Numerical Simulation of a Weakly Nonlinear Model for Water Waves with Viscosity

BY

MARIA KAKLEAS
B.S. (University of Illinois at Urbana-Champaign) 1999
M.S. (Loyola University of Chicago) 2002

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mathematics
in the Graduate College of the
University of Illinois at Chicago, 2009

Chicago, Illinois

ACKNOWLEDGMENTS

I would like to thank my parents and my advisor for all of their help and support. I could not complete this thesis without them.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

a	typical amplitude
a_p	numerical approximation to Fourier coefficient $\hat{\xi}_p$
$\lvert D\rvert$	Fourier multiplier
d_p	numerical approximation to Fourier coefficient $\hat{\eta}_p$
e_H	relative change in energy of full Euler equations
H	energy of full Euler equations
N	length of interval
N_t	number of grid points in time t
N_x	number of grid points in space x
p	wavenumber
r	order of accuracy
S_η	fluid domain
T	total time of simulation
u	vector consisting of η and ξ
$X(\sigma)\zeta$	1^{st} partial derivative with respect to x at surface
$Y(\sigma)\zeta$	1^{st} partial derivative with respect to y at surface
$Z(\sigma)\zeta$	2^{nd} partial derivative with respect to y at surface

LIST OF ABBREVIATIONS (Continued)

α	$\frac{a}{\lambda}$
$\dot{\alpha}$	amplitude rate of decay
β	$\frac{\nu}{\sqrt{g\lambda^3}}$
Γ	set of wavenumbers
γ	velocity potential in travelling frame
η	shape of free surface
η_{ex}	exact solution of η
$\eta^{N_x,Nt}$	numerical approximation of η
λ	typical wavelength
ν	constant viscosity
ξ	velocity potential at surface
ξ_{ex}	exact solution of ξ
$\xi^{N_x,Nt}$	numerical approximation of ξ
ρ	constant fluid density
Φ_p	fundamental matrix
φ	velocity potential
ψ	shape of free surface in travelling frame
ω_p	$\sqrt{g\|p\|}$

SUMMARY

In this thesis we examined Dias, Dyachenko and Zakharov's equations (DDZ08) in two spatial dimensions and restated them using differential operators that we derived valid at the fluid surface. The model derived is both viscous and weakly nonlinear and we refer to it as Water Waves with Viscosity of order approximation two (WWV2). Next, we found exact solutions for two cases of WWV2: *i)* linear viscous waves and *ii)* inviscid travelling waves. We then sought a numerical solution to WWV2 by applying a Fourier spectral collocation method along with the RK-4 time-stepping scheme. Upon comparing our numerical scheme to both of the exact solutions that we found, we extended the RK4 scheme to the full WWV2 model. We then investigated the following numerically: spatial convergence, temporal convergence, order of accuracy of the solution, decay rate and relative error in energy. Interestingly, we notice that inviscid water wave equations can be numerically estimated fairly accurately by our slightly viscous model without the use of filtering.

CHAPTER 1

INTRODUCTION

1.1 Introduction

The free-surface evolution of surface ocean waves is important in a wide array of engineering applications from wave-structure interactions in deep-sea oil rig design, to the shoaling and breaking of waves in near-shore regions, to the transport and dispersion of pollutants in lakes, seas, and oceans. The Euler equations which model this water wave problem (Lam93) are notoriously difficult for numerical schemes to simulate and the most successful approaches involve sophisticated integral simulations, subtle quadrature rules, and preconditioned iterative solution methods accelerated by, e.g. Fast Multiple methods (see (GGD01) and (FD06)). In this thesis we propose a new model which is not only simple to implement numerically, but also incorporates a physically motivated dissipation mechanism to overcome some of the difficulties mentioned above.

The computation of these surface water waves is challenging for several reasons, but the most important are that the domain of definition of the problem is one of the unknowns, and that there is no natural dissipation mechanism to damp the growth of spurious, high-frequency modes. One method for addressing the first difficulty, and reducing the size of the computational domain by a dimension, is to resort to a surface formulation. One way to accomplish this is to utilize surface integrals (for a sampling of the vast literature on this subject see the survey articles of (Mei78), (Yeu82), (SF82), (TY96), (SZ99) from the *Annual Review of Fluid Mechanics).* Another approach, is to use the Hamiltonian surface formulation of Zakharov (Zak68) which was augmented and simplified by Craig and Sulem (CS93) (see also closely related work of Watson and West (WW75), West *et al* (WBJ$^+$87), and Milder (Mil90)). The contribution of Craig and Sulem to the formulation was the introduction of the Dirichlet-Neumann operator (DNO)- in this context a surface operator which inputs surface Dirichlet data for Laplace's equation inside the fluid domain and produces surface Neumann data- together with a perturbative method for its calculation. In this thesis, we will use this perturbative approach on the surface operators to derive a weakly nonlinear model for the water wave problem.

Recently, Dias, Dyachenko, and Zakharov (DDZ08) have generalized the water wave problem to incorporate weak surface viscosity effects. While their derivation is not completely rigorous (e.g., they consider irrotational flows though viscosity will certainly destroy this property), it is correct in the linear wave limit, and they argue that it is a viable model in the case of small viscosity. The reason for putting forward a viscous water wave model is that it is significantly simpler to numerically simulate and mathematically analyze than the full Navier-Stokes equations posed on a moving domain. In this work we take a slightly different point of view to Dias, Dyachenko and Zakharov's (DDZ's) (DDZ08) model: It provides a physically-motivated mechanism for adding dissipation to the water wave equations. This is important since Craig and Sulem's (CS93) implementation of Zakharov's equations for inviscid flows required significant filtering in order to stabilize their computations. Our new contribution is to argue that it is more natural to consider the DDZ model with very small viscosity for stabilized, inviscid water wave simulations. However, we further simplify the DDZ equations to include only linear and quadratic contributions thereby constituting a weakly nonlinear model for viscous water waves. This approach has the advantage of capturing nearly all of the essential linear nonlinear effects seen in mildly nonlinear water waves, while being considerably simpler to implement that the full DDZ equations.

In the remainder of this introductory chapter, we give a brief overview of the method that Dias, Dyachenko and Zakharov (DDZ08) used to derive equations into which we would like to introduce damping.

1.2 Derivation of Linear Equations

1.2.1 Navier-Stokes

In this thesis, we seek a numerical approximation of weakly damped, free-surface flows in two spacial dimensions. Specifically, we are interested in finding a model of the equations that Dias, Dyachenko and Zakharov (DDZ08) proposed for free-surface flows that are weakly damped. Their goal was to find a new system of dissipative equations stated only in terms of the velocity potential. Using their notation, the Navier-Stokes equations are

$$\partial_t \overrightarrow{v} + (\overrightarrow{v} \cdot \nabla)\overrightarrow{v} = -\frac{1}{\rho}\nabla p + \nu\Delta\overrightarrow{v} + \overrightarrow{g} \tag{1.1a}$$

$$\nabla \cdot \overrightarrow{v} = 0 \tag{1.1b}$$

where $\overrightarrow{v}(x, z, t) := (u, w)$ is the velocity field, $p := p(x, z, t)$ is the pressure, ρ is the fluid density and assumed to be constant, $\overrightarrow{g} := (0, -g)$ is the gravitational vector and ν is the kinematic viscosity which is also assumed constant.

The first boundary condition is the kinematic condition which states that fluid particles at the surface remain at the surface

$$\partial_t \eta + u(x, \eta, t)\partial_x \eta = w(x, \eta, t), \tag{1.2}$$

where $\eta(x, t)$ is the shape of the free surface. The second boundary condition is the dynamic condition. Here, forces on both sides of the fluid's surface $z = \eta(x, t)$ must be equal so that

$$-(p - p_0)\overrightarrow{n} + \underline{\underline{\tau}} \cdot \overrightarrow{n} = 0, \tag{1.3}$$

where the viscous part of the stress tensor is

$$\underline{\underline{\tau}} = \rho\nu \begin{pmatrix} 2\partial_x u & \partial_z u + \partial_x w \\ \partial_z u + \partial_x w & 2\partial_z w \end{pmatrix},$$

and the normal to the free surface is

$$\overrightarrow{n} = \frac{1}{\sqrt{1 + (\partial_x \eta)^2}} \begin{pmatrix} -\partial_x \eta \\ 1 \end{pmatrix}.$$

Since they considered deep water flows, there is no need for a kinematic condition at infinite depth; instead they enforced

$$|\overrightarrow{v}| \to 0 \text{ as } z \to -\infty. \tag{1.4}$$

1.2.2 Potential Flow Theory

In studying water waves in potential flow theory (Ach90) and (Lam93), fluid flow is considered irrotational and viscous terms are ignored. The velocity potential φ is defined as $\overrightarrow{v} = \nabla\varphi$ so that the condition for incompressibility in terms of the velocity potential is

$$\Delta\varphi = 0. \tag{1.5}$$

Expressing the kinematic boundary condition (Equation 1.2) in terms of the velocity potential Dias, Dyachenko and Zakharov found that

$$\partial_t\eta + \partial_x\varphi\partial_x\eta = \partial_z\varphi \qquad \text{at } z = \eta(x,t). \tag{1.6}$$

The dynamic condition (Equation 1.3) on the free surface simplifies to $p(x,\eta,t) = p_0$. They then replaced the velocity field $\overrightarrow{v}$ by the velocity potential $\nabla\varphi$ in the Navier-Stokes conservation of momentum equation, integrated it and used $p(x,\eta,t) = p_0$ at the surface to get

$$\partial_t\varphi + \frac{1}{2}|\nabla\varphi|^2 + gz = 0 \qquad \text{at } z = \eta(x,t). \tag{1.7}$$

Expressing the boundary condition (Equation 1.4) of an infinitely deep layer in terms of the velocity potential, they found

$$|\nabla\varphi| \to 0 \quad \text{as } z \to -\infty. \tag{1.8}$$

1.2.3 New Water Wave Equations

Dias, Dyachenko and Zakharov (DDZ08) wanted to express the Navier-Stokes equations using only the potential part of the velocity. To do this, they first expressed the velocity field $\overrightarrow{v}$ as the sum of a scalar potential and a vector potential using Helmholtz's decomposition: $\overrightarrow{v} = \nabla\varphi + \nabla \times \overrightarrow{A}$, where $\overrightarrow{A}$ is a vector stream function.

The velocity using the Helmholtz decomposition is

$$u(x,z,t) = \partial_x\varphi - \partial_z A_y$$
$$w(x,z,t) = \partial_z\varphi + \partial_x A_y.$$

They assumed a time-factor of $e^{-i\omega t}$ and a space-factor of e^{ikx} for wavelength of $\frac{2\pi}{k}$ so that the solutions for φ and A_y are of the form

$$\varphi(x,z,t) = \varphi_0 e^{i(kx-\omega t)} e^{|k|z}$$

$$A_y(x,z,t) = A_0 e^{i(kx-\omega t)} e^{mz},$$

where

$$m^2 = k^2 - i\frac{\omega}{\nu}.$$

Laplace's equation $\Delta\varphi = 0$ and the boundary condition for infinite depth, $|\nabla\varphi| \to 0$ as $z \to -\infty$, remain unaffected. Their main analysis comes from the kinematic and dynamic boundary conditions. For very small viscosity, they found that the vortical component of velocity is much less than the potential component of velocity. More precisely, they showed that

$$\frac{|A_0|}{|\varphi_0|} \ll 1$$

and that

$$\partial_x A_y|_{z=0} = 2\nu\partial_x^2\eta \qquad \text{at } z = 0.$$

The linearized kinematic condition (Equation 1.6) at $z = 0$ is

$$\partial_t\eta = w(x, 0, t).$$

Separating it into its potential and vortical components, they arrived at

$$\partial_t\eta = \partial_z\varphi + \partial_x A_y$$

at $z = 0$. Additionally, they found that

$$\partial_x A_y|_{z=0} = 2\nu\partial_x^2\eta$$

at the surface so that the kinematic boundary condition can be written as

$$\partial_t \eta = \partial_z \varphi + 2\nu \partial_x^2 \eta \quad \text{at } z = 0 \tag{1.10}$$

and the dynamic condition (Equation 1.7) as

$$\partial_t \varphi + g\eta = -2\nu \partial_z^2 \varphi \quad \text{at } z = 0. \tag{1.11}$$

1.3 Nonlinear Equations with Dissipation

By combining these linear dissipitative equations (Equation 1.10), (Equation 1.11) with the nonlinear inviscid equations (Equation 1.6),(Equation 1.7), Dias, Dyachenko and Zakharov propose that a good model is:

$$\begin{aligned}
&\Delta\varphi = 0 && && \text{(1.12a)}\\
&|\nabla\varphi| \to 0 && z \to -\infty && \text{(1.12b)}\\
&\partial_t \eta = \partial_z \varphi + 2\nu\Delta\eta - \nabla\eta \cdot \nabla\varphi && z = \eta(x, y, t) && \text{(1.12c)}\\
&\partial_t \varphi = -g\eta - 2\nu\partial_z^2\varphi - \frac{1}{2}|\nabla\varphi|^2 && z = \eta(x, y, t). && \text{(1.12d)}
\end{aligned}$$

CHAPTER 2

GOVERNING EQUATIONS

Consider the Euler equations which model the free-surface evolution of a deep, two-dimensional ideal fluid that is irrotational, incompressible and inviscid (Lam93). We define the fluid domain

$$S_\eta := \{(x, y) \in \mathbf{R} \times \mathbf{R} |\ y < \eta(x, t)\},$$

where x and y respectively denote space in the horizontal and vertical directions, t is time and $\eta(x, t)$ measures the deviation of the fluid surface from $y = 0$. The well-known governing equations of an ideal fluid under the influence of gravity (Lam93) are

$$\begin{aligned}
&\Delta\varphi = 0 && \text{in } S_\eta && \text{(2.1a)}\\
&\partial_y\varphi \to 0 && y \to -\infty && \text{(2.1b)}\\
&\partial_t\eta = \partial_y\varphi - (\partial_x\eta)(\partial_x\varphi) && y = \eta(x, t) && \text{(2.1c)}\\
&\partial_t\varphi = -g\eta - \frac{1}{2}(\partial_x\varphi)^2 - \frac{1}{2}(\partial_y\varphi)^2 && y = \eta(x, t), && \text{(2.1d)}
\end{aligned}$$

where g is the gravitational constant, and φ is the velocity potential. This must supplemented with initial conditions:

$$\eta(x, 0) = \eta_0(x), \quad \varphi(x, y, 0) = \varphi_0(x, y), \qquad (2.2)$$

though, from the theory of elliptic partial differential equations (Eva98), the boundary data $\varphi(x, \eta(x, 0), 0)$ suffices. Additionally, lateral boundary conditions must be specified and for this we make the classical choice of periodicity:

$$\eta(x + 2\pi, t) = \eta(x, t), \quad \varphi(x + 2\pi, y, t) = \varphi(x, y, t). \tag{2.3}$$

As we saw in Chapter 1, Dias, Dyachenko and Zakharov (DDZ08) proposed a modification of (Equation 2.1) to take into account some weak effects of surface viscosity:

$$\Delta\varphi = 0 \qquad \text{in } S_\eta \tag{2.4a}$$

$$\partial_y\varphi \to 0 \qquad y \to -\infty \tag{2.4b}$$

$$\partial_t\eta = \partial_y\varphi + 2\nu\partial_x^2\eta - (\partial_x\eta)(\partial_x\varphi) \qquad y = \eta(x, t) \tag{2.4c}$$

$$\partial_t\varphi = -g\eta - 2\nu\partial_y^2\varphi - \frac{1}{2}(\partial_x\varphi)^2 - \frac{1}{2}(\partial_y\varphi)^2 \qquad y = \eta(x, t), \tag{2.4d}$$

together with initial and boundary conditions. We point out that in this modification, only (Equation 2.1c) and (Equation 2.1d) are changed, each with the addition of a linear term scaled by the viscosity ν.

2.1 Surface Variables

Zakharov (Zak68) showed that the Euler equations are a Hamiltonian system in the variables η and $\varphi|_\eta$; if the velocity potential φ is known at the surface, then φ can be reconstructed everywhere. To make this more explicit we use the approach of Craig and Sulem (CS93) and let

$$\xi(x,t) := \varphi(x,\eta(x,t),t), \tag{2.5}$$

so that ξ is the velocity potential at the free surface. From (Equation 2.4), we can see that it is necessary to produce first and second order derivatives in order to find solutions for $\eta(x,t)$ and $\xi(x,t)$ at the surface. With this in mind we define the following maps: Given a solution to the prototype elliptic equation

$$\Delta\phi = 0 \qquad y < \sigma(x) \tag{2.6a}$$

$$\partial_y\phi \to 0 \qquad y \to -\infty \tag{2.6b}$$

$$\phi = \zeta \qquad y = \sigma(x), \tag{2.6c}$$

we can compute

$$X(\sigma)[\zeta] := \partial_x\phi(x,\sigma), \quad Y(\sigma)[\zeta] := \partial_y\phi(x,\sigma), \quad Z(\sigma)[\zeta] := \partial_y^2\phi(x,\sigma), \tag{2.7a}$$

where $W(\sigma)[\zeta]$ denotes a surface operator that is similar to the Dirichlet-Neumann operator (CS93) and (NR01). The brackets indicate that W depends on ζ linearly whereas the parentheses indicate nonlinear dependence on σ. In other words, $X(\sigma)[\zeta]$ is another way of expressing the first partial derivative with respect to x, $Y(\sigma)[\zeta]$ as the first partial derivative with respect to y and $Z(\sigma)[\zeta]$ as the second partial derivative with respect to y.

