KdV Solitons in Direct Numerical Simulation of the Navier-Stokes Equation

Jake Dynes
Montclair State University

Follow this and additional works at: https://digitalcommons.montclair.edu/etd

Part of the Mathematics Commons

Recommended Citation
https://digitalcommons.montclair.edu/etd/403

This Thesis is brought to you for free and open access by Montclair State University Digital Commons. It has been accepted for inclusion in Theses, Dissertations and Culminating Projects by an authorized administrator of Montclair State University Digital Commons. For more information, please contact digitalcommons@montclair.edu.
MONTCLAIR STATE UNIVERSITY

KdV Solitons in Direct Numerical Simulation of the Navier-Stokes Equation

by

Jake Dynes

A Master’s Thesis Submitted to the Faculty of
Montclair State University
In Partial Fulfillment of the Requirements
For the Degree of
Master of Science
August 2015

College of Science and Mathematics
Department of Mathematics

Thesis Committee:
Dr. A. David Trubatch, Thesis Advisor
Dr. Philip A. Yecko, Committee Member
Dr. Bogdan G. Nita, Committee Member
Abstract

Tracking the behavior of an interface present between two fluids is critical to a wide array of different fields. Much can be learned through lab work and experimentation, however there are many limitations involved. Solving the Navier-Stokes equations with interfaces is very difficult, and in many cases intractable. One way to combat this is to make assumptions to help reduce the equation (e.g. KdV), and make it more easily studied. An entirely different approach is to use Direct Numerical Simulation on the N-S equation, weather it be by use of front-tracking, VOF, etc. In this thesis, we compare these two separate ways of studying solitons in an attempt to (i) verify the derivation of KdV, and (ii) verify the DNS code. In this thesis, we make use of a Volume of Fluid (VOF) multi-phase open source 3D simulator called PARIS, which is currently in active development and makes use of the Navier-Stokes PDE. Using this software we study KdV (Korteweg and de Vries) solitons, which are solitary non-linear waves that retain their shape and travel at a constant speed. The KdV equation is derived from Navier-Stokes under several assumptions, including small amplitude and long waves, with maximal balance in the asymptotic model, as well as zero viscosity in the bulk. An advantage of using DNS over wave-tank experimentation is our ability to implement parameters which are difficult or impossible to execute in the physical world (e.g. zero viscosity). Due to how it is derived, KdV solitons should be approximate solutions of the full set of equations, and should emerge from DNS with the appropriate initial conditions. We study KdV solitons by altering the values of the small parameters (\( \epsilon \) and \( \delta \)), and comparing the measured velocity of the traveling peak to the velocity we would expect to see. An interpolation formula applied over three cells (the cell where the apex is located, and it’s two closest neighbors) is used to accurately reconstruct and measure the peak. The results presented display the correlation between the actual/measured values of the amplitude and velocity of the peak. As we increase the value of our small parameters, the error with which the simulator predicts the velocity becomes higher. We also show that as epsilon is increased, the apex starts to shrink as dispersion takes effect, which is what we would expect to see as the small parameters leave the KdV regime.
KdV Solitons in Direct Numerical Simulation of the Navier-Stokes Equation

A THESIS

Submitted in partial fulfillment of the requirements for the degree of Masters in Mathematics

by

Jake Dynes
Montclair State University
Montclair, NJ
2015
Acknowledgments

I would first like to thank my parents (Ann and Robert), who have supported me unconditionally in every way. I would also like to extend my sincerest of thanks to the faculty of Montclair State who have always shown great interest and care in my education. Thank you to Dr. Phil Yecko for getting me started with CFD, and putting up with all of my super-long emails. Finally, a very special thank you to Dr. David Trubatch for supporting me in every way throughout my studies, and for always patiently answering my questions no matter how many times I repeatedly asked them.
# Contents

Acknowledgments .................................................. ii
List of Figures ......................................................... v
List of Tables ........................................................ vi

## 1 Introduction

## 2 Physical Principles

2.1 Bulk Fluid Properties ........................................... 3
   2.1.1 Conservation of Mass ...................................... 3
   2.1.2 Conservation of Momentum .............................. 4
   2.1.3 Newtonian Fluid ........................................... 4
   2.1.4 Incompressible Flow ...................................... 4
   2.1.5 Vorticity ..................................................... 5
   2.1.6 Decomposition of the Flow .............................. 5

2.2 Boundary Conditions........................................... 6
   2.2.1 Fixed Boundary Conditions .............................. 6
   2.2.2 Free-Surface Boundary Conditions ..................... 6
   2.2.3 Kinematic FSBC ........................................... 6
   2.2.4 Dynamic FSBC ........................................... 6
   2.2.5 Viscosity .................................................. 7
   2.2.6 Bernoulli Condition ....................................... 7

## 3 Reduced Equations

3.1 System of Equations ........................................... 10

## 4 Derivation of Non-Dimensional KdV

4.1 Physical solution to KdV ................................. 12

## 5 PARIS: A Parallel, Robust Interface Simulation Code

5.1 VOF Method .................................................. 14
5.2 Height Function ............................................... 15
5.3 VOFI .......................................................... 15
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Simulation 3 wave progression</td>
<td>21</td>
</tr>
<tr>
<td>7.2</td>
<td>Simulation 5 wave progression</td>
<td>22</td>
</tr>
<tr>
<td>7.3</td>
<td>Simulation 7 wave progression</td>
<td>23</td>
</tr>
<tr>
<td>7.4</td>
<td>Simulation 11 wave progression</td>
<td>24</td>
</tr>
<tr>
<td>7.5</td>
<td>Plot of Measured Amplitude vs. Measured Velocity</td>
<td>26</td>
</tr>
<tr>
<td>7.6</td>
<td>Plot of Measured Amplitude vs. Measured Velocity (zoom)</td>
<td>27</td>
</tr>
<tr>
<td>7.7</td>
<td>Plot of Inputted Amplitude vs. Measured Amplitude</td>
<td>27</td>
</tr>
</tbody>
</table>
List of Tables

