k1*h)).*W./k1; %Group velocity
   Cg(1)=0;

    time_in(1)=9;    % month of experiment data
    time_in(2)=24;   % day of experimental data
    time_in(3)=fix(hours(q));  % hour of experimental data
    time_in(4)=fix(60*(hours(q)-time_in(3))); % minute of experimental data
    time_in(5)=fix(60*(60*(hours(q)-fix(hours(q)))-time_in(4)));  % second of
  % experimental data
    if (time_in(3)==24);
       time_in(2)=time_in(2)+1;
       time_in(3)=0;
    end
    num2str(['Simulation for time : ',num2str(time_in)])
    WindSpeed=get_wind(time_in);
    WindSpeed=1+0.5*(q-1);

    create_env_file1; % Create first env file with eigenray option
    create_env_file2; % Create second env file with arrival 'A' option

    cs=SoundSpeedatSurface;
%-------------------------------------------------------------
% Formulation of Frequency Spectrum
% Frequency spectrum as a function of frequency
% Taken from master's thesis, Bob Heitsenrether, UD marine studies
```

```
% uses the Jonswap Spectrum given by Shore Protection Manual
%-------------------------------------------------------------
X=18520;      % Equals 10 nautical miles, 1nmi=1852 meters
U=WindSpeed;  % Wind Speed in meters per second; Bob used values between 5-13
gam=3.3;      % Parameter for Jonswap Spectrum
Wp=7*pi*g/U*(g*X/U^2)^-0.33; % Peak angular frequency
alpha=0.076*(g*X/U)^-0.22;   % constant of equation for S(W)
[dum,in]=find(W<=Wp);
mm=max(in);
SW1=alpha*g^2.*W(in).^-5.*exp(-5/4*(W(in)/Wp).^-4).*gam.^exp(-(W(in)-
Wp).^2/(2*0.07^2*Wp^2));
SW2=alpha*g^2.*W(mm+1:N/2+1).^-5.*exp(-5/4*(W(mm+1:N/2+1)/Wp).^-
4).*gam.^exp(-(W(mm+1:N/2+1)-Wp).^2/(2*0.09^2*Wp^2));
Sw=[SW1,SW2];
Sw(1)=0;
Sk=Cg.*Sw;     % Frequency Spectrum ==> Wavenumber Spectrum
Sk(1)=0;

% Following line is not necessary but it gives an idea about selection of
% xmax, Horizontal length scale
  fp=Wp/(2*pi);   Tp=1/fp;
% According to deep water formula L~1.56 T^2
% Make sure that xmax you choose is about at least 25 times of this value
% since it represents the spectrum without much error

a2=sqrt(2*pi*Sk*2/Lmax);   % Nodal amplitudes obtained from spectrum

F2(1:N/2+1)=2*pi*rand(1,N/2+1);   % Uniformly Distributed Random Number
 % generator for phase angles on the interval [0,2*pi]

kTANHkh=abs(k).*tanh(abs(k).*h);    % In order to avoid calculation at every time
                                    % step product assigned to a variable

W(1)=1; % Since a(1)=0 W(1) will drop out. No problem to change it.

%INVERSE FOURIER TRANSFORM METHOD
A2=a2(1:N/2+1).*exp(i*F2(1:N/2+1))/2;
A(1:N/2+1)=A2;  A(N/2+2:1:N)=conj(A2(N/2:-1:2));
A(N/2+1)=abs(A(N/2+1));
B(1:N/2+1)=-i*g./W.*A2;  B(N/2+2:1:N)=conj(-i*g./W(N/2:-1:2).*A2(N/2:-1:2));
B(N/2+1)=abs(B(N/2+1));
n=N*ifft(A);
```

```
Qs=N*ifft(B);

nNL=n; % Initial Conditions For Nonlinear Surface Fluctuation is Same for Linear
One
QsNL=Qs;  % Initial Conditions For Nonlinear Velocity Pot. is Same for Linear One

fortyind=1;
fortyindNL=1;
for j=1:1:Time;    %Time Indice


 %%%%%%%%%% LINEAR SIMULATION %%%%%%%%%%%%%%%%%%

   %% Runge-Kutta Time Integration with Pseudospectral Method %%

   % First Slope for Qs(surface vel.pot)
   d1=-g.*n;