Restating the kinematic condition and the dynamic condition in terms of these operators we have that (Equation 2.4c) is

$$\partial_t \eta = Y(\eta)[\xi] + 2\nu\partial_x^2\eta - (\partial_x\eta)X(\eta)[\xi]$$

and (Equation 2.4d) becomes

$$\begin{aligned}
\partial_t \xi &= (\partial_y\varphi)(\partial_t\eta) + \partial_t\varphi \\
&= Y(\eta)[\xi](\partial_t\eta) + \partial_t\varphi \\
&= Y(\eta)[\xi]\{Y(\eta)[\xi] + 2\nu\partial_x^2\eta - \partial_x\eta X(\eta)[\xi]\} - g\eta - 2\nu Z(\eta)[\xi] - \frac{1}{2}(X(\eta)[\xi])^2 - \frac{1}{2}(Y(\eta)[\xi])^2 \\
&= -g\eta - 2\nu Z(\eta)[\xi] + \frac{1}{2}(Y(\eta)[\xi])^2 - \frac{1}{2}(X(\eta)[\xi])^2 + 2\nu(\partial_x^2\eta)Y(\eta)[\xi] - (\partial_x\eta)X(\eta)[\xi]Y(\eta)[\xi]
\end{aligned}$$

for $\sigma = \eta$ and $\zeta = \xi$. The surface formulation of the water wave equations with viscosity now reads:

$$\partial_t \eta = Y(\eta)[\xi] + 2\nu\partial_x^2\eta - (\partial_x\eta)X(\eta)[\xi] \tag{2.9a}$$

$$\begin{aligned}
\partial_t \xi &= -g\eta - 2\nu Z(\eta)[\xi] + \frac{1}{2}(Y(\eta)[\xi])^2 - \frac{1}{2}(X(\eta)[\xi])^2 \\
&+ 2\nu(\partial_x^2\eta)Y(\eta)[\xi] - (\partial_x\eta)X(\eta)[\xi]Y(\eta)[\xi].
\end{aligned} \tag{2.9b}$$

2.2 Analytic Dependence of Surface Integral Operators

From Nicholls and Reitich (NR01) it can be shown that the surface operators X, Y and Z depend analytically on the surface deformation $\sigma(x)$ provided that σ has 2 classical derivatives. Setting $\sigma(x) = \epsilon f(x)$, we have

$$X(\sigma) = X(\epsilon f) = \sum_{n=0}^{\infty} X_n(f)\epsilon^n, \tag{2.10a}$$

$$Y(\sigma) = Y(\epsilon f) = \sum_{n=0}^{\infty} Y_n(f)\epsilon^n, \tag{2.10b}$$

$$Z(\sigma) = Z(\epsilon f) = \sum_{n=0}^{\infty} Z_n(f)\epsilon^n. \tag{2.10c}$$

We are interested in investigating a set of weakly nonlinear model equations that are accurate to quadratic order. Our goal is to turn (Equation 2.9) into a weakly nonlinear model by expressing it only in terms of surface operators X_0, X_1, Y_0, Y_1, Z_0 and Z_1.

We seek expressions for the surface operators X_0, X_1, Y_0, Y_1, Z_0 and Z_1 by implementing the Method of Operator Expansions proposed by Milder (Mil91) and Craig and Sulem (CS93). We use

$$\varphi_p(x, y) = e^{ipx+|p|y}, \quad p \in \mathbf{Z} \tag{2.11}$$

as the basis for our solution since we know that it satisfies the prototype elliptic problem (Equation 2.6) and is 2π periodic in x. We also define $\xi_p = \varphi_p$ at the surface.

We derive Y_0 and Y_1 from Y, and the other operators X_0, X_1, Z_0 and Z_1 can be found from X and Z similarly. To find Y_0 and Y_1 we first apply the definition of Y to φ_p at the surface where $y = \sigma$ and obtain

$$Y(\sigma)[e^{ipx+|p|\sigma}] = |p|e^{ipx+|p|\sigma}. \tag{2.12}$$

Implementing a perturbative approach to find our solution, we set $\sigma(x) = \epsilon f(x)$ and use the Taylor series for the exponential to find

$$\left(\sum_{n=0}^{\infty} Y_n(f)\epsilon^n\right)\left[e^{ipx}\sum_{n=0}^{\infty}\frac{(f|p|)^n}{n!}\epsilon^n\right] = |p|e^{ipx}\sum_{n=0}^{\infty}\frac{(f|p|)^n}{n!}\epsilon^n. \tag{2.13}$$

Expanding the sum, we obtain

$$(Y_0 + Y_1(f)\,\epsilon + Y_2(f)\,\epsilon^2 + ...)\left[e^{ipx}\{1 + f|p|\,\epsilon + \frac{(f|p|)^2}{2!}\epsilon^2 + ...\}\right] = |p|e^{ipx}\{1+f|p|\,\epsilon+\frac{(f|p|)^2}{2!}\epsilon^2+...\}. \tag{2.14}$$

Equating at $O(1)$ we find

$$Y_0(f)[e^{ipx}] = |p|e^{ipx}$$

and, using Fourier multiplier notation $D := \frac{1}{i}\partial_x$, this is

$$Y_0(f)[e^{ipx}] = |D|e^{ipx}.$$

Recalling that an L^2 function can be expressed by its Fourier series:

$$\zeta(x) = \sum_{p=-\infty}^{\infty} \hat{\zeta}_p e^{ipx}, \quad \hat{\zeta}_p = \frac{1}{2\pi}\int_0^{2\pi} \zeta(x) e^{-ipx} dx$$

we conclude that

$$Y_0(f)[\zeta(x)] = |D|\zeta(x). \tag{2.15}$$

Comparing $O(\epsilon)$ terms in (Equation 2.14), we find that

$$Y_1(f)[e^{ipx}] + Y_0(f)[f|p|e^{ipx}] = f|p|^2 e^{ipx}.$$

Using Fourier multiplier notation this is the same as

$$Y_1(f)[e^{ipx}] + Y_0(f)[f|D|e^{ipx}] = f|D|^2 e^{ipx}.$$

Again, representing a generic function ζ by its Fourier series we see that

$$Y_1(f)[\zeta(x)] = f|D|^2\zeta(x) - Y_0(f)[f|D|\zeta(x)].$$

Applying the definition of Y_0 (Equation 2.15) we obtain

$$Y_1(f)[\zeta(x)] = f|D|^2\zeta(x) - |D|[f|D|\zeta(x)].$$

In a similar way, we can find expressions for the remaining operators X_0, X_1, Z_0 and Z_1. The $O(1)$ and $O(\epsilon)$ surface operators are summarized as follows:

$$X_0(f)[\zeta] = iD\zeta = \partial_x\zeta \tag{2.16a}$$

$$Y_0(f)[\zeta] = |D|\zeta \tag{2.16b}$$

$$Z_0(f)[\zeta] = |D|^2\zeta \tag{2.16c}$$

$$X_1(f)[\zeta] = f(iD)|D|\zeta - (iD)[f|D|\zeta] \tag{2.16d}$$

$$Y_1(f)[\zeta] = f|D|^2\zeta - |D|[f|D|\zeta] \tag{2.16e}$$

$$Z_1(f)[\zeta] = f|D|^3\zeta - |D|^2[f|D|\zeta]. \tag{2.16f}$$

Noticing that the operators X_0, Y_0 and Z_0 do not depend on f, we suppress the notation for f for these operators from this point forward.

2.3 Weakly Nonlinear Model Equations

In this section we derive our weakly nonlinear model and put it in dimensionless form.

2.3.1 Dimensional Model Equations

We are now ready to form the weakly nonlinear model using the D.D.Z. equations (Equation 2.9) and the surface operators just derived. We describe our model as weakly nonlinear because we consider small amplitude waves (η, ξ small) and thus truncate (Equation 2.9) after quadratic order. Assuming that η and ξ are small and of the same order, we set $\eta = \epsilon\tilde{\eta}$ and $\xi = \epsilon\tilde{\xi}$ in (Equation 2.9) and obtain

$$\partial_t \tilde{\eta}\, \epsilon = Y(\epsilon\tilde{\eta})[\epsilon\tilde{\xi}] + 2\nu\partial_x^2\tilde{\eta}\, \epsilon - \partial_x\tilde{\eta}X(\epsilon\tilde{\eta})[\epsilon\tilde{\xi}]\, \epsilon$$

$$\partial_t \tilde{\xi}\, \epsilon = -g\tilde{\eta}\, \epsilon - 2\nu Z(\epsilon\tilde{\eta})[\epsilon\tilde{\xi}] + \frac{1}{2}(Y(\epsilon\tilde{\eta})[\epsilon\tilde{\xi}])^2 - \frac{1}{2}(X(\epsilon\tilde{\eta})[\epsilon\tilde{\xi}])^2$$
$$+ 2\nu\partial_x^2\tilde{\eta}Y(\epsilon\tilde{\eta})[\epsilon\tilde{\xi}]\, \epsilon - \partial_x\tilde{\eta}X(\epsilon\tilde{\eta})[\epsilon\tilde{\xi}]Y(\epsilon\tilde{\eta})[\epsilon\tilde{\xi}]\, \epsilon.$$

This becomes

$$\partial_t \tilde{\eta}\, \epsilon = \sum_{n=0}^{\infty} Y_n(\tilde{\eta})[\tilde{\xi}]\, \epsilon^{n+1} + 2\nu\partial_x^2\tilde{\eta}\, \epsilon - \partial_x\tilde{\eta}\sum_{n=0}^{\infty} X_n(\tilde{\eta})[\tilde{\xi}]\, \epsilon^{n+2}$$

$$\partial_t \tilde{\xi}\, \epsilon = -g\tilde{\eta}\, \epsilon - 2\nu\sum_{n=0}^{\infty} Z(\tilde{\eta})[\tilde{\xi}]\, \epsilon^{n+1} + \frac{1}{2}\sum_{n=0}^{\infty}(Y(\tilde{\eta})[\tilde{\xi}]\, \epsilon^{n+1})^2 - \frac{1}{2}\sum_{n=0}^{\infty}(X(\tilde{\eta})[\tilde{\xi}]\, \epsilon^{n+1})^2$$
$$+ 2\nu\partial_x^2\tilde{\eta}\sum_{n=0}^{\infty} Y(\tilde{\eta})[\tilde{\xi}]\, \epsilon^{n+2} - \partial_x\tilde{\eta}(\sum_{n=0}^{\infty} X(\tilde{\eta})[\tilde{\xi}]\, \epsilon^{n+1})(\sum_{n=0}^{\infty} Y(\tilde{\eta})[\tilde{\xi}]\, \epsilon^{n+1})\, \epsilon.$$

Expanding the series to include differential operators of $O(1)$ and $O(\epsilon)$ only, we can see that

$$\partial_t \tilde{\eta}\, \epsilon = \{Y_0[\tilde{\xi}]\, \epsilon + Y_1(\tilde{\eta})[\tilde{\xi}]\, \epsilon^2\} + 2\nu\partial_x^2\tilde{\eta}\, \epsilon - \partial_x\tilde{\eta}\{X_0[\tilde{\xi}]\, \epsilon^2 + X_1(\tilde{\eta})[\tilde{\xi}]\, \epsilon^3\} + O(\epsilon^3)$$

$$\partial_t \tilde{\xi}\, \epsilon = -g\tilde{\eta}\, \epsilon - 2\nu\{Z_0[\tilde{\xi}]\, \epsilon + Z_1(\tilde{\eta})[\tilde{\xi}]\, \epsilon^2\} + \frac{1}{2}(Y_0[\tilde{\xi}]\, \epsilon + Y_1(\tilde{\eta})[\tilde{\xi}]\, \epsilon^2)^2 - \frac{1}{2}(X_0[\tilde{\xi}]\, \epsilon + X_1(\tilde{\eta})[\tilde{\xi}]\, \epsilon^2)^2$$
$$+ 2\nu\partial_x^2\tilde{\eta}\{Y_0[\tilde{\xi}]\, \epsilon + Y_1(\tilde{\eta})[\tilde{\xi}]\, \epsilon^2\}\, \epsilon - \partial_x\tilde{\eta}(X_0[\tilde{\xi}]\, \epsilon + X_1(\tilde{\eta})[\tilde{\xi}]\, \epsilon^2)(Y_0[\tilde{\xi}]\, \epsilon + Y_1(\tilde{\eta})[\tilde{\xi}]\, \epsilon^2)\, \epsilon + O(\epsilon^3).$$

Upon simplifying and dividing by ϵ we get that

$$\begin{aligned}
\partial_t\tilde{\eta} &= Y_0[\tilde{\xi}] + Y_1(\tilde{\eta})[\tilde{\xi}]\,\epsilon + 2\nu\partial_x^2\tilde{\eta} - (\partial_x\tilde{\eta})X_0[\tilde{\xi}]\,\epsilon + O(\epsilon^2)\\
\partial_t\tilde{\xi} &= -g\tilde{\eta} - 2\nu Z_0[\tilde{\xi}] - 2\nu Z_1(\tilde{\eta})[\tilde{\xi}]\,\epsilon\\
&+ \frac{1}{2}(Y_0[\tilde{\xi}])^2\,\epsilon - \frac{1}{2}(X_0[\tilde{\xi}])^2\,\epsilon + 2\nu\partial_x^2\tilde{\eta}Y_0[\tilde{\xi}]\,\epsilon + O(\epsilon^2).
\end{aligned}$$

Dropping terms of $O(\epsilon^2)$, setting $\epsilon = 1$ and renaming $\tilde{\eta} = \eta$ and $\tilde{\xi} = \xi$, our model becomes

$$\begin{aligned}
\partial_t\eta &= Y_0[\xi] + Y_1(\eta)[\xi] + 2\nu\partial_x^2\eta - (\partial_x\eta)X_0[\xi]\\
\partial_t\xi &= -g\eta - 2\nu Z_0[\xi] - 2\nu Z_1(\eta)[\xi] + \frac{1}{2}(Y_0[\xi])^2 - \frac{1}{2}(X_0[\xi])^2 + 2\nu\partial_x^2\eta Y_0[\xi].
\end{aligned}$$

Inserting the expressions derived for the $O(1)$ and $O(\epsilon)$ surface operators, we arrive at the model

$$\partial_t\eta = |D|\xi + 2\nu\partial_x^2\eta + \eta|D|^2\xi - |D|[\eta|D|\xi] - (\partial_x\eta)(\partial_x\xi) \tag{2.22a}$$

$$\partial_t\xi = -g\eta - 2\nu|D|^2\xi - 2\nu\eta|D|^3\xi + 2\nu|D|^2[\eta|D|\xi] + \frac{1}{2}(|D|\xi)^2 - \frac{1}{2}(\partial_x\xi)^2 + 2\nu(\partial_x^2\eta)|D|\xi \tag{2.22b}$$

which we refer to as WWV2 for Water Waves with Viscosity of order approximation 2.

2.3.2 Dimensionless Equations

Changing WWV2 into dimensionless form we apply the following scalings

$$x = \lambda x', \quad y = \lambda y', \quad t = \sqrt{\frac{\lambda}{g}}t', \quad \eta = a\eta', \quad \xi = a\sqrt{g\lambda}\xi,' \tag{2.23}$$

where λ is a typical wavelength which we set to 2π and a is a typical amplitude. We also scale the differential operators in the following manner

$$\partial_x = \frac{1}{\lambda}\partial_{x'}, \quad \partial_x^2 = \frac{1}{\lambda^2}\partial_{x'}^2, \quad \partial_y = \frac{1}{\lambda}\partial_{y'}, \quad \partial_t = \sqrt{\frac{g}{\lambda}}\partial_{t'},$$
$$|D| = \frac{1}{\lambda}|D|', \quad |D|^2 = \frac{1}{\lambda^2}(|D|^2)', \quad |D|^3 = \frac{1}{\lambda^3}(|D|^3)'.$$

In dimensionless form we have that

$$\partial_{t'}\eta' = |D|'\xi' + \frac{2\nu}{\sqrt{g\lambda^3}}\partial_{x'}^2\eta' + \frac{a}{\lambda}\left\{\eta'(|D|^2)'\xi' - |D|'[\eta'|D|'\xi'] - \partial_{x'}\eta'\partial_{x'}\xi'\right\}$$

and

$$\partial_{t'}\xi' = -\eta' - \frac{2\nu}{\sqrt{g\lambda^3}}(|D|^2)'\xi' +$$

$$\left\{-\frac{2\nu}{\sqrt{g\lambda^3}}\frac{a}{\lambda}\eta'(|D|^3)'\xi' + \frac{2\nu}{\sqrt{g\lambda^3}}\frac{a}{\lambda}(|D|^2)'[\eta'|D|'\xi'] + \frac{1}{2}\frac{a}{\lambda}(|D|^2)'(\xi')^2 - \frac{1}{2}\frac{a}{\lambda}(\partial_{x'}\xi')^2 + \frac{2\nu}{\sqrt{g\lambda^3}}\frac{a}{\lambda}\partial_{x'}^2\eta'|D|'\xi'\right\}.$$

Letting $\alpha := \frac{a}{\lambda}, \beta := \frac{\nu}{\sqrt{g\lambda^3}}$ and renaming $\eta' = \eta, \xi' = \xi$ the model in dimensionless form can be expressed as

$$\partial_t \eta = |D|\xi + 2\beta\partial_x^2\eta + \alpha\left\{\eta|D|^2\xi - |D|[\eta|D|\xi] - (\partial_x\eta)(\partial_x\xi)\right\} \tag{2.25a}$$

$$\partial_t \xi = -\eta - 2\beta|D|^2\xi + \alpha\left\{-2\beta\eta|D|^3\xi + 2\beta|D|^2[\eta|D|\xi] + \frac{1}{2}(|D|\xi)^2 - \frac{1}{2}(\partial_x\xi)^2 + 2\beta\partial_x^2\eta|D|\xi\right\}. \tag{2.25b}$$

In our numerical experiments we set $\lambda = a$ (i.e., $\alpha = 1$) so that the lengths and amplitude are on the same scale, while varying the nondimensional viscosity β.

2.4 Two Cases with an Exact Solution

In this section we present two cases of exact solutions: *i)* linear water waves with viscosity and *ii)* nonlinear travelling waves without viscosity.