7.1 Simulation Results ................................................................. 25
B.1 Simulation 1 Data ................................................................. 34
B.2 Simulation 2 Data ................................................................. 35
B.3 Simulation 3 Data ................................................................. 36
B.4 Simulation 4 Data ................................................................. 37
B.5 Simulation 5 Data ................................................................. 38
B.6 Simulation 6 Data ................................................................. 39
B.7 Simulation 7 Data ................................................................. 40
B.8 Simulation 8 Data ................................................................. 41
B.9 Simulation 9 Data ................................................................. 42
B.10 Simulation 10 Data ............................................................. 43
B.11 Simulation 11 Data ............................................................. 44
B.12 Simulation 12 Data ............................................................. 45
Chapter 1

Introduction

J. Scott Russell made the first recorded observation of a solitary wave while riding his horse in Edinburgh, Scotland, in 1834. He witnessed a boat being drawn into a narrow channel create this solitary wave, which carried on for several miles while preserving both its velocity and shape. Later, he did extensive experiments in laboratory-scale wave tanks to deduce, among other things, that the wave’s speed depends on its height, suggesting a nonlinear effect [3]. There was vigorous debate on the linearity of these solitary waves until in 1895, when Korteweg and de Vries made a breakthrough with their famous unidirectional nonlinear equation for shallow water waves, quantitatively confirming Russell’s observations. For a time, most of the applications of the KdV equation dealt only with water waves, but in the 1960’s it was discovered that the KdV equation is universal and arises in wave problems with weak dispersion and weak quadratic nonlinearity which spurred greater interest in the dynamics of KdV [2].

A soliton is a solitary wave that retains its shape and form after colliding with another solitary wave. At the most basic level, solitons are formed from a set of governing equations that are derived from physical laws. Since they have been discovered, they have been studied in different ways based on the scientific limits of the time period. Russell was the first to observe them, and did this through rigorous experimentation. In an attempt to circumvent the unsolvable Navier-Stokes Equation, Korteweg and de Vries developed a set of reduced equations in order to form their KdV Equation. For many years this was the only way to study solitons without directly experimenting with physical water waves, until computers began to become more powerful allowing Direct Numerical Simulation of Navier-Stokes. Using DNS we attempt to verify the validity of the KdV equation, and rectify the connection between direct experimentation of the waves.

A multiphase flow setting is one where more than one medium of fluid or material is interacting with each other simultaneously, and the space where these mediums come in contact with each other is called the interface. From a mathematical point of view, multiphase flow problems are notoriously difficult. Not only are the governing
equations of the fluid flow highly nonlinear, but the position of the phase boundary must generally be found as a part of the solution [14]. Experimentation using physical wave tanks is very expensive. Furthermore, accurately recreating the necessary initial conditions in a physical wave tank is difficult, due to the challenge of accurately creating the initial shape of the wave. Simulation allows us to experiment with such situations, while simultaneously providing a flexible environment conducive to analysis. How to simulate multi-phase flows is a topic of great debate and there are several different methods being used today including: Front tracking, the level-set method, and the Volume of Fluid (VOF) method. In our study we make use of the VOF method, which is becoming a much more viable approach with the ever-increasing processing speed of computing. The VOF method has the added benefit of not having to explicitly track the interface; instead, the interface is reconstructed from the VOF scalar field and the HF-method.

In chapter 2 we cover the physical principles that are necessary to form the set of reduced equations involved in KdV. Chapters 3 and 4 elaborate on the reduced equations and method involved in deriving a non-dimensional KdV equation to be used in our simulation. In chapter 5 we describe discuss the PARIS simulator, and explain some of the functions involved. Chapters 6 and 7 contain the setup of each of our simulations, along with some visual representations of the moving waves. In chapters 8 and 9 we discuss the results gained from the DNS experimentation, and describe some possible future work which could also be studied using PARIS.
Chapter 2

Physical Principles

2.1 Bulk Fluid Properties

2.1.1 Conservation of Mass

The principle of conservation of mass states that mass cannot be created or destroyed. Consider a volume $V$ fixed in space with density $\rho$, velocity $u$ and an outward normal $n$ through a region of the surface called $dS$. This volume can only change if mass flows in or out of $V$ through some boundary $S$. [14]

\[
\int_S \mathbf{n} \cdot (\rho \mathbf{u}) \, dS = -\frac{d}{dt} \int_V \rho \, dV
\]

We can apply the divergence theorem to bring the integrand under a common boundary and obtain,

\[
\int_V \left[ \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right] \, dV = 0. \tag{2.2}
\]

This relation must hold for any arbitrary volume which can only be true if the term inside the brackets is zero everywhere. This brings us to the conservation of mass PDE,

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \tag{2.3}
\]
2.1.2 Conservation of Momentum

The principle of conservation of momentum states that the rate of change of the fluid momentum of a fixed volume V is the difference in momentum flux across the fixed boundary S plus the net forces acting on V. Thus,

\[ \frac{d}{dt} \int_V \rho \mathbf{u} \, dV = - \int_S \mathbf{n} \cdot (\rho \mathbf{u}) \, dS + \int_V \mathbf{f} \, dV + \int_S \mathbf{n} \cdot \mathbf{T} \, dS. \] (2.4)

Here, the left-hand side of the equation represents the rate of change of fluid momentum of V. The first term on the right-hand side represents the momentum flux through the boundary S, the second term is the total body force on V, and the third term is the total surface force where \( \mathbf{T} \) is the symmetric stress tensor. [14] We can apply the divergence theorem again to obtain

\[ \frac{\partial}{\partial t} (\rho \mathbf{u}) = -\nabla \cdot (\rho \mathbf{uu}) + \nabla \cdot \mathbf{T} + \mathbf{f}. \] (2.5)