   % First Slope for n(water surface fluct.)
   m1=ifft(fft(Qs).*kTANHkh);

   % Second Slope for Qs(surface vel.pot)
   d2=-g.*(n+0.5*dt.*m1);

   % Second Slope for n(water surface fluct.)
   m2=ifft(fft(Qs+0.5*dt.*d1).*kTANHkh);


   % Third Slope for Qs(surface vel.pot)
   d3=-g.*(n+0.5*dt.*m2);

   % Third Slope for n(water surface fluct.)
   m3=ifft(fft(Qs+0.5*dt.*d2).*kTANHkh);


   % Fourth Slope for Qs(surface vel.pot)
   d4=-g.*(n+dt*m3);

   % Fourth Slope for n(water surface fluct.)
   m4=ifft(fft(Qs+dt*d3).*kTANHkh);

   %Evaluate n(water surface fluc.) and Qs(surface velocity pot.) for next
```

```
   %time step
   n=n+dt/6.*(m1+2.*m2+2.*m3+m4);
   Qs=Qs+dt/6.*(d1+2.*d2+2.*d3+d4);

   % Following lines are for calculating velocity component in the normal
   % direction which is going to be important in the Doppler shift
   % calculations of the Acoustic rays reflected from ocean surface

   Qsz=ifft(fft(Qs).*kTANHkh);  % Vertical velocity
   FTQs=i*k.*fft(Qs);  FTQs(N/2+1)=abs(FTQs(N/2+1)); % At Nyquist freq.
   Qsx=ifft(FTQs);  % Horizontal velocity

   FTn=i*k.*fft(n);    FTn(N/2+1)=abs(FTn(N/2+1));
   nx=ifft(FTn);    % Slope of water surface fluctuation
   udotn=(-Qsx.*nx+Qsz)./sqrt(nx.^2+1); % Velocity in the normal direction

%%%%%%%%%%%%%%%%%%%%    RUN BELLHOP %%%%%%% %%%%%%%

% Run the Bellhop every 0.345 seconds in order to be compatible  with
% experiments.

if (mod((j-1),multiple))==0;
   num2str(['Running Bellhop : ',num2str((j-1)*dt),' s '])


%%%  WRITE Water Surface Fluctuation to Text File  %%%%%

  % Since we have already created the ocean model based on spectral
  % formulation given in Goda's book we need only to write the surface
  % fluctuation into an altimetry file which is going to be read by Bellhop
   fid=fopen('ATIFIL','w');
   % change units of horizontal surface
   fprintf(fid,'%s\n',('''L'''));  % First line of the Altimetry file
   fprintf(fid,'%d\n',length(x));
   for p=1:length(x);
      fprintf(fid,'%f %f\n',x(p)/1000,n(p));
   end
   fclose(fid);

 % Run Bellhop in order to generate RAYFIL and plot Eigenrays generated
    pause(0.01); % Pause 2 seconds in order to make sure that eyta and u dot n is
  % written into text file
```

```
    eval( [' ! bellhop.exe < envfile1.env ']);
    pause(0.01);  % Pause 2 seconds in order to make sure that Bellhop output is
written
    figure(1),subplot(2,1,1),
[TotalDopplerShiftArray]=plotray('RAYFIL',x,n,nx,Qsx,Qsz,freq,cs,receiver_depth),te
xt(5,1,strcat('{Mon., Day , Hour , Min.} :',num2str(time_in(1:4))))
    text(5,2,strcat('t = ',num2str((j-1)*dt),'s'))
    text(5,3,strcat('Wind Speed=',num2str(U),'m/s'))
    text(5,4,strcat('Tide=',num2str(h),'m'))
    hold on ,plot(x,n,'o-'),plot(1000*receiver_rng,h-receiver_height,'*') % Plot receiver
%location
    hold on, %plot(crossx,crossz,'<r')