2.4.1 Exact Solution: Linear Viscous Waves

The linearized water wave equations with viscosity from WWV2 (Equation 2.22) are

$$\partial_t \eta = |D|\xi + 2\nu\partial_x^2\eta \tag{2.26a}$$

$$\partial_t \xi = -g\eta - 2\nu|D|^2\xi. \tag{2.26b}$$

Using 2π-periodic boundary conditions, generic initial conditions $\eta_0(x)$ and $\xi_0(x)$, and the Fourier series

$$\eta(x,t) = \sum_{p=-\infty}^{\infty} \hat{\eta}_p e^{ipx}, \quad \xi(x,t) = \sum_{p=-\infty}^{\infty} \hat{\xi}_p e^{ipx},$$

we find that the Fourier transform of (Equation 2.26) is

$$\partial_t \begin{pmatrix} \hat{\eta}_p \\ \hat{\xi}_p \end{pmatrix} = \begin{pmatrix} 2\nu(ip)^2 & |p| \\ -g & -2\nu|p|^2 \end{pmatrix} \begin{pmatrix} \hat{\eta}_p \\ \hat{\xi}_p \end{pmatrix} =: A_p \begin{pmatrix} \hat{\eta}_p \\ \hat{\xi}_p \end{pmatrix}. \tag{2.27}$$

For $p \neq 0$, the corresponding eigenvalues and eigenvectors for (Equation 2.27) are

$$\lambda_+ = -2\nu p^2 + i\sqrt{g|p|}, \quad v_+ = c_+ \begin{pmatrix} |p| \\ i\sqrt{g|p|} \end{pmatrix}$$
$$\lambda_- = -2\nu p^2 - i\sqrt{g|p|}, \quad v_- = c_- \begin{pmatrix} |p| \\ -i\sqrt{g|p|} \end{pmatrix}.$$

The fundamental matrix of the homogeneous system number (Equation 2.27) is

$$\Phi_p(t) := e^{A_p t} = e^{-2\nu p^2 t} \begin{pmatrix} \cos(\omega_p t) & \frac{|p|}{\omega_p} \sin(\omega_p t) \\ -\frac{\omega_p}{|p|} \sin(\omega_p t) & \cos(\omega_p t) \end{pmatrix}, \tag{2.28}$$

where ${\omega_p}^2 = g|p|$. The solution for $p \neq 0$ is

$$\begin{pmatrix} \hat{\eta}_p(t) \\ \hat{\xi}_p(t) \end{pmatrix} = e^{-2\nu p^2 t} \begin{pmatrix} \cos(\omega_p t) & \frac{|p|\sin(\omega_p t)}{\omega_p} \\ -\frac{\omega_p \sin(\omega_p t)}{|p|} & \cos(\omega_p t) \end{pmatrix} \begin{pmatrix} \hat{\eta}_p(0) \\ \hat{\xi}_p(0) \end{pmatrix} \tag{2.29}$$

and for $p = 0$

$$\begin{pmatrix} \hat{\eta}_0(t) \\ \hat{\xi}_0(t) \end{pmatrix} = \begin{pmatrix} \hat{\eta}_0(0) \\ -g\hat{\eta}_0(0) + \hat{\xi}_0(0) \end{pmatrix}. \tag{2.30}$$

Please refer to § A for the general solution to the inhomogeneous linear problem.

2.4.2 Exact Solution: Inviscid Travelling Waves

In this section, we seek an exact solution to WWV2 (Equation 2.22) in the case of travelling waves without viscosity. Beginning with the inviscid WWV2 equations,

$$\partial_t \eta = |D|\xi + \eta|D|^2\xi - |D|[\eta|D|\xi] - (\partial_x\eta)(\partial_x\xi) \tag{2.31a}$$

$$\partial_t \xi = -g\eta + \frac{1}{2}(|D|\xi)^2 - \frac{1}{2}(\partial_x\xi)^2 \tag{2.31b}$$

we convert to a travelling frame moving with speed c by letting

$$\eta(x,t) = \psi(z,t)$$

$$\xi(x,t) = \gamma(z,t),$$

where $z = x + ct$.

The inviscid equations become

$$\partial_t\psi + c\partial_z\psi = |D|\gamma + \psi|D|^2\gamma - |D|[\psi|D|\gamma] - (\partial_z\psi)(\partial_z\gamma)$$
$$\partial_t\gamma + c\partial_z\gamma = -g\psi + \frac{1}{2}(|D|\gamma)^2 - \frac{1}{2}(\partial_z\gamma)^2.$$

Renaming the variables $\eta = \psi, \xi = \gamma, x = z$ and seeking steady state solutions we would like to solve

$$c\partial_x\eta = |D|\xi + \eta|D|^2\xi - |D|[\eta|D|\xi] - (\partial_x\eta)(\partial_x\xi) \tag{2.34a}$$
$$c\partial_x\xi = -g\eta + \frac{1}{2}(|D|\xi)^2 - \frac{1}{2}(\partial_x\xi)^2. \tag{2.34b}$$

We express these governing equations in matrix form

$$Bu = \tilde{R} \tag{2.35}$$

where,

$$B = \begin{pmatrix} c\partial_x & -|D| \\ g & c\partial_x \end{pmatrix}$$

$$u = \begin{pmatrix} \eta \\ \xi \end{pmatrix}$$

$$\tilde{R} = \begin{pmatrix} \eta|D|^2\xi - |D|[\eta|D|\xi] - (\partial_x\eta)(\partial_x\xi) \\ \frac{1}{2}(|D|\xi)^2 - \frac{1}{2}(\partial_x\xi)^2 \end{pmatrix}.$$

Since we are considering the case of steady state, we are interested in investigating a system with low amplitude waves and fluid flow with small velocity potential. We also note that the wave speed varies horizontally but not vertically. Seeking solutions to $Bu = \tilde{R}$, we expand η, ξ and c as a function of wave height δ in the Taylor series

$$u = u(x;\delta) = \sum_{n=1}^{\infty} u_n(x)\delta^n = \sum_{n=1}^{\infty} \begin{pmatrix} \eta_n(x) \\ \xi_n(x) \end{pmatrix} \delta^n, \tag{2.37a}$$

$$c = c(\delta) = c_0 + \sum_{n=1}^{\infty} c_n\delta^n. \tag{2.37b}$$

2.4.2.1 Solution for $n = 1$

Using the expansion for $c(\delta)$ we can express (Equation 2.35) $Bu = \tilde{R}$ as

$$B_0 u = R \tag{2.38}$$

where,

$$B_0 = \begin{pmatrix} c_0\partial_x & -|D| \\ g & c_0\partial_x \end{pmatrix}$$
$$u = \begin{pmatrix} \eta \\ \xi \end{pmatrix}$$
$$R = \begin{pmatrix} \eta|D|^2\xi - |D|[\eta|D|\xi] - (\partial_x\eta)(\partial_x\xi) - (c - c_0)\partial_x\eta \\ \frac{1}{2}(|D|\xi)^2 - \frac{1}{2}(\partial_x\xi)^2 - (c - c_0)\partial_x\xi \end{pmatrix}.$$

Comparing $O(\delta)$ terms we find that

$$B_0 u_1 = 0. \tag{2.40}$$

Taking the Fourier transform we obtain

$$\hat{B}_{0,p}\hat{u}_{1,p} = 0, \tag{2.41}$$

where $\hat{B}_{0,p} = \begin{pmatrix} ic_0p & -|p| \\ g & ic_0p \end{pmatrix}$. Note that the first subscript in $\hat{u}_{1,p}$ refers to the perturbation order of δ while the second subscript refers to the wavenumber p.

Case 1: $n = 1, p = \pm p_0, p \neq 0$

Non-trivial solutions to (Equation 2.41) can be found by taking the determinant of $\hat{B}_{0,p}$ (or the opposite of it):

$$\Lambda(c_0, p) = (c_0p)^2 - g|p|.$$

Now, given a specific wavenumber $p_0 \in \Gamma - \{0\}$, where Γ is a set of wavenumbers, then one can find a c_0 such that $\Lambda(c_0, p_0) = 0$; that is,

$$c_0 = \frac{\sqrt{g|p_0|}}{p_0}. \tag{2.42}$$

Using this c_0, one can find that

$$\hat{u}_{1,p_0} = \kappa \begin{pmatrix} |p_0| \\ ic_0p_0 \end{pmatrix}, \qquad \hat{u}_{1,-p_0} = -\bar{\hat{u}}_{1,p_0}. \tag{2.43}$$

Case 2: $n = 1, p = 0$

We notice that $p = 0$ enforces the condition that the velocity potential at the surface satisfies

$$\int \xi(x)dx = 0 \tag{2.44}$$

which we now assume. Since $\hat{\eta}_{1,0} = 0$, $\hat{\xi}_{1,0}$ is free, we choose $\hat{\xi}_{1,0} = 0$ and

$$\hat{u}_{1,0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{2.45}$$

Case 3: $n = 1, p \neq \pm p_0$

Since $\hat{B}_{0,p}$ is not singular, then $\hat{\eta}_{1,p} = \hat{\xi}_{1,p} = 0$,

$$\hat{u}_{1,p} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{2.46}$$

2.4.2.2 <u>Solution for $n > 1$</u>

In this section we seek a solution for $n > 1$. To do this we insert the expansions for u and c in $B_0 u = R$. We can then expand $R = R(\delta) = \sum_{n=1}^{\infty} R_n \delta^n$ where

$$R_n = \begin{pmatrix} \sum_{l=1}^{n}\{\eta_{n-l}|D|^2\xi_l - |D|[\eta_{n-l}|D|\xi_l] - (\partial_x\eta_{n-l})(\partial_x\xi_l) - (c_{n-l})(\partial_x\eta_l)\} \\ \sum_{l=1}^{n}\{\frac{1}{2}(|D|\xi_{n-l})(|D|\xi_l) - \frac{1}{2}(\partial_x\xi_{n-l})(\partial_x\xi_l) - (c_{n-l})(\partial_x\xi_l)\} \end{pmatrix}$$

so that

$$B_0 u_n = R_n = \begin{pmatrix} R_n^{\eta} \\ R_n^{\xi} \end{pmatrix} - c_{n-1}\begin{pmatrix} \partial_x\eta_1 \\ \partial_x\xi_1 \end{pmatrix}, \tag{2.47}$$

where

$$R^{\eta}_{n} = \sum_{l=1}^{n}\{\eta_{n-l}|D|^{2}\xi_{l} - |D|[\eta_{n-l}|D|\xi_{l}] - (\partial_{x}\eta_{n-l})(\partial_{x}\xi_{l})\} - \sum_{l=2}^{n}(c_{n-l})(\partial_{x}\eta_{l}), \tag{2.48a}$$

$$R^{\xi}_{n} = \sum_{l=1}^{n}\{\frac{1}{2}(|D|\xi_{n-l})(|D|\xi_{l}) - \frac{1}{2}(\partial_{x}\xi_{n-l})(\partial_{x}\xi_{l})\} - \sum_{l=2}^{n}(c_{n-l})(\partial_{x}\xi_{l}). \tag{2.48b}$$

Taking the Fourier transform of (Equation 2.48a), we can see that

$$\hat{B}_{0,p}\hat{u}_{n,p} = \begin{pmatrix} \hat{R}^{\eta}_{n,p} \\ \hat{R}^{\xi}_{n,p} \end{pmatrix} - c_{n-1}\delta_{p,\pm p_0}\begin{pmatrix} (ip)\hat{\eta}_{1,p} \\ (ip)\hat{\xi}_{1,p} \end{pmatrix}, \tag{2.49}$$

where $\delta_{p,q}$ is the Kronecker delta function. Notice that the terms involving c_{n-1} appear only at wavenumber $p = \pm p_0$. There are three cases to consider:

Case 1: $n > 1, p \neq \pm p_0, p \neq 0$

In this case $\hat{B}_{0,p}$ is non-singular, so we can multiply both sides of

$$\hat{B}_{0,p}\hat{u}_{n,p} = \begin{pmatrix} \hat{R}^{\eta}_{n,p} \\ \hat{R}^{\xi}_{n,p} \end{pmatrix}$$

by $\hat{B}^{-1}_{0,p}$ to find

$$\hat{u}_{n,p} = \frac{1}{g|p| - (c_0 p)^2}\begin{pmatrix} ic_0 p\hat{R}^{\eta}_{n,p} + |p|\hat{R}^{\xi}_{n,p} \\ -g\hat{R}^{\eta}_{n,p} + ic_0 p\hat{R}^{\xi}_{n,p} \end{pmatrix}. \tag{2.50}$$

Case 2: $n > 1, p = 0$

When $p = 0$ we find that (Equation 2.49) simplifies to $R^{\eta}_{n,0} = 0$ for all n and

$$g\hat{\eta}_{n,0} = \hat{R^{\xi}}_{n,0}.$$

We can see that $R^{\eta} = 0$ from the following computation:

$$\begin{aligned} R^{\eta} &= \eta|D|^2\xi - |D|[\eta|D|\xi] - (\partial_x\eta)(\partial_x\xi) \\ &= -\eta\partial_x^2\xi - |D|[\eta|D|\xi] - (\partial_x\eta)(\partial_x\xi) \\ &= -\partial_x(\eta\partial_x\xi) - |D|[\eta|D|\xi] \end{aligned}$$

where we have used $|D|^2 = D^2 = (-i\partial_x)^2 = -\partial_x^2$. Since the operators ∂_x and $|D|$ map generic functions to functions with zeroth Fourier coefficient equal to zero, we have $\hat{R}^{\eta}_{n,0}$ as claimed. Regarding $\hat{\xi}_{n,0}$ we simply set it to zero which enforces (Equation 2.44). Thus,

$$\hat{u}_{n,0} = \begin{pmatrix} \frac{\hat{R}^{\xi}_{n,0}}{g} \\ 0 \end{pmatrix}. \tag{2.51}$$

Case 3: $n > 1, p = \pm p_0$

In this case, $\hat{B}_{0,p_0}$ is singular but we can choose c_{n-1} to ensure a solution. We find that

$$c_{n-1} = \frac{g\hat{R}^{\eta}_{n,p_0} - ic_0p_0\hat{R}^{\xi}_{n,p_0}}{c_0p_0^2\hat{\xi}_{1,p_0} + ip_0g\hat{\eta}_{1,p_0}}. \tag{2.52}$$

However, we are left with the issue of uniqueness and follow the approach of Stokes (see(NR05)) for the full water problem, that η_n is L^2-orthogonal to η_1. As η_1 is only supported by wavenumbers $p = \pm p_0$, this is easily enforced by setting $\hat{\eta}_{n,p_0} = 0$ which results in $\hat{\xi}_{n,p_0} = \frac{\hat{R}^{\xi}_{n,p_0} - ip_0c_{n-1}\hat{\xi}_{1,p_0}}{ic_0p_0}$ so that

$$\hat{u}_{n,p_0} = \begin{pmatrix} 0 \\ \frac{\hat{R}^{\xi}_{n,p_0} - ip_0c_{n-1}\hat{\xi}_{1,p_0}}{ic_0p_0} \end{pmatrix}. \tag{2.53}$$

We remark that the case $p = -p_0$ is addressed by setting $\hat{u}_{n,-p_0} = \bar{\hat{u}}_{n,p_0}$.

CHAPTER 3

NUMERICAL METHOD AND RESULTS

In this chapter we discuss the methods that were used to find numerical approximations for the shape of the free-surface and velocity potential. We show that our numerical approximations of η and ξ appear to converge both in space and in time to the exact solutions that we found in Chapter 2 for the cases of *i*) linear viscous waves and *ii*) inviscid travelling waves. We then extend our numerical scheme to the full WWV2 model for small viscosities. We discuss the order of accuracy of our solution and decay rate of our model. Lastly, we compare our model to that of Craig and Sulem (CS93) and calculate the relative error in energy for trials with different values of wave amplitude, low viscosity and number of grid points. All of the code was programmed in MATLAB and is included in Appendices $B - F$ for reference.

3.1 Numerical Method

Our numerical scheme approximates solutions of (Equation 2.22) by using the Discrete Fourier Transform (DFT) accelerated by the Fast Fourier Transform (FFT) algorithm in the spatial variable and the Runge-Kutta scheme of Order Four (RK4) for time-stepping (GO77) (CHQZ88). According to Trefethen (Tre00), the formula for the DFT is

$$\hat{v}_p = h \sum_{j=1}^{N_x} e^{-ipx_j} v_j \tag{3.1}$$

and for the inverse DFT is

$$v_j = \frac{1}{2\pi} \sum_{p=-\frac{N_x}{2}+1}^{\frac{N_x}{2}} e^{ipx_j} \hat{v}_p \tag{3.2}$$

where,

$\hat{v}$ = DFT,

p = wavenumbers $= -\frac{N_x}{2} + 1, ..., \frac{N_x}{2}$,

N_x = number of gridpoints,

$h = \frac{2\pi}{N_x}$ = spacing of grid points in x,

j = index for spatial points $= 1, 2, ...N_x$, and

v = inverse DFT.

Our problem unknowns $\{\eta(x,t), \xi(x,t)\}$ are approximated by $\{\eta^{N_x}(x,t), \xi^{N_x}(x,t)\}$ so that

$$\eta^{N_x}(x,t) := \sum_{p=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} d_p(t) e^{ipx}, \quad \xi^{N_x}(x,t) := \sum_{p=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} a_p(t) e^{ipx}, \tag{3.3}$$

where N_x represents the number of grid points in the spatial variable x and $\{d_p(t), a_p(t)\}$ are approximations to the Fourier coefficients $\{\hat{\eta}_p(t), \hat{\xi}_p(t)\}$. We enforce WWV2 (Equation 2.22) at equally spaced gridpoints $x_j = \frac{2\pi j}{N_x}$ for $j = 0, 1, 2, ..., N_x - 1$.

Derivatives are computed by utilizing (Equation 3.3), the DFT and the FFT algorithm. The Fourier multiplier $|D|$ is computed in a similar manner except that multiplication on the Fourier is replaced by $|p|$. Note that products on the physical side are computed using the inverse DFT and pointwise multiplication. All of this specifies a system of $2 \times N_x$ ordinary differential equations whose approximate solution we denote as $\{d_p^{N_t}(t), a_p^{N_t}(t)\}$ using the Runge-Kutta of Order Four (RK4) scheme for time-stepping.

It is well-known that the RK4 scheme can be stated as follows:

$$w_{i+1} = w_i + \frac{\Delta t}{6}(s_1 + 2s_2 + 2s_3 + s_4), \tag{3.4a}$$

$$s_1 = f(t_i, w_i), \tag{3.4b}$$

$$s_2 = f(t_i + \frac{\Delta t}{2}, w_i + \frac{\Delta t}{2} s_1), \tag{3.4c}$$

$$s_3 = f(t_i + \frac{\Delta t}{2}, w_i + \frac{\Delta t}{2} s_2), \tag{3.4d}$$

$$s_4 = f(t_i + \Delta t, w_i + \Delta t \cdot s_3), \tag{3.4e}$$

where,

w = the function we would like to solve for,

Δt = step size in time,

s_1 = slope at left end of interval,

s_2 = slope at midpoint that uses s_1,

s_3 = another slope at the midpoint that uses s_2,

s_4 = approximate slope at right end of interval that implements s_3, and

$i = 0, 1, 2, 3, \dots$.