2.1.3 Newtonian Fluid

We are using water and air in our simulation which are both Newtonian fluids. For such Newtonian fluids, the stress tensor is a linear function of the rate of strain [14],

\[ \mathbf{T} = -p \mathbf{I} + \lambda (\nabla \cdot \mathbf{u}) \mathbf{I} + 2\mu \mathbf{S}. \] (2.6)

Here, \( \mathbf{I} \) is the unit tensor, \( p \) is the pressure, \( \mu \) is the viscosity, \( \lambda \) is the second coefficient of viscosity, and \( \mathbf{S} = \frac{1}{2} (\nabla \mathbf{u} + (\nabla \mathbf{u})^\top) \) is the deformation tensor. Substituting in the stress tensor we obtain the Navier-Stokes equation for fluid flow,

\[ \frac{\partial}{\partial t} (\rho \mathbf{u}) = -\nabla p + \nabla \cdot (\lambda \nabla \cdot \mathbf{u}) + \nabla \cdot (2\mu \mathbf{S}) + \mathbf{f}. \] (2.7)

2.1.4 Incompressible Flow

For this study we are only interested in fluids with a constant density, i.e. \( \rho = \rho_0 \) where \( \rho_0 \) is some constant. We can apply this condition to (2.3) to obtain

\[ \nabla \cdot \mathbf{u} = 0. \] (2.8)

Substituting this condition and the deformation tensor \( \mathbf{S} \) into (2.7) leaves us with the Navier-Stokes equation for incompressible fluid flow,
\[
\frac{\partial \mathbf{u}}{\partial t} + \rho \nabla \cdot (\mathbf{u} \mathbf{u}) = -\nabla p + \mu \Delta \mathbf{u} + \mathbf{f}.
\] (2.9)

2.1.5 Vorticity

Define the vorticity of the system, \( \omega \), as

\[
\omega = \nabla \times \mathbf{u}.
\] (2.10)

Using vector identities (A.1c) and (A.1a), we can re-write (2.9) as

\[
\frac{\partial \mathbf{\omega}}{\partial t} - \nabla \times (\mathbf{u} \times \omega) = \frac{\mu}{\rho} \Delta \omega.
\] (2.11)

Thus, the solution \( \omega = 0 \) implies that the vorticity will remain at zero for the entirety of the simulation.

2.1.6 Decomposition of the Flow

We can also re-write the velocity vector \( \mathbf{u} \) as the sum of a scalar velocity potential \( \phi \) and a vector velocity potential \( \mathbf{a} \) (with \( \nabla \cdot \mathbf{a} = 0 \)) [11], so

\[
\mathbf{u} = \nabla \phi + \nabla \times \mathbf{a}.
\] (2.12)

Then,

\[
\omega = -\Delta \mathbf{a}
\] (2.13)

by usage of vector IDs (A.1h), (A.1j), and (A.1i)

\[
\Delta \phi = 0.
\] (2.14)

This Laplacian condition is very important. It shows that if we know \( \phi \) on the surface, we also know \( \phi \) in the bulk of the fluid.
2.2 Boundary Conditions

2.2.1 Fixed Boundary Conditions

We introduce a top and bottom wall whose velocities are 0. For viscous and incompressible fluids the fluid must not pass through the wall, thus

\[ u \cdot \hat{n} = 0. \]  
\[ (2.15) \]

Additionally, we must specify a slip condition on the walls. For a no-slip boundary condition, we have

\[ u \cdot \hat{t} = 0 \]  
\[ (2.16) \]

2.2.2 Free-Surface Boundary Conditions

Additionally, we introduce a free surface boundary separating the two immiscible (non-mixing) fluids. This boundary can be described explicitly by the function \( z = \eta(x, t) \), or implicitly as the level-set function \( F(x, z, t) = z - \eta(x, t) = 0 \).

2.2.3 Kinematic FSBC

A kinematic free surface boundary condition is a statement that the fluid velocity of the free surface is the fluid velocity at the surface.

\[ \frac{DF}{Dt} = \frac{\partial F}{\partial t} + u \cdot \nabla F = 0 \]  
\[ (2.17) \]

2.2.4 Dynamic FSBC

The dynamic FSBC states that the stress difference between the two fluids is due to surface tension acting on the boundary [14], so,

\[ -[T] \hat{n} = \sigma \kappa \hat{n}. \]  
\[ (2.18) \]

Where \( \sigma \) is the surface tension and \( \kappa \) is the curvature. The normal stress condition is found by dotting \( \hat{n} \) on the left of equation (2.18) so,

\[ -[-p + 2\mu \hat{n} \cdot S \cdot \hat{n}] = \sigma \kappa. \]  
\[ (2.19) \]
The tangential stress condition is found by dotting two different unit tangent vectors (forming a tangent plane), $t^{(k)}$, on the left of equation (2.18),

$$[2\mu t^{(k)} \cdot S \cdot \hat{n}]_S = 0.\quad (2.20)$$

For every quantity in the upper fluid (fluid 2) except for pressure, we can assume to be negligible. Thus,

$$(-p + 2\mu \dot{n} \cdot S \cdot \hat{n})|_S = -p_{\text{Ambient}} + \sigma \kappa \quad (2.21)$$

and

$$2\mu t^{(k)} \cdot S \cdot \hat{n}|_S = 0.\quad (2.22)$$

2.2.5 Viscosity

A zero viscosity (inviscid) fluid is required to derive the KdV equation, and we make that assumption now. Doing so will alter our Navier-Stokes (2.9) equation,

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \nabla \cdot (\mathbf{u} \mathbf{u}) = -\nabla p - \rho \mathbf{g},\quad (2.23)$$

and the normal FSBC,

$$p = p_A + \frac{\sigma \eta_{xx}}{(\eta_x^2 + 1)^{3/2}}.\quad (2.24)$$

Additionally, the tangential FSBC is now trivially satisfied everywhere ($0 = 0$). This leaves us with an insufficient amount of equations to derive KdV, but we can use Bernoulli’s condition to establish the fifth and final equation needed.