 %%%%%%%%% READ THE ARRIVAL INFO %%%%% %%%%%%%%%
    % Run Bellhop second time in order to generate ARRFIL
    eval( [' ! bellhop.exe < envfile2.env ']);
    % Read the arrival time/amplitude data computed by BELLHOP
    %[ amp, delay, SrcAngle, RcvrAngle, NumTopBnc, NumBotBnc, narrmat, Pos ] =
read_arrivals_asc( ARRFIL, narrmx );
     ARRFIL='ARRFIL';
     narrmx=200; % Maximumn number of arrivals, taken from read_arrivals_asc.m
%written by Mike Porter
    [amp,delay,SrcAngle, RcvrAngle,NumTopBnc,NumBotBnc,narrmat,Pos]=
read_arrivals_asc( ARRFIL, narrmx );
    [di1,di2all]=find(NumTopBnc>=1 & NumBotBnc==0);
    [indd1,indd2]=find(delay(di2all)==min(delay(di2all))); % Select the ray with the
minimum delay time
    if (isempty(TotalDopplerShiftArray(indd2))==1);
       ArrivalTime(fortyind)=0;
       DS(fortyind)=0;
    else
      ArrivalTime(fortyind)=delay(di2all(indd2));
      DS(fortyind)=TotalDopplerShiftArray(indd2);
    end
    fortyind=fortyind+1;
end
  text(5,5,strcat('ArrivalTime=',num2str(delay(di2all(indd2))),'s'))
  text(5,6,strcat('D.S=',num2str(DS(fortyind-1)),'Hz'))

  %xlim([xmin xmax])
  %ylim([-2 2])
```

```
% subplot(2,1,2), plot(x,udotn); title('Velocity In Outward Normal Direction')
%pause(0.1)
 % j
 %hold on

 %%%%%%%%%%%%% NONLINEAR SIMULATION %%%%%%%%%%

 %% Runge-Kutta Time Integration with Pseudospectral Method %%

X=Term(QsNL,nNL,M,filter,h);       % Calculates the double-summed big term in
% the evolution equations
QsxNL=ifft(fft(QsNL).*i.*filter); % x derivative of Qs
QsxNL=real(QsxNL);
nxNL=ifft(fft(nNL).*i.*filter);   % x derivative of n
nxNL=real(nxNL);

% First Slope for Qs(surface vel.pot)
k1=-g.*nNL-
0.5*dealias(QsxNL,QsxNL)+0.5*(1+dealias(nxNL,nxNL)).*dealias(X,X);
% First Slope for n(water surface fluct.)
s1=-dealias(QsxNL,nxNL)+(1+dealias(nxNL,nxNL)).*X;


X=Term(QsNL+0.5*dt.*k1,nNL+0.5*dt.*s1,M,filter,h); % Calculates the double-
% summed big term in the evolution equations
QsxNL=ifft(fft(QsNL+0.5*dt.*k1).*i.*filter);   % x derivative of Qs
QsxNL=real(QsxNL);
nx=ifft(fft(nNL+0.5*dt.*s1).*i.*filter);     % x derivative of n
nxNL=real(nxNL);

% Second Slope for Qs(surface vel.pot)
k2=-g.*(nNL+0.5*dt.*s1)-
0.5*dealias(QsxNL,QsxNL)+0.5*(1+dealias(nxNL,nxNL)).*dealias(X,X);
% Second Slope for n(water surface fluct.)
s2=-dealias(QsxNL,nxNL)+(1+dealias(nxNL,nxNL)).*X;



X=Term(QsNL+0.5*dt.*k2,nNL+0.5*dt.*s2,M,filter,h); % Calculates the double

% summed big term in the evolution equations
QsxNL=ifft(fft(QsNL+0.5*dt.*k2).*i.*filter);    % x derivative of Qs
```

```
nxNL=ifft(fft(nNL+0.5*dt.*s2).*i.*filter);      % x derivative of n

% Third Slope for Qs(surface vel.pot)
k3=-g.*(nNL+0.5*dt.*s2)-
0.5*dealias(QsxNL,QsxNL)+0.5*(1+dealias(nxNL,nxNL)).*dealias(X,X);
% Third Slope for n(water surface fluct.)
s3=-dealias(QsxNL,nxNL)+(1+dealias(nxNL,nxNL)).*X;