3.2 Spatial Convergence

Before presenting our numerical results, we verify our codes by displaying convergence of our numerically generated solutions to the exact solutions discussed earlier: linear viscous waves from § 2.4.1 and nonlinear inviscid travelling water waves from § 2.4.2. Note that in all of the following simulations we have set $g = 1$ and have chosen waves with 2π periodicity so that ν can be viewed as the non-dimensional quantity β from § 2.3.2. In this section we focus on the spatial convergence (with a fixed temporal discretization) while in the next section we examine the temporal discretization (with a fixed spatial resolution). To accomplish this, consider the following quantities:

$$e_{\eta}(N_x, N_t, T) := |\eta^{N_x, N_t}(\cdot, T) - \eta_{ex}(\cdot, T)|_{L^\infty} \tag{3.5a}$$

$$e_{\xi}(N_x, N_t, T) := |\xi^{N_x, N_t}(\cdot, T) - \xi_{ex}(\cdot, T)|_{L^\infty}, \tag{3.5b}$$

where

T = total time of simulation,

N_x = number of grid point in space x,

N_t = number of grid points in time t,

$\{\eta^{N_x, N_t}(x, t), \xi^{N_x, N_t}(x, t)\}$ = numerical approximations of η and ξ,

$\{\eta_{ex}(x, t), \xi_{ex}(x, t)\}$ = exact solutions of η and ξ,

$\{e_\eta(N_x, N_t, T), e_\xi(N_x, N_t, T)\}$ = error between numerical approximation and exact solution of η and ξ estimated by the L^∞-norm.

The initial conditions used for the exact solution of linear water waves (Equation 2.30) are

$$\eta_0(x) = \frac{1}{10}\cos(x), \quad \xi_0(x) = \frac{1}{10}\sin(x). \tag{3.6}$$

From Table I we can see that in the case of linear viscous waves with a fixed time-step of $\triangle t = 2.45 \times 10^{-3}$ and simulation time of $T = 2$, e_η and e_ξ are of the order of 10^{-14} as $N_x \to 64$ for viscosities $\nu = 0, 0.01$ and 0.1. Similarly, in Table II we can see that for $T = 10$, e_η and e_ξ approach the order of 10^{-13} as $N_x \to 64$ for $\nu = 0$ and 0.01 and these errors approach the order of 10^{-14} when $\nu = 0.1$. In both Table I and Table II, the linearized water wave equations with viscosity (Equation 2.26) with initial conditions (Equation 3.6) are compared against the exact solution (Equation 2.29), (Equation 2.30).

We further test the spatial convergence of our time-stepping algorithm by comparing it to the approximations of the travelling wave solutions in the inviscid case and in our simulations we set $M = 20$ and let $\delta = 0.01$. We approximate the solutions by

$$u^M := \sum_{n=1}^{M} \begin{pmatrix} \eta_n(x) \\ \xi_n(x) \end{pmatrix} \delta^n, \quad c^M := c_0 + \sum_{n=1}^{M} c_n \delta^n. \tag{3.7}$$

In Table III we can see that in the case of nonlinear inviscid travelling waves, e_η and e_ξ (Equation 3.5) approach the order of 10^{-15} as $N_x \to 64$ for $T = 2$ and the order of 10^{-14}

ν	N_x	e_η	e_ξ
0	16	1.66601×10^{-11}	1.66601×10^{-11}
	32	1.03899×10^{-12}	1.03898×10^{-12}
	64	5.89095×10^{-14}	5.90188×10^{-14}
0.01	16	1.59005×10^{-11}	1.59005×10^{-11}
	32	9.98009×10^{-13}	9.97957×10^{-13}
	64	5.68504×10^{-14}	5.68027×10^{-14}
0.1	16	1.21801×10^{-11}	1.21800×10^{-11}
	32	7.69267×10^{-13}	7.69260×10^{-13}
	64	4.36456×10^{-14}	4.36456×10^{-14}

TABLE I

SPATIAL CONVERGENCE OF LINEARIZED WWV2 MODEL TO EXACT SOLUTION WITH FIXED TIME-STEP OF 2.45×10^{-3} AND $T = 2$.

as $N_x \rightarrow 64$. Here, the WWV2 model (Equation 2.22) with $\nu = 0$ is compared to the exact solution and initial conditions of travelling wave solutions derived in § 2.4.2 with a fixed time-step of $\triangle t = 2.45 \times 10^{-3}$.

3.2.1 Graphical Comparisons Between the Numerical Approximations and Exact Solutions.

As can be seen in Figure 1 parts a) and b), our numerical approximations of η and ξ are very close to the exact solutions for both the cases of *i*) viscous linear waves and *ii*) inviscid nonlinear waves. Part a) graphically portrays the numerical scheme and the exact solution for the case of linear waves with viscosity of 0.1 whereas the graph for part b) depicts our numerical scheme against the exact solution for inviscid travelling waves. In part c) the plots differ because the travelling wave is not an exact solution. In all three cases $N_x = 64, T = 10, \triangle t = 0.0024544$.

ν	N_x	e_η	e_ξ
0	16	8.20554×10^{-11}	8.20558×10^{-11}
	32	5.22295×10^{-12}	5.22374×10^{-12}
	64	1.86613×10^{-13}	1.86566×10^{-13}
0.01	16	6.78144×10^{-11}	6.78143×10^{-11}
	32	4.26790×10^{-12}	4.26798×10^{-12}
	64	1.53369×10^{-13}	1.53350×10^{-13}
0.1	16	1.24571×10^{-11}	1.24572×10^{-11}
	32	7.79649×10^{-13}	7.79680×10^{-13}
	64	3.51351×10^{-14}	3.51837×10^{-14}

TABLE II

SPATIAL CONVERGENCE OF LINEARIZED WWV2 MODEL TO EXACT SOLUTION WITH FIXED TIME-STEP OF 2.45×10^{-3} AND $T = 10$.

T	N_x	e_η	e_ξ
2	16	1.72501×10^{-12}	1.75794×10^{-12}
	32	1.09069×10^{-13}	1.09654×10^{-13}
	64	6.21920×10^{-15}	6.24397×10^{-15}
10	16	8.22048×10^{-12}	8.26373×10^{-12}
	32	5.24426×10^{-13}	5.26383×10^{-13}
	64	1.87376×10^{-14}	1.88053×10^{-14}

TABLE III

SPATIAL CONVERGENCE OF WWV2 MODEL WITH $\nu = 0$ TO EXACT SOLUTION OF INVISCID TRAVELLING WAVES WITH FIXED TIME-STEP OF 2.45×10^{-3}.

Please refer to § B for the code used to calculate the graph in part a) and § C for parts b) and c).

3.3 Temporal Convergence

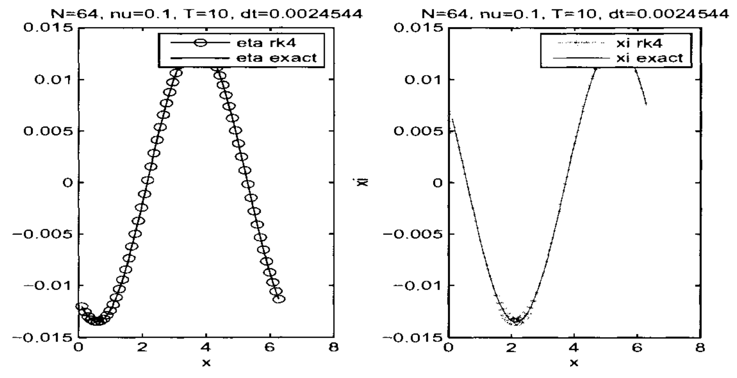
We now investigate the temporal rate of convergence of our scheme by fixing the number of spatial collocation points at $N_x = 64$ and examining the quantities $\{e_\eta, e_\xi\}$. Carrying this out with the exact solution of linear water waves (Equation 2.29) and (Equation 2.30) with initial conditions (Equation 3.6) for final times of $T = 2$ and $T = 10$ we have the data presented in Tables IV and V respectively. The results are presented for $\nu = 0, 0.01, 0.1$.

From Table IV we can see that as $\Delta t \to 2.45 \times 10^{-3}, e_\eta$ and e_ξ are of order 10^{-14} for $\nu = 0, 0.01$ and 0.1 with $T = 2$. In Table V, as $\Delta t \to 2.45 \times 10^{-3}$ with $T = 10, e_\eta$ and e_ξ are of order 10^{-13} for $\nu = 0, 0.01$ and of order 10^{-14} for $\nu = 0.1$. Table VI indicates that in the case of nonlinear inviscid travelling waves, e_η and e_ξ are of order 10^{-15} and 10^{-14} as $\Delta t \to 2.45 \times 10^{-3}$ for simulation times of $T = 2$ and $T = 10$ respectively.

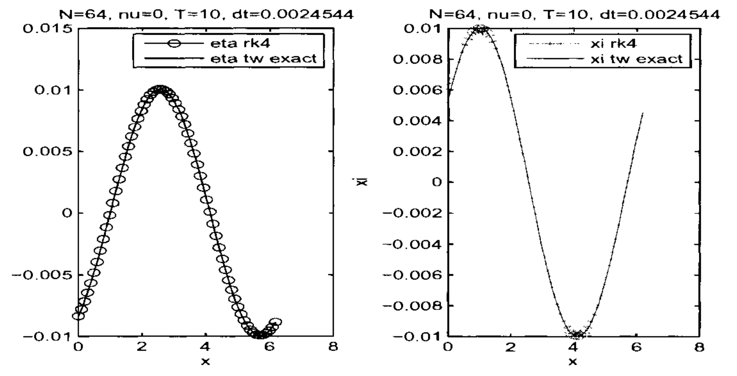
3.3.1 Temporal Order of Accuracy of Solution

We define the order of accuracy of a solution as the number r such that

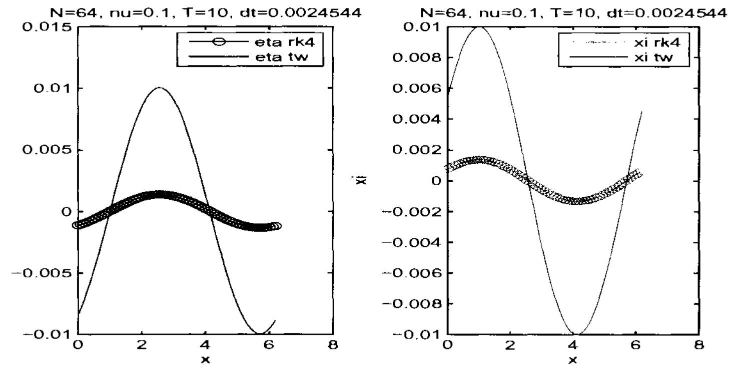
$$error(t_n) = O((\Delta t)^r). \tag{3.8}$$



(a) $N_x = 64, \nu = 0.1, T = 10, dt = 0.0024544$



(b) $N_x = 64, \nu = 0, T = 10, dt = 0.0024544$



(c) $N_x = 64, \nu = 0.1, T = 10, dt = 0.0024544$

Figure 1. Graphical comparison of numerical approximation using RK-4 scheme of a) linearized water waves with viscosity to exact linear solution, b) WWV2 with $\nu = 0$ to exact solution of inviscid travelling wave solution, and c) WWV2 with $\nu = 0.1$ to inviscid travelling wave solution.

ν	Δt	e_η	e_ξ
0	9.82×10^{-3}	1.67×10^{-11}	1.67×10^{-11}
	4.91×10^{-3}	1.04×10^{-12}	1.04×10^{-12}
	2.45×10^{-3}	5.89×10^{-14}	5.90×10^{-14}
0.01	9.82×10^{-3}	1.60×10^{-11}	1.60×10^{-11}
	4.91×10^{-3}	9.99×10^{-13}	9.99×10^{-13}
	2.45×10^{-3}	5.69×10^{-14}	5.68×10^{-14}
0.1	9.82×10^{-3}	1.23×10^{-11}	1.23×10^{-11}
	4.91×10^{-3}	7.69×10^{-13}	7.69×10^{-13}
	2.45×10^{-3}	4.36×10^{-14}	4.36×10^{-14}

TABLE IV

TEMPORAL CONVERGENCE OF LINEARIZED WWV2 MODEL TO EXACT SOLUTION WITH $N_X = 64$ AND $T = 2$.

ν	Δt	e_η	e_ξ
0	9.82×10^{-3}	8.33×10^{-11}	8.33×10^{-11}
	4.91×10^{-3}	5.22×10^{-12}	5.22×10^{-12}
	2.45×10^{-3}	1.87×10^{-13}	1.87×10^{-14}
0.01	9.82×10^{-3}	6.83×10^{-11}	6.83×10^{-11}
	4.91×10^{-3}	4.28×10^{-12}	4.28×10^{-12}
	2.45×10^{-3}	1.53×10^{-13}	1.53×10^{-13}
0.1	9.82×10^{-3}	1.25×10^{-11}	1.25×10^{-11}
	4.91×10^{-3}	7.80×10^{-13}	7.80×10^{-13}
	2.45×10^{-3}	3.51×10^{-14}	3.51×10^{-14}

TABLE V

TEMPORAL CONVERGENCE OF LINEARIZED WWV2 MODEL TO EXACT SOLUTION WITH $N_X = 64$ AND $T = 10$.

T	Δt	e_η	e_ξ
2	9.82×10^{-3}	1.75×10^{-12}	1.76×10^{-12}
	4.91×10^{-3}	1.09×10^{-13}	1.10×10^{-13}
	2.45×10^{-3}	6.22×10^{-15}	6.24×10^{-15}
10	9.82×10^{-3}	8.36×10^{-12}	8.39×10^{-12}
	4.91×10^{-3}	5.24×10^{-13}	5.26×10^{-13}
	2.45×10^{-3}	1.87×10^{-14}	1.88×10^{-14}

TABLE VI

TEMPORAL CONVERGENCE OF WWV2 MODEL WITH $\nu = 0$ TO EXACT SOLUTION OF INVISCID TRAVELLING WAVES WITH $N_X = 64$.

In this case, the error in (Equation 3.8) is the supremum between the numerical approximation and one of the exact solutions that we found for η and ξ. The order of accuracy r was found by setting

$$error \approx C(\Delta t)^r,$$

taking the logs of both sides

$$log(error) \approx log(C) + rlog(\Delta t)$$

and completing a least-squares fit of $log(error)$ vs. $log(\Delta t)$ to find the slope. Well-known theory tells us that r should be 4.

As seen in Table VII, the order of accuracy r_η and r_ξ is about 4.07 for both the linearized and the inviscid WWV2 models when compared to their corresponding exact solutions with

$T = 2$. For a simulation time of $T = 10$ we notice that r_η and r_ξ is about 4.4 for $\nu = 0$ and 0.01 in the linear case and for $\nu = 0$ in WWV2. Interestingly, we notice that the order of accuracy improved to 4.23 from about 4.40 when the viscosity was increased from $\nu = 0.01$ to $\nu = 0.1$ in the linear model with $T = 10$. All of the simulations described in Table VII were taken with $N_x = 64$. In the case of inviscid travelling waves, $M = 20$ and $\delta = 0.01$ in (Equation 3.7).

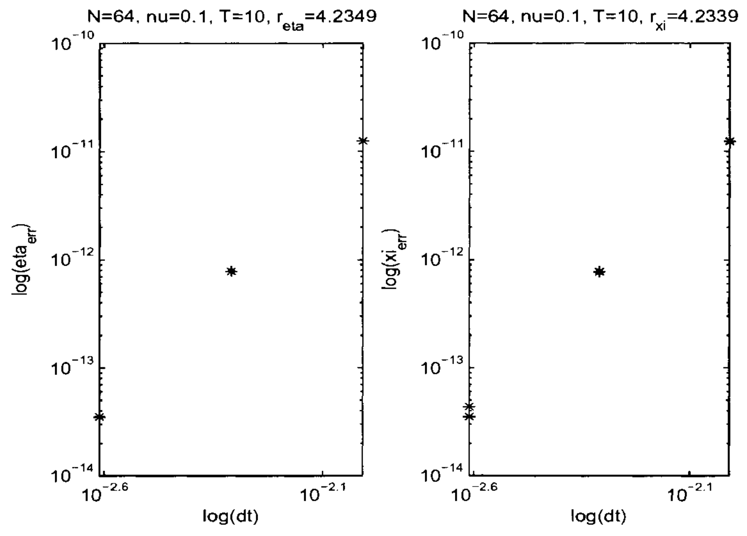
We can also see the order of accuracy r graphically in Figure 2 as it is the slope of $log(error)$ vs. $log(\triangle t)$. In part a) the log($error$) vs. the $log(\triangle t)$ was plotted comparing the linearized water wave model to the exact solution with $\nu = 0.1$. Similarly, in part b) the log($error$) vs. the $log(\triangle t)$ was plotted comparing the water waves model (WWV2) with $\nu = 0$ and the exact solution of inviscid travelling waves. Both of these plots were executed for a simulation time of $T = 10$ and $N_x = 64$. Again, a good order of accuracy for the RK4 scheme is 4 or close to it. In part a) r_η and r_ξ was about 4.23 and in part b) r_η and r_ξ was about 4.4.