2.2.6 Bernoulli Condition

After applying our inviscid assumption we lose the tangential FSBC, but can add another equation to our system of equations by applying the Bernoulli Condition. Inserting $\mathbf{u} = \nabla \phi$ into our Navier-Stokes equation (2.23) yields,

$$\nabla \left( \frac{\partial \phi}{\partial t} + \frac{1}{2} |\nabla \phi|^2 + \frac{1}{\rho} p + g z \right) = 0 \quad (2.25)$$
The term inside the parenthesis must be equal to a constant (Let’s assume this constant is zero). Evaluating this at $z = \eta$ yields

$$\phi_t + \frac{1}{2} (\phi_x^2 + \phi_y^2) = -\frac{1}{\rho} p - gh$$

(2.26)

where $h = h_0 + \eta(x, t)$ and $h_0$ is the average depth of the water. This is our fifth, and final, equation needed to finally derive KdV.
Chapter 3

Reduced Equations

The unit normal of the surface is given by,

\[ \hat{n} = -\frac{\nabla F}{|\nabla F|} = \frac{\eta_x \hat{i} - \hat{k}}{\sqrt{\eta_x^2 + 1}} = \frac{\eta_x}{\sqrt{\eta_x^2 + 1}} \hat{i} - \frac{1}{\sqrt{\eta_x^2 + 1}} \hat{k}. \]  

(3.1)

and thus the unit tangent vector is given by,

\[ \hat{t} = \frac{\hat{i} + \eta_x \hat{k}}{\sqrt{\eta_x^2 + 1}} = \frac{1}{\sqrt{\eta_x^2 + 1}} \hat{i} + \frac{\eta_x}{\sqrt{\eta_x^2 + 1}} \hat{k}. \]  

(3.2)

We can now calculate the curvature \( \kappa \) to be,

\[ \kappa = -\nabla \cdot \hat{n} = -\frac{\eta_{xx}}{(\eta_x^2 + 1)^{3/2}} \]  

(3.3)

using (2.12), and allowing \( \mathbf{a} = c \mathbf{j} \) we can calculate \( \mathbf{u} \) to be

\[ \mathbf{u} = (\phi_x + c_z) \hat{i} + (\phi_z - c_x) \hat{k} \]  

(3.4)

The gradient and Laplacian of \( \mathbf{u} \) are calculated as

\[ \nabla \mathbf{u} = (\phi_{xx} + c_{zz}) \hat{i} + (\phi_{zx} - c_{xz}) \hat{k} + (\phi_{xz} + c_{zx}) \hat{i} \hat{k} + (\phi_{zz} - c_{zz}) \hat{k} \hat{k} \]  

(3.5)

\[ \Delta \mathbf{u} = (c_{xx} + c_{zz}) \hat{i} + (-c_{xx} - c_{zz}) \hat{k} = \nabla \times \Delta c \]  

(3.6)

The symmetric stress tensor \( \mathbf{S} \) is calculated to be
\[
S = (\phi_{xx} + c_{xz}) \hat{t}t + \left( \phi_{xx} - \frac{1}{2}(c_{xx} - c_{zz}) \right) \hat{k}k + \\
\left( \phi_{xz} - \frac{1}{2}(c_{xx} - c_{zz}) \right) \hat{i}i + (\phi_{yy} - c_{xz}) \hat{k}k
\]  

(3.7)

### 3.1 System of Equations

We now have a system of five equations:

\[
\frac{\partial \mathbf{u}}{\partial t} + \rho \nabla \cdot (\mathbf{uu}) = -\nabla p - g \hat{k} \tag{2.23}
\]

\[
\Delta \phi = 0 \tag{2.14}
\]

\[
\frac{\partial F}{\partial t} + \mathbf{u} \cdot \nabla F = 0 \tag{2.17}
\]

\[
p = p_A + \frac{\sigma \eta_{xx}}{(\eta_x^2 + 1)^{3/2}} \tag{2.24}
\]

\[
\phi_t + \frac{1}{2} (\phi_x^2 + \phi_z^2) = -\frac{1}{\rho} p - gh \tag{2.26}
\]
Chapter 4

Derivation of Non-Dimensional KdV

In order to write these equations in non-dimensionalized form we will re-scale our variables (cf.e.g. [6]).

\[ x = lx' \]  \hspace{1cm} (4.1)

\[ z = h_0 z' \]  \hspace{1cm} (4.2)

\[ t = \frac{l}{c_0} t' \]  \hspace{1cm} (4.3)

\[ \eta = a \eta' \]  \hspace{1cm} (4.4)

\[ \phi = \frac{g la}{c_0} \phi' \]  \hspace{1cm} (4.5)

where \( c_0 = \sqrt{g h_0} \). \( l \) is related to the length of the wave in the \( x \) direction, \( c_0 \) is the linear wave speed, \( h_0 \) is the equilibrium height, and \( a \) is the height of the wave from the equilibrium height.