X=Term(QsNL+dt.*k3,nNL+dt.*s3,M,filter,h);
QsxNL=ifft(fft(QsNL+dt.*k3).*i.*filter);   % x derivative of Qs
QsxNL=real(QsxNL);
nxNL=ifft(fft(nNL+dt.*s3).*i.*filter);     % x derivative of n
nxNL=real(nxNL);

% Fourth Slope for Qs(surface vel.pot)
k4=-g.*(nNL+dt*s3)-
0.5*dealias(QsxNL,QsxNL)+0.5*(1+dealias(nxNL,nxNL)).*dealias(X,X);
% Fourth Slope for n(water surface fluct.)
s4= -dealias(QsxNL,nxNL)+(1+dealias(nxNL,nxNL)).*X;

%Evaluate n(water surface fluc.) and Qs(surface velocity pot.) for next time step
nNL=nNL+dt/6.*(s1+2*s2+2*s3+s4);
QsNL=QsNL+dt/6.*(k1+2*k2+2*k3+k4);


% Following lines are for calculating velocity component in the normal
% direction which is going to be important in the Doppler shift
% calculations of the Acoustic rays reflected from ocean surface

X=Term(QsNL,nNL,M,filter,h);  % Vertical velocity
FTQsNL=i*k.*fft(QsNL);  FTQsNL(N/2+1)=abs(FTQsNL(N/2+1)); % At Nyquist
% freq.
QsxNL=ifft(FTQsNL);  % Horizontal velocity

FTnNL=i*k.*fft(nNL);    FTnNL(N/2+1)=abs(FTnNL(N/2+1));
nxNL=ifft(FTnNL);    % Slope of water surface fluctuation
udotnNL=(-QsxNL.*nxNL+X)./sqrt(nxNL.^2+1); % Velocity in the normal
direction


%%%%%%%%%%%%%%%%%%%%   RUN BELLHOP %%%%%%%%%%%%
```

```
% Run the Bellhop every 0.345 seconds in order to be compatible  with
% experiments.

if (mod((j-1),multiple))==0;
   num2str(['Running Bellhop : ',num2str((j-1)*dt),' s '])

%%%  WRITE Water Surface Fluctuation to Text File   %%%
  % Since we have already created the ocean model based on spectral
  % formulation given in Goda's book we need only to write the surface
  % fluctuation into an altimetry file which is going to be read by Bellhop
   fid=fopen('ATIFIL','w');
   % change units of horizontal surface
   fprintf(fid,'%s\n',('''L'''));  % First line of the Altimetry file
   fprintf(fid,'%d\n',length(x));
   for p=1:length(x);
      fprintf(fid,'%f %f\n',x(p)/1000,nNL(p));
   end
   fclose(fid);

 % Run Bellhop in order to generate RAYFIL and plot Eigenrays generated
    pause(0.01); % Pause 2 seconds in order to make sure that eyta and u dot n is
  written into text file
    eval( [' ! bellhop.exe < envfile1.env ']);
    pause(0.01);  % Pause 2 seconds in order to make sure that Bellhop output is
 written
    figure(1),subplot(2,1,2),
[TotalDopplerShiftNLArray]=plotray('RAYFIL',x,nNL,nxNL,QsxNL,X,freq,cs,receive
r_depth),text(5,1,strcat('{Mon., Day , Hour , Min.} :',num2str(time_in(1:4))))
    text(5,2,strcat('t = ',num2str((j-1)*dt),'s'))
    text(5,3,strcat('Wind Speed=',num2str(U),'m/s'))
    text(5,4,strcat('Tide=',num2str(h),'m'))
    hold on ,plot(x,n,'o-'),plot(1000*receiver_rng,h-receiver_height,'*') % Plot receiver
location
    hold on, %plot(crossx,crossz,'<r')


%%%%%%%%%%%%%%%% READ THE ARRIVAL INFO %%%%%%%%%%
    % Run Bellhop second time in order to generate ARRFIL
    eval( [' ! bellhop.exe < envfile2.env ']);
    % Read the arrival time/amplitude data computed by BELLHOP
```