3.4 Numerical Results

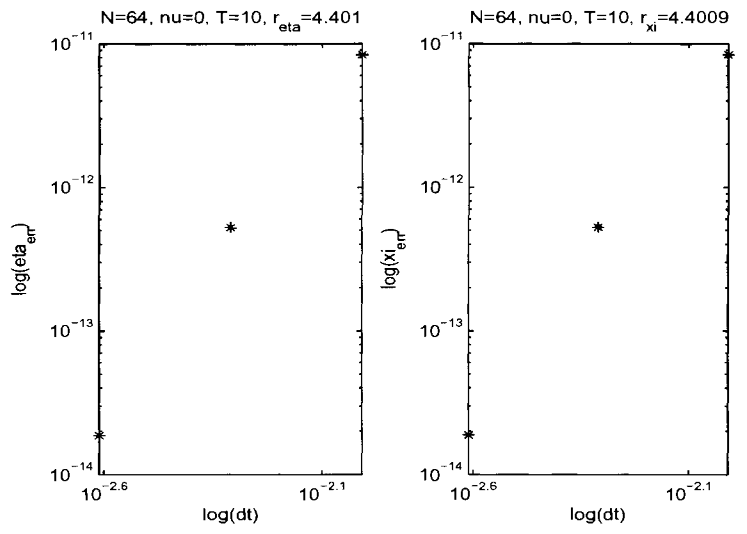
We now present numerical results which illustrate the properties of the solutions to our model equations (Equation 2.22) and the capabilities of our numerical simulation strategy. In particular, we display the decay rates of our solutions to the nonlinear WWV2 equations and then show how our numerical scheme can be used to stably compute inviscid surface water waves.

3.4.1 Travelling Waves

We created some plots to investigate the behavior of our inviscid travelling waves model. Referring to the top two plots in Figure 3, we graphed both η and ξ vs. x and δ. With $M = 20$



(a) Log(error) vs. Log(dt) for Linear Water Waves with Viscosity.



(b) Log(error) vs. Log(dt) for Inviscid Water Waves.

Figure 2. a) Plot of log(error) vs. log($\triangle t$) between linearized water wave equations with viscosity and exact linear solution. b) Plot of log(error) vs. log($\triangle t$) between water wave equations with viscosity (WWV2) with $\nu = 0$ and exact inviscid travelling wave solution.

Linearized WWV2 Compared to Exact Linear Solution or WWV2 Compared to Exact Inviscid Travelling Waves Solution	ν	T	r_η	r_ξ
Linear	0	2	4.07184	4.07051
Linear	0	10	4.40103	4.40121
Linear	0.01	2	4.06946	4.07006
Linear	0.01	10	4.39915	4.39924
Linear	0.1	2	4.07145	4.07145
Linear	0.1	10	4.23492	4.23392
WWV2	0	2	4.06728	4.06862
WWV2	0	10	4.40097	4.40093

TABLE VII

ORDER OF ACCURACY OF SOLUTION, $N_X = 64$.

and perturbation parameter δ in (Equation 3.7) ranging from 0 to 0.05 in increments of 0.001, we can see that the waveform starts out flat for $\delta = 0$ but gains amplitude and becomes more nonlinear as δ increases to a value of 0.05 as expected. The bottom two graphs in Figure 3 show the amplitude of η and ξ against the wave speed c. These graphs indicate that the wave amplitude increases as the wave speed increases. Please refer to § D for the code used to create these graphs.

3.4.2 Decay Rate

We note that from the exact solution, linear solutions (Equation 2.29) should decay like $e^{-2\nu t p^2}$ at wavenumber p. We have numerically simulated such solutions using initial conditions (Equation 3.6) so that $p = 1$ and report experimental decays in Table VIII for both $T = 2$ and $T = 10$. We see that within a very small tolerance (e.g. 10^{-5}) the theoretical decay is realized.

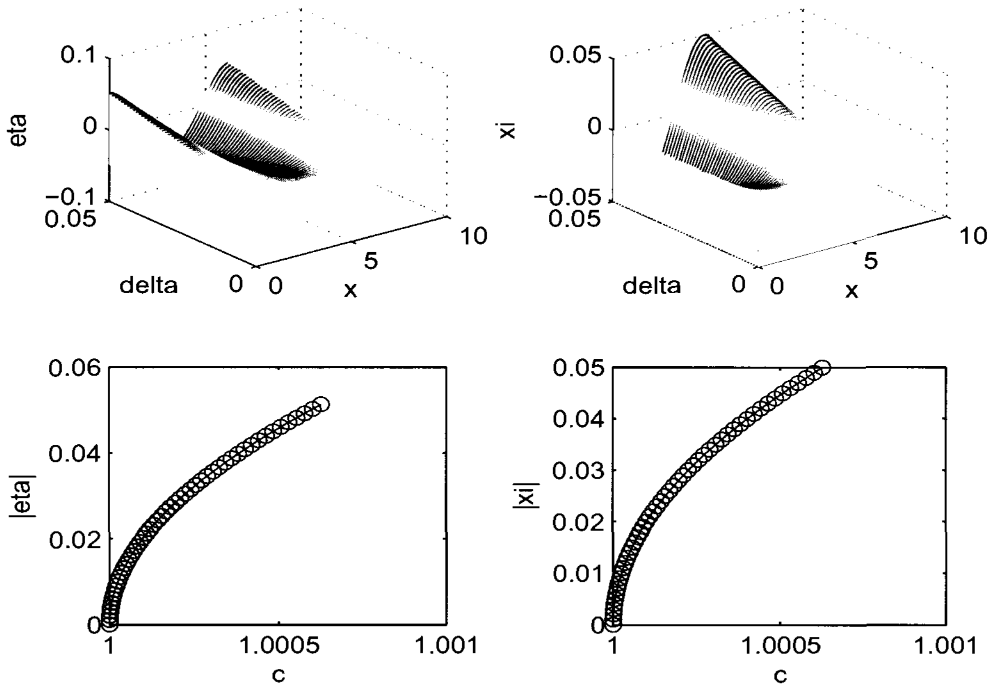


Figure 3. The top two graphs are waterfall plots of travelling waves with different values of δ. The bottom two graphs show the amplitude of η and ξ vs. the wave speed c.

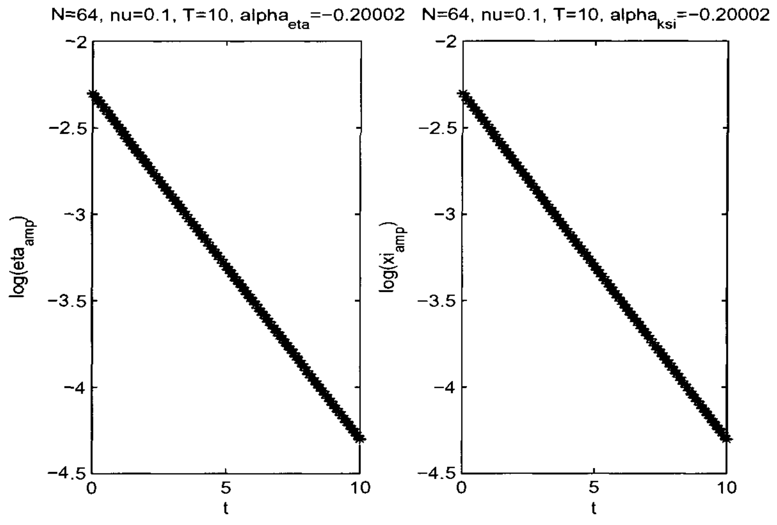
T	ν	$\dot{\alpha}_\eta$	$\dot{\alpha}_\xi$
2	0	-3.33×10^{-4}	-3.33×10^{-4}
	0.01	-2.03×10^{-2}	-2.03×10^{-2}
	0.1	-2.00×10^{-1}	-2.00×10^{-1}
10	0	-1.78×10^{-5}	-1.78×10^{-5}
	0.01	-2.00×10^{-2}	-2.00×10^{-2}
	0.1	-2.00×10^{-1}	-2.00×10^{-1}

TABLE VIII

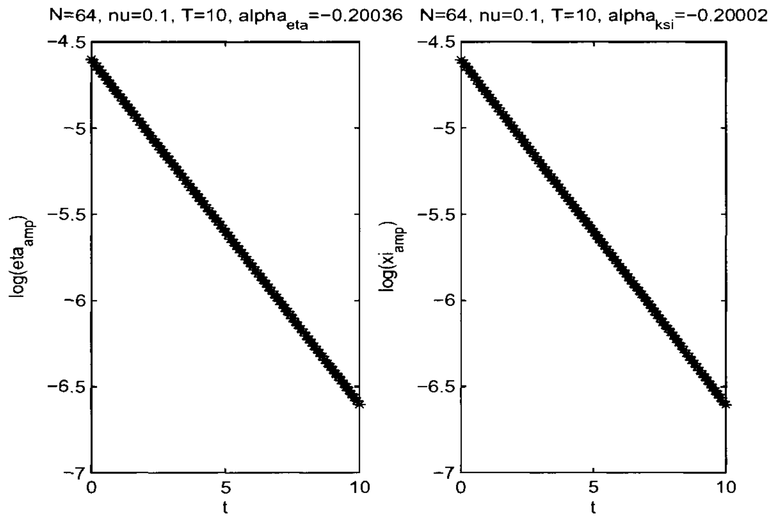
RATE OF DECAY OF THE AMPLITUDE OF SIMULATED SOLUTIONS OF LINEARIZED WWV2. THESE AMPLITUDES ARE MEASURED AT TIMES $T = 2$ AND $T = 10$.

Additionally, we have evolved the travelling waveforms, (Equation 3.7), in the nonlinear WWV2 equations with initial conditions provided by the travelling wave solutions derived in § 2.4.2 and report in Table IX our results for $T = 2$ and $T = 10$. We see how strong the effects of viscosity can be as these nonlinear solutions also decay at roughly the rate expected for linear solutions.

In Figure 4 the rate of decay can be seen graphically as the slope of the log(amplitude) vs. time. These plots were created with $N_x = 64, \nu = 0.1, T = 10, \triangle t = 0.0024544$. The plot in part a) was created using simulated solutions to the linearized wave equations with viscosity (Equation 2.26) whereas the plot in part b) was created using the nonlinear WWV2 equations (Equation 2.22). In part a) the rate of decay is -0.20002 for both η and ξ and in part b) the rate of decay is -0.20036 for η and -0.20002 for ξ. Please refer to § B for the code used to create Figure 4 part a) and § C for part b).



(a) log(amplitude) vs. t, linearized model



(b) log(amplitude) vs. t, nonlinear model

Figure 4. Rate of decay of the amplitude of simulated solutions of a) linearized water wave equations with viscosity and b) nonlinear water wave equations with viscosity. These amplitudes were measured with $N_x = 64, \nu = 0.1, T = 10, \triangle t = 0.0024544$.

T	ν	$\dot{\alpha}_\eta$	$\dot{\alpha}_\xi$
2	0	-3.39×10^{-4}	-1.51×10^{-4}
	0.01	-2.05×10^{-2}	-2.01×10^{-2}
	0.1	-2.01×10^{-1}	-2.00×10^{-1}
10	0	-1.78×10^{-5}	-8.80×10^{-6}
	0.01	-2.01×10^{-2}	-2.00×10^{-2}
	0.1	-2.00×10^{-1}	-2.00×10^{-1}

TABLE IX

RATE OF DECAY OF THE AMPLITUDE OF SIMULATED SOLUTIONS OF WWV2. THESE AMPLITUDES ARE MEASURED AT TIMES $T = 2$ AND $T = 10$.

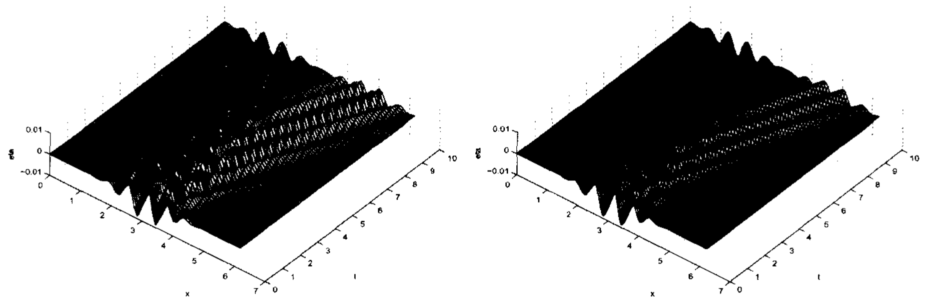
3.4.3 Evolving Inviscid Surface Water Waves Using a Slightly Viscous Model

The results shown in § 3.4.2 also suggest a new strategy for evolving inviscid surface water waves in a stable way. As noted in the publication (CS93), the computation of these waves is quite delicate and filtering is typically required to ensure that the solutions do not blow up. The reason for this is the energy conserving nature of the equations implying no natural energy dissipation mechanism coupled to very strong nonlinearities. Of course our new set of equations circumvent this first challenge with the introduction of viscous dissipation terms. Thus, it seems natural to consider the possibility of approximating inviscid water waves by solving slightly viscous equations.

We have carried out this program for the modulated cosine profile

$$\eta_0(x) = A\cos(10x)e^{-\frac{4}{3}(x-\frac{L}{2})^2}, \quad \xi_0(x) = 0, \tag{3.9}$$

proposed by Craig and Sulem (CS93). To study the evolution of this profile we have chosen the same physical parameter values as those given in (CS93), namely $L = 2\pi, A = 0.01$ and final time $T = 10$. In all of these simulations, $\triangle x = \frac{2\pi}{N_x}$. For this configuration we were able to satisfactorily evolve the initial conditions (Equation 3.9) without the need of any filtering or viscosity $\nu = 0$, for $N_x = 64, 128$ and $\triangle t = \frac{\triangle x}{10}$. Please refer to Figure 5.



(a) $N_x = 64, \nu = 0$ (b) $N_x = 128, \nu = 0$

Figure 5. Evolution of WWV2 with modulated cosine initial condition (Equation 3.9), $A = 0.01, \nu = 0, T = 10, \triangle t = \frac{\triangle x}{10}$. In part a) $N_x = 64$ and in part b) $N_x = 128$.

However, if A is increased to a value of $A = 0.045$ we found that with a moderate number of Fourier collocation points, $N_x = 64$, and a reasonable time-step, $\triangle t = \frac{\triangle x}{10}$, we were unable to resolve a believable solution. To make these ideas more precise we note that from (CS93), the energy of the full Euler equations (Equation 2.1) is

$$H = \frac{1}{2}\int (\xi G(\eta)[\xi] + g\eta^2)dx, \tag{3.10}$$

where $G(\eta)[\xi] = \nabla\varphi \cdot N, \nabla\varphi = (X(\eta)[\xi], Y(\eta)[\xi]), N = (-\partial_x\eta, 1)$ and $G(\eta)[\xi]$ is the DNO (CS93).

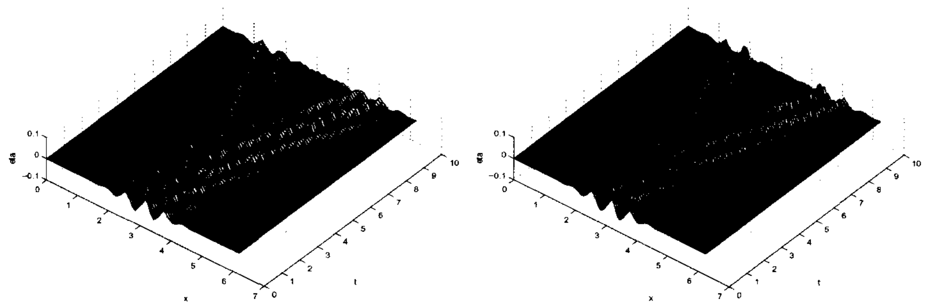
The inviscid version of our model equations is essentially (Equation 2.1) truncated after quadratic contributions. Thus it has energy given by (Equation 3.10) truncated after quadratic order, i.e.,

$$H_2 = \frac{1}{2}\int_0^L \xi\{Y_0[\xi] + Y_1(\eta)[\xi] - (\partial_x\eta)X_0[\xi]\} + g\eta^2 dx. \tag{3.11}$$

To measure the integrity of our solutions we measure the relative change of this energy from the initial to the final time:

$$e_H := \frac{H_2(t=T) - H_2(t=0)}{H_2(t=0)}. \tag{3.12}$$

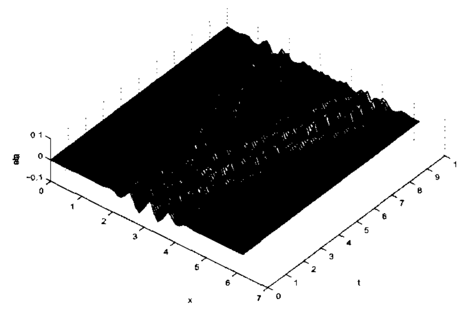
In the case mentioned above ($A = 0.045, N_x = 64$) this relative error is approximately 0.39, while this quantity is unchanged if the time step is reduced by a factor of 10. If the number of collocation points is increased to $N_x = 128$, then for both $\triangle t = \frac{\triangle x}{10}$ and $\triangle t = \frac{\triangle x}{100}$, the solution blows up after $t = 2$. By contrast, if we select $\nu = 2.4 \times 10^{-5}$ with $N_x = 64$ and $\triangle t = \frac{\triangle x}{10}$, then we can produce a solution which not only looks quite reasonable, but also produces a relative energy error of $e_H \approx 7 \times 10^{-3}$, under 1%. If we select $\nu = 1.095 \times 10^{-4}$ with $N_x = 128$ and $\triangle t = \frac{\triangle x}{10}$, then we can compute the solution depicted in Figure 6 b) with $e_H \approx 8 \times 10^{-2}$.



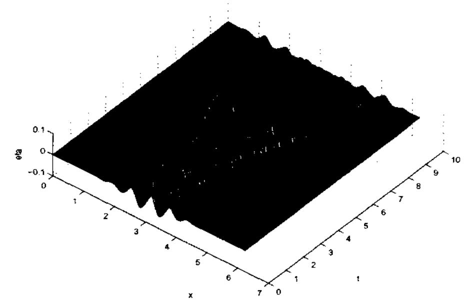
(a) $N_x = 64, \nu = 2.4 \times 10^{-5}$ (b) $N_x = 128, \nu = 1.095 \times 10^{-4}$

Figure 6. Evolution of WWV2 with modulated cosine initial condition (Equation 3.9), $A = 0.045, T = 10, \triangle t = \frac{\triangle x}{10}$. In part a)$N_x = 64, \nu = 2.4 \times 10^{-5}$ and part b) $N_x = 128, \nu = 1.095 \times 10^{-4}$.