The KdV equation is an asymptotic model for small-amplitude waves with a long wavelength. In order to incorporate this stipulation into our equations we can create small parameters \( \epsilon \) and \( \delta^2 \) such that,

\[ \epsilon = \frac{a}{h_0} \]  \hspace{1cm} (4.6)
\[ \delta = \frac{h_0}{l} \]  

where both \( \epsilon \ll 1 \) and \( \delta \ll 1 \). Note that \( \epsilon \) states that the distance of the wave from the equilibrium point must be small, i.e. small-amplitude waves. Also notice that \( \delta^2 \) states that the wave length must be much larger than the equilibrium level of the water, i.e. a long wave. Now omitting the primes, expanding the pressure term up to order \( \epsilon \), and setting \( \epsilon \sim \delta^2 \), the Euler equations and boundary conditions become

\[
\begin{align*}
\delta^2 \phi_{xx} + \phi_{zz} &= 0 \\
\eta_t + \epsilon \phi_x \eta_x - \frac{1}{\delta^2} \phi_z &= 0 \\
\eta + \phi_t + \frac{1}{2} \left( \epsilon \phi_x^2 + \frac{\epsilon}{\delta^2} \phi_z^2 \right) - \sigma \delta^2 \eta_{xx} &= 0
\end{align*}
\]

where \( \sigma = \frac{a}{h_0 \rho g} \) is the dimensionless Bond number. Referring to a method used in [6], keeping terms up to \( \epsilon \) and \( \delta^2 \), we obtain the dimensionless KdV equation,

\[
\eta_t + \eta_x + \frac{3}{2} \epsilon \eta \eta_x + \frac{1}{6} \delta^2 (1 - 3 \sigma) \eta_{xxx} = 0.
\]

### 4.1 Physical solution to KdV

For our equation for \( \eta \), we use the well-known soliton solution

\[
\eta(x,t) = A \text{sech}^2(x - vt).
\]

Inserting this solution into the following version of the KdV equation,

\[
\eta_t + c \eta_x + p \eta \eta_x + q \eta_{xxx} = 0,
\]

yields,

\[
A = \frac{12q}{p}
\]

\[
v = c + 4q.
\]

Setting \( c, p, \) and \( q \) to the coefficients of the KdV equation in (4.11) yields,
However, we must use a dimensional version of these coefficients for our initial condition in the simulation. After undoing the scaling, and making the assumption that $A = a$ we are left with

$$A = \left(\frac{4}{3}\right) \frac{h_0^3}{l^2} (1 - 3\sigma) \quad (4.16)$$

$$v = c_0 \left(1 + \frac{2}{3}(1 - 3\sigma) \left(\frac{h_0}{l}\right)^2\right) \quad (4.17)$$

After making these selections, we finally have our dimensional equation,

$$z = h_0 + \eta(x, t) = h_0 + \left(\frac{4}{3}\right) \left(\frac{h_0^3}{l^2}\right) (1 - 3\sigma) \operatorname{sech}^2 \left(\frac{1}{l} \left(x - (1 + \frac{2}{3} \left(\frac{h_0}{l}\right)^2 (1 - 3\sigma) c_0 t\right)\right) \quad (4.18)$$

Note that after assuming $A = a$, we show that $\epsilon$ is comparable to $\delta^2$,

$$\epsilon = \frac{4}{3} (1 - 3\sigma) \delta^2. \quad (4.19)$$
Chapter 5

PARIS: A Parallel, Robust Interface Simulation Code

Currently in active development, the VOF multi-phase fluid code PARIS [15], is fully parallel and models three-dimensional flows. PARIS avoids using cumbersome algorithms and structures that can impede the extension of VOF codes for complex multi-physics application areas and as a testbed for new algorithms. PARIS is therefore built on a simple three-dimensional fixed mesh and parallelism is handled by message passing interface (MPI) using ghost layers. The code shows good scalability on several architectures, meaning it can be used efficiently for extensive production runs.

5.1 VOF Method

The Volume of Fluid method is becoming an increasingly useful approach for coding multi-phase flows. VOF is a simple-finite-differences based method which exhibits excellent volume conservation and topological properties [14]. Consider a fixed grid placed upon a domain containing two immiscible fluids. Each of the computational cells contains a fractional value, called the color function, which is the percentage of the cell that is occupied by the reference phase. A value of 1 represents a grid box that is completely filled, while a value of 0 represents a completely empty grid box [7]. The interface is then reconstructed and advected geometrically using sophisticated methods including the recently published height function. One main benefit of VOF over its competitors is that the interface is not tracked, but rather reconstructed from the VOF data, which allows for waves to curl and break without causing issues in the code due to discontinuities. Front tracking also exists in PARIS, but we have exclusively used VOF in this thesis.
5.2 Height Function

The Height Function method was recently incorporated into the PARIS [8]. The method is used to improve the estimates of the normal to the interface and the interface curvature. The method evaluates curvature through derivatives of a function obtained by integrating the color function [10]. This method was implemented into PARIS and helped to increase the accuracy of the curvature computation when taking into account surface tension, which is applicable to our simulations.

5.3 VOFI

The VOFI library is a software package written to compute the volume fraction scalar field from an implicit analytical representation of the interface. In the cells cut by the interface, i.e. where the volume of the cell is a fractional value, the algorithm calculates the first primary, secondary and tertiary coordinate directions. The internal and external limits of integration are then determined by the interface intersections with the cell boundary along the secondary and tertiary directions, while the local heights are computed along the primary direction. The numerical integration in three dimensions of the height function is performed with a double Gauss-Legendre integration [9].
Chapter 6

Simulation Setup

We refer to Engineeringtoolbox.com [1] for the appropriate physical quantities. We input a stationary wave and let gravity and the numerics of PARIS split the wave into two opposite-moving waves and track the movement of the right-moving wave. For this reason we multiply the value of $A$ by 2 before inputting it into our initial condition. We off-set the wave from the center to avoid confusion and put the focus on the right-moving wave. Also, note that PARIS is coded specifically for SI units.

We enable the HYPRE solver for PARIS, as well as the VOFI library [7]. We split the domain into 64 separate processes by setting each of “npx”, “npy”, and “npz” to 4, and parallelize it on the Montclair Kruskal server. The processor on Kruskal is running AMD "Magny-Cour" Opteron 6128MS Eight Core CPU 2.0 GHz with 8x 512 KB L2 Cache and 12 MB L3 Cache Shared, 75W, with 4GB DDR3 1333 MHz ECC/Registered Memory RAM.