```
    %[ amp, delay, SrcAngle, RcvrAngle, NumTopBnc, NumBotBnc, narrmat, Pos ] =
read_arrivals_asc( ARRFIL, narrmx );
      ARRFIL='ARRFIL';
      narrmx=200; % Maximumn number of arrivals, taken from read_arrivals_asc.m
%written by Mike Porter
     [ampNL,delayNL,SrcAngleNL,
RcvrAngleNL,NumTopBncNL,NumBotBncNL,narrmatNL,PosNL]=
read_arrivals_asc( ARRFIL, narrmx );
     [di1NL,di2allNL]=find(NumTopBncNL>=1 & NumBotBncNL==0);
     [indd1NL,indd2NL]=find(delayNL(di2allNL)==min(delayNL(di2allNL)));
% Select the ray with the minimum delay time
     if (isempty(TotalDopplerShiftNLArray(indd2NL))==1);
        ArrivalTimeNL(fortyindNL)=0;
        DSNL(fortyindNL)=0;
     else
       ArrivalTimeNL(fortyindNL)=delayNL(di2allNL(indd2NL));
       DSNL(fortyindNL)=real(TotalDopplerShiftNLArray(indd2NL));
     end
     fortyindNL=fortyindNL+1;
end
     text(5,5,strcat('ArrivalTime=',num2str(delayNL(di2allNL(indd2NL))),'s'))
     text(5,6,strcat('D.S=',num2str(DSNL(fortyindNL-1)),'Hz'))
%     % Plot the water surface fluctuation
%     plot(x,nNL),%title(['t=',num2str((j-1)*dt),'s']);
%     xlim([xmin xmax])
%     ylim([-0.5 0.5])
%     xlabel('x[m]'),ylabel('Water Surface Fluctuation[m]')
%     pause(0.05)
%      hold off
end


   figure(2),
   [nzi1,nzi2]=find(DS>0);
   tt=0:dtExperiments:tmax;
   subplot(2,1,1),plot(tt(nzi2),ArrivalTime(nzi2)),xlabel('Geotime(s)'),ylabel('Arrival
Time Of Eigenray(s)')
   text(1,0.2592,strcat('Start Time {Mon., Day , Hour , Min.}
:',num2str(time_in(1:4))))
   text(1,0.2590,strcat('Wind Speed=',num2str(U),'m/s'))
   text(7,0.2590,strcat('Tide=',num2str(h),'m'))
```

```
   subplot(2,1,2),plot(tt(nzi2),DS(nzi2)),xlabel('Geotime(s)'),ylabel('Doppler
Shift(Hz)')

   stdArrivaltime(q)=std(ArrivalTime(nzi2));
   WindSpeedseries(q)=WindSpeed;
   meanDS(q)=mean(DS(nzi2));


   figure(3)
   [nzi1NL,nzi2NL]=find(DSNL>0);
   tt=0:dtExperiments:tmax;
   subplot(2,1,1),
plot(tt(nzi2NL),ArrivalTimeNL(nzi2NL)),xlabel('Geotime(s)'),ylabel('Arrival Time Of
Eigenray(s)')
   text(1,0.2592,strcat('Start Time {Mon., Day , Hour , Min.}
:',num2str(time_in(1:4))))
   text(1,0.2590,strcat('Wind Speed=',num2str(U),'m/s'))
   text(7,0.2590,strcat('Tide=',num2str(h),'m'))
   subplot(2,1,2),plot(tt(nzi2NL),DS(nzi2NL)),xlabel('Geotime(s)'),ylabel('Doppler
Shift(Hz)')

   stdArrivaltimeNL(q)=std(ArrivalTimeNL(nzi2NL));
   meanDSNL(q)=mean(DSNL(nzi2NL));

end
   load all_HFA97_std;
   Windspeedex=all_std(:,3);
   STDarrivalex=all_std(:,2);
   figure(4)

plot(WindSpeedseries,stdArrivaltime,'black',WindSpeedseries,stdArrivaltimeNL,'blac
k--',Windspeedex,STDarrivalex,'x black'),xlabel('WindSpeed(m/s)'),ylabel('Standard
Dev. of Arrival Time (s)');
   legend('Linear Simulations','Nonlinear Simulations','Experiments')
   figure(5)
   plot(WindSpeedseries,meanDS,'black',WindSpeedseries,meanDSNL,'black--
'),xlabel('WindSpeed(m/s)'),ylabel('Average Doppler Shift (Hz)')
   legend('Linear Simulations','Nonlinear Simulations')
```

6.1 Function: plotray.m