In a similar fashion we also investigated the slightly more nonlinear case $A = 0.05$. Here, regardless of our choice of $N_x = 64$ or 128 or our time-step $\triangle t = \frac{\triangle x}{10}$ or $\frac{\triangle x}{100}$, we were unable to obtain a finite solution at $T = 10$ using our code with $\nu = 0$. In this case, filtering of some sort is required. However, if we set $\nu = 5.5 \times 10^{-5}$ then, again, we found a physically reasonable solution with a relative energy error of $e_H \approx 6 \times 10^{-2}$, just over 6%. If we refine to $N_x = 128$ with $\triangle t = \frac{\triangle x}{10}$ then with $\nu = 1.9356 \times 10^{-4}$ we find a solution with $e_H \approx 0.38$. While not really a very satisfactory solution, it at least provides a profile without finite-time blow-up.



(a) $N_x = 64, \nu = 5.5 \times 10^{-5}$



(b) $N_x = 128, \nu = 1.9365 \times 10^{-4}$

Figure 7. Evolution of WWV2 with modulated cosine initial condition (Equation 3.9), $A = 0.05, T = 10, \triangle t = \frac{\triangle x}{10}$. In part a)$N_x = 64, \nu = 5.5 \times 10^{-5}$ and part b) $N_x = 128, \nu = 1.9365 \times 10^{-4}$.

While these wave simulations with small viscous effects were quite successful, the values of ν chosen were quite specific. In general we found that values much larger than the ones chosen resulted in solutions which were overly damped and had energies tending to zero quite rapidly. On the other hand, if ν were chosen much smaller than those reported above, oftentimes solutions would blow up significantly before $T = 10$. However, we do view this as an interesting alternative to other filtering techniques which, themselves, can be quite delicate and subtle.

CHAPTER 4

CONCLUSION

We took Dias, Dyachenko and Zakharov's equations [DDZ08] with small viscosity effects and restated them in terms of the boundary quantities advocated by Zakharov [Zak68] for a Hamiltonian formulation of the water wave problem, namely the surface shape η and surface velocity potential φ.

Upon analyzing the relevant surface integral operators (related to the Dirichlet-Neumann Operator), we used their analyticity properties to derive a new, second order weakly nonlinear model with small viscosity:

$$\partial_t \eta = |D|\xi + 2\nu\partial_x^2\eta + \eta|D|^2\xi - |D|[\eta|D|\xi] - (\partial_x\eta)(\partial_x\xi) \tag{4.1a}$$

$$\partial_t \xi = -g\eta - 2\nu|D|^2\xi - 2\nu\eta|D|^3\xi + 2\nu|D|^2[\eta|D|\xi] + \frac{1}{2}(|D|\xi)^2 - \frac{1}{2}(\partial_x\xi)^2 + 2\nu(\partial_x^2\eta)|D|\xi \tag{4.1b}$$

which we refer to as Water Waves with Viscosity of order approximation two (WWV2).

We found exact solutions for two cases of WWV2: *i*) viscous linear waves and *ii*) inviscid travelling waves. The exact solution for viscous linear waves is

$$\begin{pmatrix} \hat{\eta}_p(t) \\ \hat{\xi}_p(t) \end{pmatrix} = e^{-2\nu p^2 t} \begin{pmatrix} \cos(\omega_p t) & \frac{|p|\sin(\omega_p t)}{\omega_p} \\ -\frac{\omega_p \sin(\omega_p t)}{|p|} & \cos(\omega_p t) \end{pmatrix} \begin{pmatrix} \hat{\eta}_p(0) \\ \hat{\xi}_p(0) \end{pmatrix}, \tag{4.2}$$

where ${\omega_p}^2 = g|p|$. In the case that $p = 0$, we have that

$$\begin{pmatrix} \hat{\eta}_0(t) \\ \hat{\xi}_0(t) \end{pmatrix} = \begin{pmatrix} \hat{\eta}_0(0) \\ -g\hat{\eta}_0(0) + \hat{\xi}_0(0) \end{pmatrix}. \tag{4.3}$$

In the case of inviscid travelling waves we have that the solution for perturbation order $n = 1$ is

$$\hat{u}_{1,p_0} = \kappa \begin{pmatrix} |p_0| \\ ic_0p_0 \end{pmatrix}, \qquad \hat{u}_{1,-p_0} = -\bar{\hat{u}}_{1,p_0}. \tag{4.4}$$

for $p = \pm p_0$ with

$$c_0 = \frac{\sqrt{g|p_0|}}{p_0} \tag{4.5}$$

and we obtain the trivial solution for all other p. For perturbation orders $n > 1$ the solution falls into one of following these 3 cases:

Case 1: $n > 1, p \neq \pm p_0, p \neq 0$

$$\hat{u}_{n,p} = \frac{1}{g|p| - (c_0p)^2} \begin{pmatrix} ic_0p\hat{R}^{\eta}_{n,p} + |p|\hat{R}^{\xi}_{n,p} \\ -g\hat{R}^{\eta}_{n,p} + ic_0p\hat{R}^{\xi}_{n,p} \end{pmatrix}, \tag{4.6}$$

Case 2: $n > 1, p = 0$

$$\hat{u}_{n,0} = \begin{pmatrix} \frac{\hat{R}^{\xi}_{n,0}}{g} \\ 0 \end{pmatrix}, \tag{4.7}$$

Case 3: $n > 1, p = \pm p_0$

$$\hat{u}_{n,p_0} = \begin{pmatrix} 0 \\ \frac{\hat{R}^{\xi}_{n,p_0} - ip_0 c_{n-1}\hat{\xi}_{1,p_0}}{ic_0 p_0} \end{pmatrix}. \tag{4.8}$$

We then outlined a Fourier spectral collocation method for their numerical simulation combined with the RK-4 time-stepping scheme. Furthermore, we investigated our schemes by performing various numerical tests.

In studying spatial convergence we observed errors e_η and e_ξ of the order of $10^{-13}, 10^{-14}$ with $\nu = 0, 0.01, 0.1$ as $N_x \to 64$ in the linear model for $T = 2, 10$. In the case of inviscid nonlinear waves we saw that the error approached the order of 10^{-15} and 10^{-14} for $T = 2$ and $T = 10$ respectively as $N_x \to 64$.

Next, we investigated temporal convergence or our numerical approximation. We saw that as $\triangle t \to 2.45 \times 10^{-3}$ the error was of the order of 10^{-14} for $T = 2$ and $10^{-13}, 10^{-14}$ for $T = 10$ with $\nu = 0, 0.01, 0.1$ in the linearized water wave equations. The error measured between approximations of WWV2 with $\nu = 0$ to the exact solution of inviscid travelling solution was of the order of 10^{-15} and 10^{-14} as $\triangle t \to 2.45 \times 10^{-3}$ for $T = 2$ and $T = 10$ respectively.

Our results also show that the order of accuracy was close to the expected value of 4 and that the rate of decay was near the expected value of 2ν for both η and ξ.

We then examined the numerical scheme with a modulated cosine initial condition while varying the amplitude and small values of viscosity. These results were then compared to that of Craig and Sulem (CS93). To validate our findings we also calculated the relative error in energy. Our results show that our scheme is quite stable and we propose that inviscid equations can sometimes be approximated numerically without filtering using our nonlinear schemes for small viscosities.

APPENDICES

Appendix A

GENERAL SOLUTION TO INHOMOGENEOUS LINEAR PROBLEM

The inhomogeneous linear problem relative to (Equation 2.26) is

$$\partial_t \eta = |D|\xi + 2\nu\partial_x^2\eta + J(x,t) \tag{A.1a}$$

$$\partial_t \xi = -g\eta - 2\nu|D|^2\xi + K(x,t). \tag{A.1b}$$

Upon taking its Fourier transform, we get

$$\partial_t \begin{pmatrix} \hat{\eta}_p \\ \hat{\xi}_p \end{pmatrix} = \begin{pmatrix} 2\nu(ip)^2 & |p| \\ -g & -2\nu|p|^2 \end{pmatrix} \begin{pmatrix} \hat{\eta}_p \\ \hat{\xi}_p \end{pmatrix} + \begin{pmatrix} \hat{J}_p \\ \hat{K}_p \end{pmatrix} =: A_p \begin{pmatrix} \hat{\eta}_p \\ \hat{\xi}_p \end{pmatrix} + \begin{pmatrix} \hat{J}_p \\ \hat{K}_p \end{pmatrix}. \tag{A.2}$$

Using the method of variation of parameters we are able to find the general solution for (Equation A.2). That is, for

$$\partial_t X_p(t) = A_p X_p(t) + F_p(t), \tag{A.3}$$

where

$$X_p(t) = \begin{pmatrix} \hat{\eta}_p(t) \\ \hat{\xi}_p(t) \end{pmatrix}, \quad F_p(t) = \begin{pmatrix} \hat{J}_p(t) \\ \hat{K}_p(t) \end{pmatrix} \tag{A.4}$$

and A_p is part of the fundamental matrix in (Equation 2.28), the general solution is

$$X_p(t) = \Phi_p(t)C + \Phi_p(t)\int \Phi_p{}^{-1}(s)F_p(s)\,ds, \tag{A.5}$$

where,

$\Phi_p =$ the fundamental matrix stated in (Equation 2.11)

$$C = \begin{pmatrix} \hat{\eta}_p(0) \\ \hat{\xi}_p(0) \end{pmatrix}$$

$$\Phi_p{}^{-1} = e^{2\nu p^2 t}\begin{pmatrix} cos(\omega_p t) & -\frac{|p| sin(\omega_p t)}{\omega_p} \\ \frac{\omega_p sin(\omega_p t)}{|p|} & cos(\omega_p t) \end{pmatrix}$$

which leads to the general solution of

$$\begin{pmatrix} \hat{\eta}_p(t) \\ \hat{\xi}_p(t) \end{pmatrix} = e^{-2\nu p^2 t}\begin{pmatrix} \cos(\omega_p t)\hat{\eta}_p(0) + \frac{|p|}{\omega_p}\sin(\omega_p t)\hat{\xi}_p(0) \\ -\frac{\omega_p}{|p|}\sin(\omega_p t)\hat{\eta}_p(0) + \cos(\omega_p t)\hat{\xi}_p(0) \end{pmatrix}$$
$$+ \int_0^t e^{-2\nu p^2(t-s)}\begin{pmatrix} \cos(\omega_p(t-s))\hat{J}_p(s) + \frac{|p|}{\omega_p}\sin(\omega_p(t-s))\hat{K}_p(s) \\ -\frac{\omega_p}{|p|}\sin(\omega_p(t-s))\hat{J}_p(s) + \cos(\omega_p(t-s))\hat{K}_p(s) \end{pmatrix} ds. \tag{A.6}$$

Appendix B

CODE COMPARING RK-4 SCHEME TO EXACT SOLUTION OF VISCOUS LINEAR WATER WAVES

All of the programming was done on in MATLAB R2007a. The computer processor speed was 2.2 GHz, the speed for the memory was 667 MHz, the memory for the system was 3.0 G and the operating system used was Windows Vista. Five programs were used for the linear water waves. The main algorithm is lin.m which is supplemented by functions exactsoln.m, feta.m, fksi.m and plotapprox.m. Function exactsoln.m calculates the exact linear solution found in § 2.4.1. The function plotapprox.m plots the numerical approximation for η and ξ against the exact solution. Functions feta.m and fksi.m calculate the right hand side of (Equation 2.29) and (Equation 2.30).

B.1 Algorithm for Viscous Linear Water Waves: lin.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% lin.m                              %
% RK4 scheme applied to linear model %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; clf; N = 64; g = 1; nu = 0.1; L = 2*pi; h = L/N; x =
h*(1:N); p = (2*pi/L)*[0:N/2-1,-N/2:-1]; pmax = (2*pi/L)*N/2;
% Use different values of dt to produce figure(2) in this program.
% dt = [h/10, h/20, h/40];

% Use a small value for dt to output figure(1) and figure(3) in this program.

dt = h/40; dtmax = 2*nu*pmax^2/(nu^2*pmax^4 + g*pmax)


for m = 1:length(dt)

    t = 0;

    dtm = dt(m);


A = 0.1; D = 0.1; eta0 = A*cos(x); ksi0 = D*sin(x); v_eta = eta0;

v_ksi = ksi0; tmax = 10.0; tplot = 0.1; plotgap =

round(tplot/dtm); dtm = tplot/plotgap; nplots = round(tmax/tplot);

data_eta = [v_eta; zeros(nplots,N)]; data_ksi = [v_ksi;

zeros(nplots,N)]; amp_eta = norm(v_eta,inf); amp_ksi =

norm(v_ksi,inf); tdata = t;


for j = 1:nplots

  for n = 1:plotgap

    t = t + dtm;

    k1_eta = feta(v_eta,v_ksi,g,nu,L,p);

    k1_ksi = fksi(v_eta,v_ksi,g,nu,L,p);

    k2_eta = feta(v_eta + 0.5*dtm*k1_eta,v_ksi + 0.5*dtm*k1_ksi,g,nu,L,p);
  k2_ksi = fksi(v_eta + 0.5*dtm*k1_eta,v_ksi + 0.5*dtm*k1_ksi,g,nu,L,p);

  k3_eta = feta(v_eta + 0.5*dtm*k2_eta,v_ksi + 0.5*dtm*k2_ksi,g,nu,L,p);

  k3_ksi = fksi(v_eta + 0.5*dtm*k2_eta,v_ksi + 0.5*dtm*k2_ksi,g,nu,L,p);

  k4_eta = feta(v_eta + dtm*k3_eta,v_ksi + dtm*k3_ksi,g,nu,L,p);

  k4_ksi = fksi(v_eta + dtm*k3_eta,v_ksi + dtm*k3_ksi,g,nu,L,p);


  slope_eta = dtm*(k1_eta + 2*k2_eta + 2*k3_eta + k4_eta)/6;

  slope_ksi = dtm*(k1_ksi + 2*k2_ksi + 2*k3_ksi + k4_ksi)/6;


  v_eta_new = v_eta + slope_eta;

  v_ksi_new = v_ksi + slope_ksi;


  v_eta = v_eta_new;

  v_ksi = v_ksi_new;

end


data_eta(j,:) = v_eta;

data_ksi(j,:) = v_ksi;

amp_eta = [amp_eta; norm(v_eta,inf)];

amp_ksi = [amp_ksi; norm(v_ksi,inf)];

tdata = [tdata; t];
```

```
  [ex_eta,ex_ksi] = exactsoln(eta0,ksi0,t,g,nu,p);

  exact_eta(j,:) = ex_eta;

  exact_ksi(j,:) = ex_ksi;


end


error_eta(m) = norm(v_eta - ex_eta, inf); error_ksi(m) =

norm(v_ksi - ex_ksi, inf);


% Plotting figure(1): comparison between numerical and exact solution vs. x

figure(1); for j=1:nplots

    plotapprox(x,data_eta(j,:),data_ksi(j,:),exact_eta(j,:),

    exact_ksi(j,:),N,nu,tmax,dt);

    fprintf('[j=%d] t=%g |err_eta|=%g |err_ksi|=%g\n',j,tdata(j),...

      norm(data_eta(j,:)-exact_eta(j,:),inf),...

      norm(data_ksi(j,:)-exact_ksi(j,:),inf),N);

end


end


% Calculating and plotting figure(2): log(error) vs. log(dt)
```

```
cc_eta = polyfit(log(dt), log(error_eta), 1); cc_ksi =

polyfit(log(dt), log(error_ksi), 1);

fprintf('cc_eta(1) = %g cc_eta(2) = %g cc_ksi(1) = %g cc_ksi(2) = %g\n',

cc_eta(1),cc_eta(2),cc_ksi(1),cc_ksi(2));

figure(2); subplot(1,2,1); loglog(dt, error_eta, '*')

xlabel('log(dt)') ylabel('log(eta_e_r_r)')

title(['N=',num2str(N),', nu=',num2str(nu),', T=',num2str(tmax),',

r_e_t_a=',num2str(cc_eta(1))]) subplot(1,2,2); loglog(dt,

error_ksi, '*') xlabel('log(dt)') ylabel('log(xi_e_r_r)')

title(['N=',num2str(N),', nu=',num2str(nu),', T=',num2str(tmax),',

r_x_i=',num2str(cc_ksi(1))]) hold on;


% Calculating decay rate alpha and plotting figure(3): log(amplitude) vs. t

cc2_eta = polyfit(tdata, log(amp_eta), 1); cc2_ksi =

polyfit(tdata, log(amp_ksi), 1);

fprintf('alpha_eta = %g\n', cc2_eta(1));

fprintf('alpha_ksi = %g\n', cc2_ksi(1));

figure(3); subplot(1,2,1) plot(tdata, log(amp_eta),'*')

xlabel('t') ylabel('log(eta_a_m_p)') title(['N=',num2str(N),',

nu=',num2str(nu),', T=',num2str(tmax), ',

alpha_e_t_a=',num2str(cc2_eta(1))]) subplot(1,2,2) plot(tdata,
log(amp_ksi),'*') xlabel('t') ylabel('log(xi_a_m_p)')

title(['N=',num2str(N),', nu=',num2str(nu),', T=',num2str(tmax),',

alpha_k_s_i=',num2str(cc2_ksi(1))])
```

B.2 Function exactsoln.m

```
%%%%%%%%%%%%%%%

% exactsoln.m %

%%%%%%%%%%%%%%%


function [eta,ksi] = exactsoln(eta0,ksi0,t,g,nu,p);


nx = length(p); etahat = 0*eta0; ksihat = 0*ksi0; eta0hat =

fft(eta0); ksi0hat = fft(ksi0);


for j=1:nx

  pp = p(j);

  omega = sqrt(g*abs(pp));


  if(abs(pp)<1e-14)

    a11 = 1.0;

    a12 = 0.0;

    a21 = -g*t;
    a22 = 1.0;

    ee = 1.0;

  else

    a11 = cos(omega*t);

    a12 = (abs(pp)/omega)*sin(omega*t);

    a21 = -(omega/abs(pp))*sin(omega*t);

    a22 = cos(omega*t);

    ee = exp(-2*nu*abs(pp)^2*t);

  end


  etahat(j) = ee*(a11*eta0hat(j)+a12*ksi0hat(j));

  ksihat(j) = ee*(a21*eta0hat(j)+a22*ksi0hat(j));

end


eta = real(ifft(etahat)); ksi = real(ifft(ksihat));
```

B.3 Function feta.m

```
%%%%%%%%%%

% feta.m %

%%%%%%%%%%
```

```
function [fe] = feta(v_eta,v_ksi,g,nu,L,p)


fe1 = real( ifft( abs(p).*fft(v_ksi) ) ); fe2 = 2*nu*real( ifft(

(-p.^2).*fft(v_eta) ) ); fe = fe1 + fe2;
```

B.4 Function fksi.m

```
%%%%%%%%%%

% fksi.m %

%%%%%%%%%%


function [fk] = fksi(v_eta,v_ksi,g,nu,L,p)


fk1 = -g*v_eta; fk2 = -2*nu*real( ifft( (abs(p).^2).*fft(v_ksi) )