Each simulation is parallelized onto 64 processors, and use a dt of .00001 seconds. We have chosen the size of the time-step carefully, as setting it too low will cause unnecessarily large computing time, and setting it too high will negatively effect the accuracy of the simulation. Our time scale for our first simulation is related to the quantity $\nu$; for our first simulation this quantity is approximately 1.2303, and thus the ratio between this quantity and dt is of order $10^{-5}$. Setting $l = 5$ dictates that the length of the tank in the x-direction should be at least 77.5 meters, and judging how large the apex of the wave will be shows that the ceiling of the tank should be at least 10 meters. PARIS and VOFI both require that each VOF grid cell is a perfect cube (As of the completion of this paper there is NO check for this in PARIS!), to easily accommodate this we set the length of the tank in the x-direction to be 80 meters, the length of the tank in the y-direction to be 5 meters, and the length of the tank in the z-direction to be 10 meters. We discretize the x-direction into 512 grid boxes, the y-direction in 32 grid boxes (the minimum for a 64 processes PARIS run), and the z-direction into 64 grid boxes. This ensures that the length of each side of the grid cells is 0.15625, and thus each grid cell is a perfect cube.

Physical parameters are set as follows: $\sigma = 0.0712$ (water surface tension), $\rho = 995.7$
(density of liquid), \( \mu_{\text{air}} = 0 \) (viscosity of the air), \( \mu_{\text{water}} = 0 \) (viscosity of the water), \( g = 9.8 \) (gravity constant). It is important to note that putting in such physically impossible quantities (e.g. \( \mu_{\text{water}} = 0 \)) is an added benefit to conducting this study through DNS, as the KdV equation is derived with the assumption of zero viscosity.

We test our simulations with varying values of epsilon and delta. KdV waves are valid for \( \epsilon \ll 1 \) and \( \delta^2 \ll 1 \), so we start with \( \epsilon \approx \delta^2 \approx 1 \), and work our way up to see what will happen to the wave as it leaves the KdV regime. All of the parameters in each simulation are the exact same with the exception of \( h_0 \), which will range between 1 and 3.75. Altering the value for \( h_0 \) makes changes in the simulation that effect the initial interface condition; raising \( h_0 \) will increase both \( \epsilon \) and \( \delta^2 \).

### 6.1 Choice of Dimensions for Numerical Wave-Tank

For our KdV simulation we want to attain as much resolution as possible in the x and z directions, while at the same time keeping our simulation at a reasonable run-time. Since VOF codes are, by definition, calculated on a grid of rectangles, we have to ensure that the height of our wave is not too small in comparison to the rectangle height, and also that \( l \) is not too big in comparison to the length of our numerical wave-tank.

Ensuring that our grid height is small enough is relatively easy. We only need to check that the height of a single box is small in comparison to the \( A \) value that we calculate.

\[
\frac{\text{box height}}{A} \leq \frac{1}{3}
\]

Ensuring that the length of the tank is big enough to account for the length of the wave is a little bit more difficult. We are using \( \text{sech}^2(x) \) for the level-set function which gives the profile of the wave, and so we can use that to calculate how big we should make the tank.

\[
\text{sech}(x) = \frac{2}{e^x + e^{-x}}
\]

\[
\text{sech}^2(x) = \frac{4}{e^{2x} + e^{-2x} + 2}
\]

For high \( x \) the \( e^{-2x} \) term goes to zero. Bringing the other \( e \) term to the top gives us

\[
\text{sech}^2(x) \approx 4e^{-2x}.
\]
If we wish for this value to be within $p$ digits of accuracy we set

$$4e^{-2x} \approx 10^{-p},$$

and solve for $x$, which gives us,

$$x = \frac{p}{2} \ln 10 + 2.$$

Setting $x$ equal to $\frac{X_{\text{max}}}{l}$ gives

$$\frac{X_{\text{max}}}{l} = \frac{p}{2} \ln 10 + 2$$

$$X_{\text{max}} = \frac{l}{2} p \ln 10 + 2.$$

Setting $p = 5$ gives an $X_{\text{max}}$ of approximately 7.75. This means that we should make the length of our wave tank to be at least twice this value, giving $15.5l$.

### 6.2 Choice of NOUT

In PARIS, the parameter "NOUT" is used to determine how often the calculations of the simulation are outputted to a VTK file. We can use this to our advantage and only print out the time-steps that we wish to take measurements on. Recall that our moving wave has a speed of $v$ which we calculated in an earlier section. Since PARIS is based on a grid, we need to set a certain number of boxes that we wish for the apex of the wave to move. Using $t = \frac{d}{v}$ we get,

$$\text{NOUT} = \frac{(\text{Number of boxes to travel}) \cdot (\text{lateral box length})}{dt \cdot v}.$$

We choose the number of boxes that we want the apex to travel per snapshot to be .5.

### 6.3 Accurately measuring the apex

In order to accurately measure the location and height of the wave, we can fit a curve onto the VOF cell where the apex is located and it's closest two neighbors. Assume that the grid size in the x-direction is $h$ and the grid size in the z-direction is $k$ and let the VOF values of the three cells in question be represented by $v_1$, $v_2$, and $v_3$ ($v_2$
is the location of the apex). The left and right edges of $v_1$ will be represented by $x_1$ and $x_2$, respectively, the left and right edges of $v_2$ will be represented by $x_2$ and $x_3$, respectively, and the left and right edges of $v_3$ will be represented by $x_3$ and $x_4$, respectively. In order for the parabola to fit properly, we know that the evaluated integral of each grid box must be equal to the product of $h, k$, and the corresponding VOF value, so,

$$\int_{x_1}^{x_2} a x^2 + b x + c = h k v_1$$

$$\int_{x_2}^{x_3} a x^2 + b x + c = h k v_2$$

$$\int_{x_3}^{x_4} a x^2 + b x + c = h k v_3$$

After we solve for $a, b, c$, and substitute in $x_2 = x_1 + h, x_3 = x_1 + 2h$ and $x_4 = x_1 + 3h$, we find that the location of the apex on the x-axis is given by,

$$\frac{h(2v_1 - 3v_2 + v_3)}{v_1 - 2v_2 + v_3} + x_1. \quad (6.1)$$

If we insert this solution into our parabola we obtain the height of the apex defined as,

$$- \frac{k(v_1^2 + 13v_2^2 - 7v_2v_3 + v_3^2 - v_1(7v_2 + v_3))}{6(v_1 - 2v_2 + v_3)}. \quad (6.2)$$