```
function [TotalDopplerShiftArray]=plotray(
rayfil,x,n,nx,Qsx,Qsz,freq,cs,receiver_depth)

% plot the RAYfil produced by Bellhop
% useage: plotray( rayfil )
% where rayfil is the ray file (without the extension)
% e.g. plotray( 'foofoo' )
%
% MBP July 1999

if ( strcmp( rayfil, 'RAYFIL' ) == 0 )
  rayfil = [ rayfil '.ray' ]; % append extension
end

% plots a BELLHOP ray file

zr = 90.0;        % use this to just plot eigenrays

% open the file

fid = fopen( rayfil, 'r' );
if ( fid == -1 )
   warndlg( 'No ray file exists; you must run BELLHOP first (with ray ouput selected)',
'Warning' );
end

% read header stuff

TITLE  = fgetl(  fid );
FREQ   = fscanf( fid, '%f', 1 );
NBEAMS = fscanf( fid, '%i', 1 );
DEPTHT = fscanf( fid, '%f', 1 );
DEPTHB = fscanf( fid, '%f', 1 );

ii = findstr( TITLE(3:end), "");   % find last quote
TITLE = deblank( TITLE(3:1:ii-1) );  % remove whitespace

% read rays

set( gca, 'YDir', 'Reverse' )   % plot with depth-axis positive down
```

```
xlabel( 'Range (km)' )
xlabel( 'Range (m)' )
ylabel( 'Depth (m)' )
title( TITLE )
hold on

% axis limits
rmin = +1e9;
rmax = -1e9;
zmin = +1e9;
zmax = -1e9;

count=1; % Added by Cihan Bayindir
TotalDopplerShift=0;
TotalDopplerShiftArray(1)=0; % Added by Cihan Bayindir for Doppler Shift
Calculations
 raycount=1;

for ibeam = 1:NBEAMS
  alpha0    = fscanf( fid, '%f', 1 );
  nsteps    = fscanf( fid, '%i', 1 );
  NumTopBnc = fscanf( fid, '%i', 1 );
  NumBotBnc = fscanf( fid, '%i', 1 );
   if isempty( nsteps ); break; end
   ray = fscanf( fid, '%f', [2 nsteps] );
   r = ray( 1, : );
   z = ray( 2, : );

   %r = r / 1000;   % convert to km

   lincol = 'kbgrcmy';
   ii = NumBotBnc;
   ii = mod( ii, 3 ) + 1;

   % plot( r, z, lincol(ii) );
   if NumTopBnc > 1 && NumBotBnc > 1
     % plot( r, z, 'k' )    % hits both boundaries
   elseif NumBotBnc >= 1
     % plot( r, z, 'b'  )    % hits bottom only
   elseif NumTopBnc >= 1
      plot( r, z, 'k-x'  )    % hits surface only
```

```
    cou=1;

    %TotalDopplerShift=0; % Added by Cihan Bayindir for Doppler Shift
Calculations

    if abs(mean(z(length(z)-5:length(z)))-receiver_depth)<1.5; % There is a bug in
Bellhop
                        % If the ray does not hit the receiver
                        % it is not an eigenray
    for q=15:length(z)-25; % Added by Cihan BAYINDIR to find the points of ray
hitting in the surface
                    % in order to examine Doppler Shift

        [a,b]=find(x<r(q));
        indice=max(b);
        % Linearly interpolated elevation
        El=(n(indice+1)-n(indice))*(r(q)-x(indice))/(x(indice+1)-x(indice))+n(indice);
        if (abs(El-z(q))<0.000055);  % If elevation of ray exceeds water surface, that is
% an reflection point
                            % this is the criteria implemented in Bellhop.
                            % See fortran source code
            nxinter=(nx(indice+1)-nx(indice))*(r(q)-x(indice))/(x(indice+1)-
x(indice))+nx(indice);
            Qsxinter=(Qsx(indice+1)-Qsx(indice))*(r(q)-x(indice))/(x(indice+1)-
x(indice))+Qsx(indice);
            Qszinter=(Qsz(indice+1)-Qsz(indice))*(r(q)-x(indice))/(x(indice+1)-
x(indice))+Qsz(indice);
            udotn=(-Qsxinter*nxinter+Qszinter)/sqrt(nxinter^2+1);    % Velocity in the
% normal direction