); fk = fk1 + fk2;
```

B.5 Function plotapprox.m

```
%%%%%%%%%%%%%%%%

% plotapprox.m %

%%%%%%%%%%%%%%%%


function [] = plotapprox(x,eta,ksi,eta_ex,ksi_ex,N,nu,tmax,dt)
```

```
subplot(1,2,1); plot(x,eta,'b-o',x,eta_ex,'r'); xlabel('x');

ylabel('eta'); title(['N=',num2str(N),', nu=',num2str(nu),',

T=',num2str(tmax),', dt=' num2str(dt)]) legend('eta rk4','eta

exact');


subplot(1,2,2); plot(x,ksi,'g-o',x,ksi_ex,'c'); xlabel('x');

ylabel('xi'); title(['N=',num2str(N),', nu=',num2str(nu),',

T=',num2str(tmax),', dt=' num2str(dt)]) legend('xi rk4','xi

exact'); pause(0.1);
```

Appendix C

CODE COMPARING RK-4 SCHEME FOR WWV2 TO EXACT SOLUTION OF INVISCID TRAVELLING WAVES

Seven programs were used to compare the numerical approximation of WWV2 through the RK-4 scheme to the exact solution of inviscid travelling waves found in § 2.4.2. The main program is nonlin.m that is supported by functions exactsoln_nltw.m, feta_nl.m, fksi_nl.m, newconv.m, plotapprox.m and tw_ww2.m. Program nonlin.m calculates η and ξ using the RK-4 scheme. Functions feta_nl.m and fksi_nl.m calculate the right hand side of $\partial_t\eta$ and $\partial_t\xi$ in WWV2. Function newconv.m is used by feta_nl.m, fksi_nl.m and tw_ww2.m to execute multiplication. Function plotapprox.m plots the numerical estimate of η and ξ against the exact solution of the inviscid travelling waves calculated by tw_ww2.m

C.1 Algorithm for WWV2: nonlin.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% nonlin.m                              %
% RK4 scheme applied to nonlinear model %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; clf;


N = 64; nx = 64; g = 1; nu = 0; L = 2*pi; h = L/N; x = h*[0:N-1];
```

Appendix C (Continued)

```
p = (2*pi/L)*[0:N/2-1,-N/2:-1]; pmax = (2*pi/L)*N/2;


% Use different values of dt to produce figure(2) in this program.

% dt = [h/10, h/20, h/40];

% Use a small value for dt to output figure(1) and figure(3) in this program.

dt =h/40; dtmax = 2*nu*pmax^2/(nu^2*pmax^4 + g*pmax); M = 20; j =

1; delta = 0.01; nt = 1000; [tw_eta, tw_ksi, c] =

tw_ww2(L,g,nx,M,j,delta); num_dt = length(dt);


for m = 1:length(dt)

    t = 0;

    dtm = dt(m);

    eta0_nltw = tw_eta';

    ksi0_nltw = tw_ksi';

A = .1; D = .1; v_eta = tw_eta'; v_ksi = tw_ksi';


tmax = 10.0; tplot = 0.1; plotgap = round(tplot/dtm); dtm =

tplot/plotgap; nplots = round(tmax/tplot); data_eta = [v_eta;

zeros(nplots,N)]; data_ksi = [v_ksi; zeros(nplots,N)]; amp_eta =

norm(v_eta,inf); amp_ksi = norm(v_ksi,inf); tdata = t;
```

```
for j = 1:nplots

  for n = 1:plotgap

    t = t + dtm;

    k1_eta = feta_nl(v_eta,v_ksi,g,nu,L,p);

    k1_ksi = fksi_nl(v_eta,v_ksi,g,nu,L,p);

    k2_eta = feta_nl(v_eta + 0.5*dtm*k1_eta,v_ksi + 0.5*dtm*k1_ksi,g,nu,L,p);

    k2_ksi = fksi_nl(v_eta + 0.5*dtm*k1_eta,v_ksi + 0.5*dtm*k1_ksi,g,nu,L,p);

    k3_eta = feta_nl(v_eta + 0.5*dtm*k2_eta,v_ksi + 0.5*dtm*k2_ksi,g,nu,L,p);

    k3_ksi = fksi_nl(v_eta + 0.5*dtm*k2_eta,v_ksi + 0.5*dtm*k2_ksi,g,nu,L,p);

    k4_eta = feta_nl(v_eta + dtm*k3_eta,v_ksi + dtm*k3_ksi,g,nu,L,p);

    k4_ksi = fksi_nl(v_eta + dtm*k3_eta,v_ksi + dtm*k3_ksi,g,nu,L,p);


    slope_eta = dtm*(k1_eta + 2*k2_eta + 2*k3_eta + k4_eta)/6;

    slope_ksi = dtm*(k1_ksi + 2*k2_ksi + 2*k3_ksi + k4_ksi)/6;


    v_eta_new = v_eta + slope_eta;

    v_ksi_new = v_ksi + slope_ksi;


    v_eta = v_eta_new;

    v_ksi = v_ksi_new;

  end
  data_eta(j,:) = v_eta;

  data_ksi(j,:) = v_ksi;

  amp_eta = [amp_eta; norm(v_eta,inf)];

  amp_ksi = [amp_ksi; norm(v_ksi,inf)];

  tdata = [tdata; t];

  [ex_eta, ex_ksi] = exactsoln_nltw(eta0_nltw,ksi0_nltw,-c,L,x,t);

  exact_eta(j,:) = ex_eta;

  exact_ksi(j,:) = ex_ksi;

end eta_final(m,:) = v_eta; ksi_final(m,:) = v_ksi;


error_eta(m) = norm(v_eta - ex_eta, inf); error_ksi(m) =

norm(v_ksi - ex_ksi, inf);


% Plotting figure(1): comparison between numerical and exact solution vs. x

figure(1); for j=1:nplots

  plotapprox(x,data_eta(j,:),data_ksi(j,:),exact_eta(j,:),exact_ksi(j,:),

  N,nu,tmax,dt);

  fprintf('[j=%d] t=%g |err_eta|=%g |err_ksi|=%g\n',j,tdata(j),...

      norm(data_eta(j,:)-exact_eta(j,:),inf),...

      norm(data_ksi(j,:)-exact_ksi(j,:),inf));

end
```

```
end

% Calculating and plotting figure(2): log(error) vs. log(dt)

cc_eta = polyfit(log(dt), log(error_eta), 1); cc_ksi =

polyfit(log(dt), log(error_ksi), 1);

fprintf('cc_eta(1) = %g cc_eta(2) = %g cc_ksi(1) = %g cc_ksi(2) = %g\n',

cc_eta(1),cc_eta(2),cc_ksi(1),cc_ksi(2));

figure(2); subplot(1,2,1); loglog(dt, error_eta, '*')

xlabel('log(dt)') ylabel('log(eta_e_r_r)')

title(['N=',num2str(N),', nu=',num2str(nu),', T=',num2str(tmax),',

r_e_t_a=',num2str(cc_eta(1))]) subplot(1,2,2); loglog(dt,

error_ksi, '*') xlabel('log(dt)') ylabel('log(xi_e_r_r)')

title(['N=',num2str(N),', nu=',num2str(nu),', T=',num2str(tmax),',

r_x_i=',num2str(cc_ksi(1))]) hold on;

% Calculating decay rate alpha and plotting figure(3): log(amplitude) vs. t

cc2_eta = polyfit(tdata, log(amp_eta), 1); cc2_ksi =

polyfit(tdata, log(amp_ksi), 1);

fprintf('alpha_eta = %g\n', cc2_eta(1));

fprintf('alpha_ksi = %g\n', cc2_ksi(1));
```

```
figure(3); subplot(1,2,1) plot(tdata, log(amp_eta),'*')

xlabel('t') ylabel('log(eta_a_m_p)') title(['N=',num2str(N),',

nu=',num2str(nu),', T=',num2str(tmax), ',

alpha_e_t_a=',num2str(cc2_eta(1))]) subplot(1,2,2) plot(tdata,

log(amp_ksi),'*') xlabel('t') ylabel('log(xi_a_m_p)')

title(['N=',num2str(N),', nu=',num2str(nu),', T=',num2str(tmax),',

alpha_k_s_i=',num2str(cc2_ksi(1))])
```

C.2 Function exactsoln_nltw.m

```
%%%%%%%%%%%%%%%%%%%%

% exactsoln_nltw.m %

%%%%%%%%%%%%%%%%%%%%


function [eta_nltw,ksi_nltw] =

exactsoln_nltw(eta0_nltw,ksi0_nltw,c,L,x,t);


nx = length(eta0_nltw);


% Exact solution is eta(x,t) = eta0_nltw(x-ct), ksi(x,t) = ksi0_nltw(x-ct)


y = x-c*t;
```

```
eta0hat_nltw = fft(eta0_nltw); ksi0hat_nltw = fft(ksi0_nltw);


eta_nltw = 0.0*eta0_nltw; ksi_nltw = 0.0*ksi0_nltw;


for j=1:nx

  eta_nltw(j) = eta0hat_nltw(0+1)/nx;

  for k=1:nx/2-1

    eta_nltw(j) = eta_nltw(j) ...

      + 2*real(eta0hat_nltw(k+1)/nx)*cos(2*pi*k*y(j)/L) ...

      - 2*imag(eta0hat_nltw(k+1)/nx)*sin(2*pi*k*y(j)/L);

  end

end


for j=1:nx

  ksi_nltw(j) = ksi0hat_nltw(0+1)/nx;

  for k=1:nx/2-1

    ksi_nltw(j) = ksi_nltw(j) ...

      + 2*real(ksi0hat_nltw(k+1)/nx)*cos(2*pi*k*y(j)/L) ...

      - 2*imag(ksi0hat_nltw(k+1)/nx)*sin(2*pi*k*y(j)/L);

  end

end
```

C.3 Function feta_nl.m

```
%%%%%%%%%%%%%

% feta_nl.m %

%%%%%%%%%%%%%


function [fe] = feta_nl(v_eta,v_ksi,g,nu,L,p)


pp = p'; absp = abs(pp); absp2 = abs(pp).^2; etahat =

fft(v_eta.'); xihat = fft(v_ksi.');


fe1 = absp.*xihat; fe2 = 2*nu*(((i*pp).^2).*etahat); temp =

absp2.*xihat; fe3 = newconv(etahat,temp); temp = absp.*xihat;

temp2 = newconv(etahat,temp); fe4 = -absp.*temp2; temp =

(i*pp).*xihat; temp2 = (i*pp).*etahat; fe5 = -newconv(temp,temp2);


fe = real( ifft( fe1+fe2+fe3+fe4+fe5 ) ).';
```

C.4 Function fksi_nl.m

```
%%%%%%%%%%%%%

% fksi_nl.m %

%%%%%%%%%%%%%
```

```
function [fk] = fksi_nl(v_eta,v_ksi,g,nu,L,p)


pp = p'; absp = abs(pp); absp2 = abs(pp).^2; absp3 = abs(pp).^3;

etahat = fft(v_eta.'); xihat = fft(v_ksi.');


fx1 = -g*etahat; fx2 = -2*nu*(absp2.*xihat); temp = absp3.*xihat;

fx3 = -2*nu*newconv(etahat,temp); temp = absp.*xihat; temp2 =

newconv(etahat,temp); fx4 = 2*nu*absp2.*temp2; temp = absp.*xihat;

temp2 = absp.*xihat; fx5 = 0.5*newconv(temp,temp2); temp =

(i*pp).*xihat; temp2 = (i*pp).*xihat; fx6 =

-0.5*newconv(temp,temp2); temp = absp.*xihat; temp2 =

((i*pp).^2).*etahat; fx7 = 2*nu*newconv(temp,temp2);


fk = real( ifft( fx1+fx2+fx3+fx4+fx5+fx6+fx7) ).';
```

C.5 Function newconv.m

```
%%%%%%%%%%%%%

% newconv.m %

%%%%%%%%%%%%%


function [c] = newconv(a,b)
```

```
n = length(a); atilde = [a(n/2+1:n);a(1:n/2)]; btilde =
[b(n/2+1:n);b(1:n/2)]; ctilde = conv(atilde,btilde); c = fft(
ifft(a).*ifft(b) );
```

C.6 Function plotapprox.m

```
%%%%%%%%%%%%%%%%
% plotapprox.m %
%%%%%%%%%%%%%%%%

function [] = plotapprox(x,eta,ksi,eta_ex,ksi_ex,N,nu,tmax,dt)

subplot(1,2,1); plot(x,eta,'b-o',x,eta_ex,'r'); xlabel('x');
ylabel('eta'); title(['N=',num2str(N),', nu=',num2str(nu),',
T=',num2str(tmax),', dt=' num2str(dt)]); legend('eta rk4','eta tw
exact');
% For travelling wave approximation use: % legend('eta rk4','eta tw approx');

subplot(1,2,2); plot(x,ksi,'g-o',x,ksi_ex,'c'); xlabel('x');
ylabel('xi'); title(['N=',num2str(N),', nu=',num2str(nu),',
T=',num2str(tmax),', dt=' num2str(dt)]); % For travelling wave approximation use:
```

```
%legend('xi rk4','xi tw approx');


pause(0.1);
```

C.7 Function tw_ww2.m

```
%%%%%%%%%%%

% tw_ww2.m %

%%%%%%%%%%%


function [eta,xi,c] = tw_ww2(L,g,nx,N,j,delta)


% Data structures

%

% Eta -> d_{p,n}

% Xi  -> a_{p,n}


dpn = zeros(nx,N+1); apn = zeros(nx,N+1); cn = zeros(N+1); dx =

L/nx; x = dx*[0:nx-1]'; p = (2*pi/L)*[0:nx/2-1,-nx/2:-1]'; absp =

abs(p); absp2 = absp.^2;


%% n=1

p0 = p(j+1); c0 = sqrt(g*abs(p0))/p0; cn(0+1) = c0; dpn(j+1,1+1) =
abs(p0)*(nx/2.0); dpn(nx-j+1,1+1) = conj(dpn(j+1,1+1));

apn(j+1,1+1) = i*c0*p0*(nx/2.0); apn(nx-j+1,1+1) =

conj(apn(j+1,1+1)); den = g*(i*p0)*dpn(j+1,1+1) +

c0*p0^2*apn(j+1,1+1);


%% n>1


for n=2:N


  % Set up REta


  REta = zeros(nx,1);


  for l=1:n-1

    temp = absp2.*apn(:,l+1);

    REta = REta + newconv(dpn(:,n-l+1),temp);


    temp = absp.*apn(:,l+1);

    temp2 = newconv(dpn(:,n-l+1),temp);

    REta = REta - absp.*temp2;
  temp = (i*p).*apn(:,l+1);

  temp2 = (i*p).*dpn(:,n-l+1);

  REta = REta - newconv(temp,temp2);

end


for l=2:n-1

  REta = REta - cn(n-l+1)*((i*p).*dpn(:,l+1));

end


% Set up RXi


RXi = zeros(nx,1);


for l=1:n-1

  temp = absp.*apn(:,l+1);

  temp2 = absp.*apn(:,n-l+1);

  RXi = RXi + (1.0/2.0)*newconv(temp,temp2);


  temp = (i*p).*apn(:,l+1);

  temp2 = (i*p).*apn(:,n-l+1);

  RXi = RXi - (1.0/2.0)*newconv(temp,temp2);
```

```
end


for l=2:n-1

  RXi = RXi - cn(n-l+1)*((i*p).*apn(:,l+1));

end


% Solve for c_{n-1}


num = g*REta(j+1) - i*c0*p0*RXi(j+1);

cn(n-1+1) = real( num/den );


% Correct REta, RXi


REta = REta - cn(n-1+1)*(i*p).*dpn(:,1+1);

RXi = RXi - cn(n-1+1)*(i*p).*apn(:,1+1);


% Solve for d_{p,n}, a_{p,n}


dpn(0+1,n+1) = RXi(0+1)/g;

apn(0+1,n+1) = 0;
```

```
  for q=1:nx/2-1

    pp = p(q+1);



    if(q~=j)

      deter = g*abs(pp) - (c0*pp)^2;

      dpn(q+1,n+1) = (i*pp*c0*REta(q+1) + abs(pp)*RXi(q+1))/deter;

      apn(q+1,n+1) = (-g*REta(q+1) + i*pp*c0*RXi(q+1))/deter;

    else

      dpn(q+1,n+1) = 0.0;

      apn(q+1,n+1) = RXi(q+1)/(i*c0*p0);

    end



    dpn(nx-q+1,n+1) = conj(dpn(q+1,n+1));

    apn(nx-q+1,n+1) = conj(apn(q+1,n+1));

  end



end



% Find Eta(x), Xi(x) by summing series for d_{p,n}, a_{p,n}



dp = zeros(nx,1); ap = zeros(nx,1); c = c0;
```

Appendix C (Continued)

```
for l=1:N

  dp = dp + dpn(:,l+1)*delta^l;

  ap = ap + apn(:,l+1)*delta^l;

  c = c + cn(l+1)*delta^l;

end


eta = real( ifft( dp ) ); xi = real( ifft( ap ) );
```

Appendix D

CODE FOR TRAVELLING WAVE WATERFALL PLOTS

Three programs were used to create the waterfall plots for travelling waves along with the plots of $|\eta|$ vs. c and $|\xi|$ vs. c. They are plotetaxi.m, tw_ww2.m and newconv.m. Please refer to § C for the codes to functions tw_ww2.m and newconv.m.