We have written a MATLAB function to perform these operations, which can be found in the appendix.
Chapter 7

Simulations

Below, we show snapshots of only a few different simulations. We could put snapshots from every simulation in, however we don’t feel it is necessary to get the point across, and it would take up unnecessary pages. Also, the simulations look alike when they have $\epsilon$ near each other; we have chosen snapshots from simulations which show a large difference. The data tables for every simulation can be found in the Appendix.
7.1 Simulation 3

For this simulation we chose $h_0 = 1.5$, which yields $\epsilon \approx 0.119999$, $\delta^2 = 0.09$, $A \approx 0.179998$, and $v \approx 4.064099$. Below are several other figures which show how the wave progresses through time. The waves are very small here due to the low $\epsilon$; it becomes easier to witness the traveling wave for higher $\epsilon$.

Figure 7.1: Simulation 3 wave progression
7.2 Simulation 5

For this simulation we chose $h_0 = 2$, which yields $\epsilon \approx 0.213332$, $\delta^2 = 0.16$, $A \approx 0.426664$, and $v \approx 4.899420$. Below are several other figures which show how the wave progresses through time. Here it is a bit easier to observe the traveling wave.

\[ t = 0 \text{ seconds (initial condition)} \]
\[ t = 1.754500 \text{ seconds} \]
\[ t = 3.875850 \text{ seconds} \]
\[ t = 6.730900 \text{ seconds} \]

Figure 7.2: Simulation 5 wave progression
7.3 Simulation 7

For this simulation we chose $h_0 = 2.5$, which yields $\epsilon \approx 0.333332$, $\delta^2 = 0.25$, $A \approx 0.833330$, and $v \approx 5.77470$. Below are several other figures which show how the wave progresses through time. One can now witness ripples on the wave.

Figure 7.3: Simulation 7 wave progression
7.4 Simulation 11

For this simulation we chose $h_0 = 3.5$, which yields $\epsilon \approx 0.653332$, $\delta^2 = 0.49$, $A \approx 2.286663$, and $v \approx 7.769780$. 

Figure 7.4: Simulation 11 wave progression
7.5 Results

In examination of the waves, we see several properties of solitons. For each simulation, the waves take a few seconds to separate completely, but do produce two opposite-moving solitons. For the low $\epsilon$ simulations, we see a slight hump which seems to have a symmetric shape progress forward, before colliding with its counterpart. As the waves collide, it appears that they form a single curve very gracefully, and seem to pass through each other. As $\epsilon$ increased, the waves take longer to separate, and also do not seem to have a completely symmetric shape. Looking at the figures for simulation 11, one can see the the wave is shorter on one end, or 'back-loaded'. In order to get quantitative evidence of what appears to be happening, we make use of our interpolation algorithm.

Due to the oscillations in the data, we have to establish a set of criteria for proper analysis. Since the length of the tank is 80 meters, and the location of the wave at the start of the simulation is 20 meters from the left, the waves will be the furthest apart from each other when the right-moving wave reaches a location of 40 meters. Due to some rippling in the higher epsilon simulations, we have chosen to start our measurements at 40 meters, and end our measurements at 50 meters. These ripples can easily been seen in higher simulation videos. We track the speed and height, and take the average of all of the outputted timesteps for our final values of the velocity and height.

<table>
<thead>
<tr>
<th>$h_0$</th>
<th>$\epsilon$</th>
<th>$A_{\text{input}}$</th>
<th>$A_{\text{measured}}$± std. dev.</th>
<th>$v_{\text{predicted}}$</th>
<th>$v_{\text{measured}}$± std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>0.053</td>
<td>0.053</td>
<td>0.208±0.001</td>
<td>3.214</td>
<td>3.109±0.009</td>
</tr>
<tr>
<td>1.25</td>
<td>0.083</td>
<td>0.104</td>
<td>0.249±0.001</td>
<td>3.646</td>
<td>4.126±0.018</td>
</tr>
<tr>
<td>1.50</td>
<td>0.120</td>
<td>0.180</td>
<td>0.344±0.004</td>
<td>4.064</td>
<td>4.064±0.023</td>
</tr>
<tr>
<td>1.75</td>
<td>0.163</td>
<td>0.286</td>
<td>0.445±0.001</td>
<td>4.479</td>
<td>4.349±0.022</td>
</tr>
<tr>
<td>2.00</td>
<td>0.213</td>
<td>0.427</td>
<td>0.526±0.006</td>
<td>4.899</td>
<td>5.068±0.081</td>
</tr>
<tr>
<td>2.25</td>
<td>0.270</td>
<td>0.607</td>
<td>0.677±0.004</td>
<td>5.330</td>
<td>5.014±0.074</td>
</tr>
<tr>
<td>2.50</td>
<td>0.333</td>
<td>0.833</td>
<td>0.846±0.009</td>
<td>5.775</td>
<td>5.131±0.141</td>
</tr>
<tr>
<td>2.75</td>
<td>0.403</td>
<td>1.109</td>
<td>1.019±0.131</td>
<td>6.238</td>
<td>5.703±0.242</td>
</tr>
<tr>
<td>3.00</td>
<td>0.480</td>
<td>1.440</td>
<td>1.232±0.017</td>
<td>6.723</td>
<td>5.728±0.255</td>
</tr>
<tr>
<td>3.25</td>
<td>0.563</td>
<td>1.831</td>
<td>1.459±0.026</td>
<td>7.233</td>
<td>5.805±0.805</td>
</tr>
<tr>
<td>3.50</td>
<td>0.653</td>
<td>2.287</td>
<td>1.690±0.278</td>
<td>7.770</td>
<td>6.232±0.214</td>
</tr>
<tr>
<td>3.75</td>
<td>0.750</td>
<td>2.812</td>
<td>1.439±0.028</td>
<td>8.335</td>
<td>6.737±0.318</td>
</tr>
</tbody>
</table>
In the below figure, we plot the measured amplitude vs. the measured velocity in each of the simulations. There are slight oscillations, which is why we show the average ± std. dev. What we see is an upward trend, with most of the measured values being slightly under the predicted values curve. It may not predict the anticipated values very well, but what it does seem to show is that there is a nonlinear effect. We also plot the Inputted Amplitude vs. the Measured Amplitude. As shown in the graph, the measured amplitude is slightly higher than what we expect for low values of $\epsilon$. We also see that the amplitude shrinks as $\epsilon$ increases, which is what we predicted to see.