            [repind1,repind2]=find(r(q)==r); % Bellhop writes the same points more than
once
                                    % Detect if they are same points.
            cou=length(repind2);
            lind=max(repind2);

            if (cou>1); % In some of the runs Bellhop generates two points above
%elevation.
                        % Take the upper one as a reflection point

                % Clay and Medwin, Acoustical Oceanography page 335 for
                % Doppler Shift Calculations
```

```
            % Use dot product for calculation of doppler shift
            cosTetas=((r(lind)-r(lind-cou))*nxinter+(z(lind)-z(lind-
cou)))/(sqrt(nxinter^2+1)*sqrt((r(lind)-r(lind-cou))^2+(z(lind)-z(lind-cou))^2));
            cosTetar=((r(lind)-r(lind+1))*nxinter+(z(lind+1)-
z(lind)))/(sqrt(nxinter^2+1)*sqrt((r(lind)-r(lind+1))^2+(z(lind)-z(lind+1))^2));

            origfreq=freq;
            Onehitfreq=freq*(cs+udotn*cosTetas)/(cs-udotn*cosTetar);
             if (r(q)==r(q-1));   % Do not account for Doppler Shift for few times at the
%same point
              else
                freq=Onehitfreq;
              end
            DopplerShift=origfreq-freq;
            TotalDopplerShift=TotalDopplerShift+DopplerShift;

            plot( r(q), z(q), 'magenta-<'  )
          else
          end
        end
    end
        TotalDopplerShiftArray(raycount)=TotalDopplerShift;
        raycount=raycount+1;
    else
        TotalDopplerShiftArray(raycount)=0; % This ray is not an eigenray. Do not
count as a valid result
        raycount=raycount+1;
    end
    cou

  else
  %  plot( r, z, 'r' )      % !!!! 3 commented out by Entin's suggestion in
                        % order to plot only the rays hitting the
                        % surface
  end
  % update axis limits
  rmin = min( [ r rmin ] );
  rmax = max( [ r rmax ] );
  zmin = min( [ z zmin ] );
  zmax = max( [ z zmax ] );
  if ( zmin == zmax ) % horizontal ray causes axis scaling problem
    zmax = zmin + 1;
```

```
  end
  axis( [ rmin, rmax, zmin, zmax ] )