D.1 Algorithm for Plotting Travelling Wave Waterfall Plots: plotetaxi.m

```
%%%%%%%%%%%%%%%
% plotetaxi.m %
%%%%%%%%%%%%%%%


% Plotting travelling solutions of WWV2 (with nu=0)
% for different values of delta.


clear all; clf; L = 2.0*pi; g = 1.0; nx = 128; N = 20; j = 1; dx =
L/nx; x = dx*[0:nx-1]'; delta = [0:0.001:0.05]; nd =
length(delta); etaplot = zeros(nd,nx); xiplot = zeros(nd,nx); h =
waitbar(0,'please wait...');


for m=1:nd
```

```
  waitbar(m/nd);

  [eta,xi,c] = tw_ww2(L,g,nx,N,j,delta(m));

  cc(m) = c;

  etamax(m) = norm(eta,inf);

  ximax(m) = norm(xi,inf);

  etaplot(m,:) = eta;

  xiplot(m,:) = xi;

end


close(h);


subplot(2,2,1); waterfall(x,delta,etaplot);

xlabel('x');ylabel('delta');zlabel('eta'); subplot(2,2,2);

waterfall(x,delta,xiplot); xlabel('x'); ylabel('delta');

zlabel('xi'); subplot(2,2,3); plot(cc,etamax,'b-o'); xlabel('c');

ylabel('|eta|'); subplot(2,2,4); plot(cc,ximax,'b-o');

xlabel('c'); ylabel('|xi|');
```

Appendix E

CODE FOR WATERFALL PLOTS WITH MODULATED COSINE INITIAL CONDITION

Four programs were used in creating the waterfall plots. The algorithm waterfall.m requires subroutines feta_nl.m, fksi_nl.m and newconv.m. Functions feta_nl.m and fksi_nl.m calculate the right hand side of $\partial_t \eta$ and $\partial_t \xi$ in WWV2. Function newconv.m is used by feta_nl.m and fksi_nl.m to execute multiplication. Please refer to § C for feta_nl.m, fksi_nl.m and newconv.m.

E.1 Algorithm for Waterfall Plots with Modulated Cosine IC: waterfall.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% waterfall.m                                    %
% Full model with modulated cosine IC simulating %
% Craig and Sulem waterfall plots                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; clf;


N = 64; nx = 64; g = 1; nu = 0; L = 2*pi; h = L/N; x = h*[0:N-1];
p = (2*pi/L)*[0:N/2-1,-N/2:-1]; pmax = (2*pi/L)*N/2; dt = 0.1;
dtmax = 2*nu*pmax^2/(nu^2*pmax^4 + g*pmax); M = 20; j = 1; delta =
0.01; nt = 1000; num_dt = length(dt);
```

```
for m = 1:length(dt)

    t = 0;

    dtm = dt(m);

eta0_nltw = 0.01*exp(-(4/3)*(x - pi).*(x - pi)).*cos(10*x);

ksi0_nltw = 0*x; v_eta = eta0_nltw; v_ksi = ksi0_nltw; tmax = 5;

tplot = 0.1; plotgap = round(tplot/dtm); dtm = tplot/plotgap;

nplots = round(tmax/tplot); data_eta = [v_eta; zeros(nplots,N)];

data_ksi = [v_ksi; zeros(nplots,N)]; amp_eta = norm(v_eta,inf);

amp_ksi = norm(v_ksi,inf); tdata = t;


for j = 1:nplots

  for n = 1:plotgap

    t = t + dtm;

    k1_eta = feta_nl(v_eta,v_ksi,g,nu,L,p);

    k1_ksi = fksi_nl(v_eta,v_ksi,g,nu,L,p);

    k2_eta = feta_nl(v_eta + 0.5*dtm*k1_eta,v_ksi + 0.5*dtm*k1_ksi,g,nu,L,p);

    k2_ksi = fksi_nl(v_eta + 0.5*dtm*k1_eta,v_ksi + 0.5*dtm*k1_ksi,g,nu,L,p);

    k3_eta = feta_nl(v_eta + 0.5*dtm*k2_eta,v_ksi + 0.5*dtm*k2_ksi,g,nu,L,p);

    k3_ksi = fksi_nl(v_eta + 0.5*dtm*k2_eta,v_ksi + 0.5*dtm*k2_ksi,g,nu,L,p);

    k4_eta = feta_nl(v_eta + dtm*k3_eta,v_ksi + dtm*k3_ksi,g,nu,L,p);
    k4_ksi = fksi_nl(v_eta + dtm*k3_eta,v_ksi + dtm*k3_ksi,g,nu,L,p);

    slope_eta = dtm*(k1_eta + 2*k2_eta + 2*k3_eta + k4_eta)/6;

    slope_ksi = dtm*(k1_ksi + 2*k2_ksi + 2*k3_ksi + k4_ksi)/6;

    v_eta_new = v_eta + slope_eta;

    v_ksi_new = v_ksi + slope_ksi;

    v_eta = v_eta_new;

    v_ksi = v_ksi_new;

  end

  data_eta(j,:) = v_eta;

  data_ksi(j,:) = v_ksi;

  amp_eta = [amp_eta; norm(v_eta,inf)];

  amp_ksi = [amp_ksi; norm(v_ksi,inf)];

  tdata = [tdata; t];

end eta_final(m,:) = v_eta; ksi_final(m,:) = v_ksi;


figure(1); timestep = tmax/nplots; time = [0:timestep:tmax];

waterfall(x,time,data_eta); xlabel('x');ylabel('t');zlabel('eta');

figure(2); waterfall(x,time,data_ksi);

xlabel('x');ylabel('t');zlabel('xi');


end
```

Appendix F

CODE FOR CALCULATING RELATIVE ENERGY

The following programs were used to calculate the relative energy in § 3.4.3: rk4_energy.m, energy.m, feta_nl.m, fksi_nl.m and newconv.m.

F.1 Algorithm for Calculating Numerical Approximation and Energy: rk4_energy.m

```
%%%%%%%%%%%%%%%%
% rk4_energy.m %
%%%%%%%%%%%%%%%%


clear all; clf; g = 1; runnum = 1; if(runnum==1)
    % relerr = 0.000205323
    nx = 64;
    A = 0.01;
    nu = 0.0;
elseif(runnum==2)
    % relerr = 1.19193e-009
    nx = 128;
    A = 0.01;
    nu = 0.0;
```

Appendix F (Continued)

```
elseif(runnum==3)

    % relerr = -0.00758532

    nx = 64;

    A = 0.045;

    nu = 2.4e-5;

elseif(runnum==4)

    % relerr = -0.0882224

    nx = 128;

    A = 0.045;

    nu = 1.095e-4;

elseif(runnum==5)

    % relerr = -0.0664213

    nx = 64;

    A = 0.05;

    nu = 5.5e-5;

elseif(runnum==6)

    % relerr = 0.38409

    nx = 128;

    A = 0.05;

    nu = 1.9365e-4;

elseif(runnum==7)
```

```
    % relerr = 0.771103

    nx = 64;

    A = 0.1;

    nu = 1.277e-3;

end L = 2*pi; h = L/nx; x = h*[0:nx-1]; p =

(2*pi/L)*[0:nx/2-1,-nx/2:-1]; dt = h/10; inflate = 0;

fprintf('rk4_nl_waterfall:\n'); fprintf('----------------\n');

fprintf('nx = %d\n',nx);

fprintf('g = %g  nu = %g  L = %g\n',g,nu,L);

fprintf('h = %g  dt = %g\n',h,dt);

fprintf('A = %g\n',A);

fprintf('\n');


for m = 1:length(dt)

  t = 0;

  dtm = dt(m);

  v_eta = A*exp(-(4/3)*(x - L/2.0).*(x - L/2.0)).*cos(10*x);

  v_ksi = 0*x;

  tmax = 10.0;

  tplot = 0.1;

  plotgap = round(tplot/dtm);
```

```
dtm = tplot/plotgap;

nplots = round(tmax/tplot);

data_eta = [v_eta; zeros(nplots,nx)];

data_ksi = [v_ksi; zeros(nplots,nx)];

amp_eta = norm(v_eta,inf);

amp_ksi = norm(v_ksi,inf);

en = energy(v_eta,v_ksi,g,nu,L,p);

tdata = t;


fprintf('t=%g: |eta|=%g |xi|=%g E=%g\n',...

  tdata(1),amp_eta(1),amp_ksi(1),en(1));

for j = 1:nplots

  for n = 1:plotgap

    t = t + dtm;

    k1_eta = feta_nl(v_eta,v_ksi,g,nu,L,p);

    k1_ksi = fksi_nl(v_eta,v_ksi,g,nu,L,p);

    k2_eta = feta_nl(v_eta + 0.5*dtm*k1_eta,v_ksi + 0.5*dtm*k1_ksi,g,nu,L,p);

    k2_ksi = fksi_nl(v_eta + 0.5*dtm*k1_eta,v_ksi + 0.5*dtm*k1_ksi,g,nu,L,p);

    k3_eta = feta_nl(v_eta + 0.5*dtm*k2_eta,v_ksi + 0.5*dtm*k2_ksi,g,nu,L,p);

    k3_ksi = fksi_nl(v_eta + 0.5*dtm*k2_eta,v_ksi + 0.5*dtm*k2_ksi,g,nu,L,p);

    k4_eta = feta_nl(v_eta + dtm*k3_eta,v_ksi + dtm*k3_ksi,g,nu,L,p);
    k4_ksi = fksi_nl(v_eta + dtm*k3_eta,v_ksi + dtm*k3_ksi,g,nu,L,p);


    slope_eta = dtm*(k1_eta + 2*k2_eta + 2*k3_eta + k4_eta)/6;

    slope_ksi = dtm*(k1_ksi + 2*k2_ksi + 2*k3_ksi + k4_ksi)/6;


    v_eta_new = v_eta + slope_eta;

    v_ksi_new = v_ksi + slope_ksi;


   v_eta = v_eta_new;

   v_ksi = v_ksi_new;

 end

  data_eta(j+1,:) = v_eta;

  data_ksi(j+1,:) = v_ksi;

  amp_eta = [amp_eta; norm(v_eta,inf)];

  amp_ksi = [amp_ksi; norm(v_ksi,inf)];

  en = [en; energy(v_eta,v_ksi,g,nu,L,p)];

  tdata = [tdata; t];



  fprintf('t=%g: |eta|=%g |xi|=%g E=%g\n',...

      tdata(j+1),amp_eta(j+1),amp_ksi(j+1),en(j+1));
```

```
end

eta_final(m,:) = v_eta;

ksi_final(m,:) = v_ksi;


timestep = tmax/nplots;

time = [0:timestep:tmax];


figure(1);

subplot(2,2,1);

waterfall(x,time,data_eta);

title('eta');

xlabel('x');ylabel('t');zlabel('eta');

subplot(2,2,2);

waterfall(x,time,data_ksi);

title('xi');

xlabel('x');ylabel('t');zlabel('xi');

subplot(2,2,3);

semilogy(tdata,amp_eta,'b-o',tdata,amp_ksi,'r-*');

xlabel('t'); ylabel('amplitude');

legend('|eta|','|xi|');

subplot(2,2,4);
```

```
  semilogy(tdata,en,'g-o');

  xlabel('t'); ylabel('energy');

  fprintf('Relative energy change = %g\n',(en(1)-en(length(en)))/en(1));

  figure(2);

  hh = surf(x,time,data_eta);

  view(39,76);

  xlabel('x'); ylabel('t'); zlabel('eta');

  saveas(hh,'etasurf.fig');

  saveas(hh,'etasurf.eps');

end
```

F.1.1 Function energy.m

```
%%%%%%%%%%%

% energy.m %

%%%%%%%%%%%


function [en] = energy(v_eta,v_ksi,g,nu,L,p)


pp = p'; absp = abs(pp); absp2 = abs(pp).^2; etahat =

fft(v_eta.'); xihat = fft(v_ksi.'); etaxhat = (i*pp).*etahat;


% Kinetic energy
```

```
X0hat = (i*pp).*xihat; Y0hat = absp.*xihat; temp1 = absp2.*xihat;

Y1hat = newconv(etahat,temp1); temp1 = absp.*xihat; temp2 =

newconv(etahat,temp1); temp3 = absp.*temp2; Y1hat = Y1hat - temp3;

Xhat = X0hat; Yhat = Y0hat + Y1hat; Ghat = Yhat -

newconv(etaxhat,Xhat); Khat = 0.5*newconv(xihat,Ghat);


% Potential energy

Vhat = 0.5*g*newconv(etahat,etahat); EHat = Khat + Vhat; nx =

length(etahat); en = real(EHat(0+1))/nx;
```

CITED LITERATURE

D. J. Acheson. *Elementary fluid dynamics.* The Clarendon Press, Oxford University Press, New York, 1990.

Claudio Canuto, M. Yousuff Hussaini, Alfio Quarteroni, and Thomas A. Zang. *Spectral methods in fluid dynamics.* Springer-Verlag, New York, 1988.

Walter Craig and Catherine Sulem. Numerical simulation of gravity waves. *Journal of Computational Physics*, 108:73–83, 1993.

F. Dias, A.I. Dyachenko, and V.E. Zakharov. Theory of weakly damped free-surface flows: A new formulation based on potential flow solutions. *Phys. Lett. A*, 372:1297–1302, 2008.

Lawrence C. Evans. *Partial differential equations.* American Mathematical Society, Providence, RI, 1998.

Christophe Fochesato and Frédéric Dias. A fast method for nonlinear three-dimensional free-surface waves. *Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, 462(2073):2715–2735, 2006.

S.T. Grilli, P. Guyenne, and F. Dias. A fully nonlinear model for three-dimensional overturning waves over an arbitrary bottom. *Int. J. Numer. Meth. Fluids*, 35:829–867, 2001.

David Gottlieb and Steven A. Orszag. *Numerical analysis of spectral methods: theory and applications.* Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1977. CBMS-NSF Regional Conference Series in Applied Mathematics, No. 26.

Horace Lamb. *Hydrodynamics.* Cambridge University Press, Cambridge, sixth edition, 1993.

C.C̃. Mei. Numerical methods in water-wave diffraction and radiation. *Annual Review of Fluid Mechanics*, 10:393–416, 1978.

D. Michael Milder. The effects of truncation on surface-wave Hamiltonians. *J. Fluid Mech.*, 217:249–262, 1990.

D. Michael Milder. An improved formalism for rough-surface scattering of acoustic and electromagnetic waves. In *Proceedings of SPIE - The International Society for Optical Engineering (San Diego, 1991)*, volume 1558, pages 213–221. Int. Soc. for Optical Engineering, Bellingham, WA, 1991.

David P. Nicholls and Fernando Reitich. A new approach to analyticity of Dirichlet-Neumann operators. *Proc. Roy. Soc. Edinburgh Sect. A*, 131(6):1411–1433, 2001.

David P. Nicholls and Fernando Reitich. On analyticity of traveling water waves. *Proc. Roy. Soc. Lond., A*, 461(2057):1283–1309, 2005.

L. W. Schwartz and J. D. Fenton. Strongly nonlinear waves. In *Annual review of fluid mechanics, Vol. 14*, pages 39–60. Annual Reviews, Palo Alto, Calif., 1982.

Ruben Scardovelli and Stéphane Zaleski. Direct numerical simulation of free-surface and interfacial flow. *Annual Review of Fluid Mechanics*, 31:567–603, 1999.

Lloyd N. Trefethen. *Spectral Methods in MATLAB*. SIAM, Philadelphia, 2000.

Wu-Ting Tsai and Dick K. P. Yue. Computation of nonlinear free-surface flows. In *Annual review of fluid mechanics, Vol. 28*, pages 249–278. Annual Reviews, Palo Alto, CA, 1996.

B. J. West, K. A. Brueckner, R. S. Janda, D. M. Milder, and R. L. Milton. A new numerical method for surface hydrodynamics. *J. Geophys. Res.*, 92:11803–11824, 1987.

Kenneth M. Watson and Bruce J. West. A transport–equation description of nonlinear ocean surface wave interactions. *Journal of Fluid Mechanics*, 70:815–826, 1975.

R.W̃. Yeung. Numerical methods in free-surface flows. *Annual Review of Fluid Mechanics*, 14:395–442, 1982.

Vladimir Zakharov. Stability of periodic waves of finite amplitude on the surface of a deep fluid. *Journal of Applied Mechanics and Technical Physics*, 9:190–194, 1968.

VITA

MARIA KAKLEAS

mariakakleas@yahoo.com

Objective

To teach mathematics as a full-time instructor with tenure at the junior college level.

Education

-Ph.D. in Mathematics, University of Illinois at Chicago, 5/09

-M.S. in Mathematics, Loyola University of Chicago, 12/02

-B.S. in Agricultural Engineering and Minor in Spanish, University of Illinois at Urbana-Champaign, 5/99

Work Experience

-Math Instructor & Teaching Assistant, University of Illinois at Chicago, 8/03 - present

-Teaching Fellow, National Teacher's Academy & Hugh Manley High School in Chicago, IL, 5/07 -5/08

-Adjunct Math Instructor, Wilbur Wright College in Chicago, IL, 1/03 - 8/03

-Adjunct Math Instructor, Oakton Community College in Des Plaines & Skokie, IL, 1/03 - 8/03

-Lecturer & Teaching Assistant, Loyola University of Chicago, 8/01 - 12/02

-Math Supervisor & Tutor, Huntington Learning Center in Park Ridge, IL, 1/01 - 8/01

-Mechanical Engineer, United Conveyor Corporation in Waukegan, IL, 8/99 - 1/01

Courses Taught

Elementary Algebra, Intermediate Algebra, College Algebra, PreCalculus, Differential Calculus, Integral Calculus, Multivariable Calculus, Business Calculus, Advanced Math for Business, Math for Elementary School Teachers, Finite Mathematics, Statistics, Linear Algebra

Educational Software

MATLAB, Maple, Cognitive Tutor, MyMathLab

Fellowship

Scientists, Kids and Teachers (SKIT) Fellowship, Funded by the NSF, 5/07 - 5/08

Colloquium

Numerical Simulation of a Weakly Nonlinear Model for Water Waves with Viscosity, University of Illinois at Chicago, 11/19/08

Professional Memberships

-Mathematical Graduate Student Association, Vice-President at UIC, 8/05 - 7/06

-Association of Mathematical Society

-Association for Women in Mathematics

-Society for Industrial and Applied Mathematics