  if rem( ibeam, fix( NBEAMS / 10 ) ) == 0,    % flush graphics buffer every 10th ray
     drawnow
  end;
  %end
end      % next beam
fclose( fid );
hold off
zoom on
```

REFERENCES

Badiey, M., Lenain, L., Wong, K.C., Heitsenrether, R., Sundberg, A. (2003). "Long-term acoustic monitoring of environmental parameters in estuaries", Proc. Oceans 2003 Marine Technology and Ocean Science Conference, San Diego, California, 1234-1237.

Bryant, P.J. (1983). "Cyclic gravity waves in deep water", J. Austral. Math. Soc. B 25, 2-15.

Canuto, C., Hussaini, M.Y., Quarteroni, A. and Zang, T.A. (2006). *Spectral Methods: Fundamentals in Single Domains.* Springer, New York.

Clay, C.S. and Medwin, H. (1977). *Acoustical Oceanography: Principles and Applications.* John Wiley and Sons, New York.

Dahl, P.H. (1996). "On the spatial coherence and angular spreading of sound forward scattered from the sea surface: measurements and interpretive model", J. Acoust. Soc. Am. 100, 748-758.

Dahl, P.H. (1999). "On bistatic sea surface scattering: field measurements and modeling", J. Acoust. Soc. Am. 105 (4), 2155-2169.

Dahl, P.H. (2001). "High-frequency forward scattering from the sea surface: the characteristic scales of time and angle spreading", IEEE Ocean Eng. 26, 141-151.

Dommermuth, D.G. (1999). "The initialization of nonlinear waves using an adjustment scheme", Wave Motion 32 (2000), 307-317.

Dommermuth, D.G. and Yue, D.K.P. (1987). "A High-order spectral method for the study of nonlinear gravity waves", J. Fluid Mech., Vol. 184, pp.267- 288.

Donelan, M.A., Hamilton, J. and Hui, W.H. (1985). "Directional spectra of wind-generated waves", Phil.Trans. R. Soc. Lond. A 315, 509-562.

Eckart, C. (1953). "The scattering of sound from the sea surface", J. Acoust. Soc. Am. 25, 566-570.

Goda, Y. (1985). *Random Seas and Design of Maritime Structures.* University of Tokyo Press, Tokyo, Japan.

Greenberg, M.D. (1998). *Advanced Engineering Mathematics.* Prentice Hall, Upper Saddle River, New Jersey.

Hasselmann, K., Barnett, T.P., Bouws, E., Carlson, H., Cartwright, D. E., Enke, K., Ewing, J. A., Gienapp, H., Haselmann, D. E., Kruseman, P., Meerburg, A., Müller, P., Olbers, D. J., Richter, K., Sell, W. and Walden, H., (1973). "Measurements of wind-wave growth and swell decay during the joint North Sea wave project (JONSWAP)", Deut. Hydrogr. Inst., Hamburg, 8 (12), 6-95.

Heitsenrether, R.M. (2004). "The influence of fetch limited sea surface roughness on high frequency acoustic propagation in shallow water", Master's Thesis, University of Delaware.

Kuryanov, B.F. (1963). "The scattering of sound at a rough surface with two types of irregularity", Sov. Phys. Acoust. 8, 252-257.

Longuet-Higgins, M.S. and Cokelet, E.D. (1976). “The deformation of steep surface waves on water. I. A numerical method of computation.”, Proc. R. Soc. Lond. A 350, 1-26.

McDaniel, S.T. (1986). “Diffractive corrections to the high-frequency Kirchhoff approximation”, J. Acoust. Soc. Am. 79, 957-952.

McDaniel, S.T. (1987). “Composite-roughness theory applied to scattering from fetch limited seas”, J. Acoust. Soc. Am. 82, 1712-1719.

McDaniel, S.T. and Gorman, A.D. (1983). “Examination of the composite-roughness scattering model”, J. Acoust. Soc. Am. 73, 1476-1486.

Mei, C.C., Stiassnie, M. and Yue, D.K.P. (2005). *Theory and Applications of Ocean Surface Waves, Part 2: Nonlinear Aspects.* World Scientific, New Jersey.

Newland, D.E. (1993). *An Introduction to Random Vibrations, Spectral & Wavelet Analysis.* Dover Publications, Mineola, New York.

Porter, M.B. (1995). *BELLHOP users manual* (available with software downloads).

Porter, M.B. and Bucker, H.P. (1987). “Gaussian beam tracing for computing ocean acoustic fields”, J. Acoust. Soc. Am. 82 (4), pp. 1349-1359.

Rice, S.O. (1951). “Reflection of electromagnetic waves from slightly rough surfaces”, Commun. Pure Appl. Math 4, 351-278.

Rienecker, M.M. and Fenton, J.D. (1981). “A Fourier approximation method for steady water waves”, J. Fluid Mech. 104, 119-137.

Siderius, M. and Porter, B.M. (2008). “Modeling broadband ocean acoustic transmissions with time-varying sea surfaces”, J. Acoust. Soc. Am. 124 (1), pp. 137-150.

Thorsos, E.I. (1990). “Acoustic scattering from a Pierson-Moskowitz sea surface”, J. Acoust. Soc. Am 88 (1), 335-349.

Whitham, G.B. (1974). *Linear and Nonlinear Waves.* Wiley.

Zakharov, V.E. (1968). “Stability of periodic waves of finite amplitude on the surface of a deep fluid”, Soviet Physics JETP, vol. 2, pp. 190-194.