Figure 7.5: Plot of Measured Amplitude vs. Measured Velocity

- dotted red line: predicted velocity
- blue circles + vertical error bar: measured velocity ± std. dev.
- blue circles + horizontal error bar: measured amplitude ± std. dev.
Figure 7.6: Plot of Measured Amplitude vs. Measured Velocity (zoom)

Figure 7.7: Plot of Inputted Amplitude vs. Measured Amplitude

solid red line: predicted amplitude
blue circles + error bar: measured amplitude ± std. dev.
Chapter 8

Conclusion

We have shown that for low values of $\epsilon$, the PARIS simulator produces a wave which has properties of solitons. For low $\epsilon$ the waves clearly stay at a constant height and travel at a constant speed (with some slight variations in the data). As $\epsilon$ increases, we observe that the waves shrink, caused by dispersion, as one would expect. Improvements could be made on analyzing the data by making use of the Height-Function values to get a more accurate reading of the curvature of the apex [this was, in fact, the purpose of the Height Function]. Also, perhaps it may have been prudent to include simulations with real-world levels of viscosity, as we are using all of the other real-world values for air, water, surface tension, etc.
Chapter 9

Future Work

A multidimensional generalization of the KdV equation was discovered in 1970 by Kadomtsev and Petviashvili (KP Equation), whose soliton solutions can be observed daily on certain beaches. Along with a phase shift, some of the distinctive nonlinear interactions of KP solitons are related to the stem height, which can be very important in the modeling of tsunami propagation. Satellite images reveal local X-type interactions for the 2011 Japanese tsunami, which made the effects of the tsunami even worse [2]. The accurate modeling of solitons is therefore a worthy area of study, as its implications are far reaching. PARIS has 3D VOF capabilities, and the KP equation would be an appropriate next topic to study using the software.
Appendix A

Notation, Definitions and Identities

Vectors are represented by a bold lowercase letter and are comprised of the orthonormal unit vectors \( \mathbf{i}, \mathbf{j}, \) and \( \mathbf{k} \) in the \( x, y, \) and \( z \) directions, respectively.

Matrices (tensors) are represented by a bold uppercase letter, e.g. \( \mathbf{A} \).

Two vectors written next to each other without a space represents the dyadic product.

\[
\text{If } \mathbf{a} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k} \quad \text{and} \quad \mathbf{b} = f\mathbf{i} + g\mathbf{j} + h\mathbf{k} \quad \text{then}
\]

\[
\mathbf{a}\mathbf{b} = af\mathbf{ii} + ag\mathbf{ij} + ah\mathbf{ik} +
bf\mathbf{ji} + bg\mathbf{jj} + bh\mathbf{jk} +
cf\mathbf{ki} + cg\mathbf{kj} + ch\mathbf{kk}
\]

The Transpose operator, denoted \( \mathbf{A}^T \), swaps the diads of every term in a tensor.

\[
(\mathbf{a}\mathbf{b})^T = af\mathbf{ii} + bf\mathbf{ij} + cf\mathbf{ik} +
ag\mathbf{ji} + bg\mathbf{jj} + cg\mathbf{jk} +
ah\mathbf{ki} + bh\mathbf{kj} + ch\mathbf{kk} = \mathbf{b}\mathbf{a}
\]

The gradient in Cartesian coordinates is defined as

\[
\nabla = \frac{\partial}{\partial x} \mathbf{i} + \frac{\partial}{\partial y} \mathbf{j} + \frac{\partial}{\partial z} \mathbf{k}.
\]
The *Laplacian* scalar operator is defined as

\[
\Delta = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}.
\]

The material derivative describes the rate of change of a particular physical quantity. We will use this frequently in the calculation of the change in momentum. If \(q(x, y, z, t)\) is some physical quantity at a particular point in space and time, then the material derivative of \(q\) is

\[
\frac{Dq}{Dt} = \frac{\partial q}{\partial x} \frac{dx}{dt} + \frac{\partial q}{\partial y} \frac{dy}{dt} + \frac{\partial q}{\partial z} \frac{dz}{dt} + \frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q
\]

We denote the jump in a quantity from one fluid to another between a surface \(S\) as,

\[
[q]_S = q_2 - q_1
\]

where \(q_1\) and \(q_2\) are the quantities of the lower and upper fluids, respectively.

The Divergence Theorem relates the integral of a field over a surface,

\[
\int_S \mathbf{n} \cdot \mathbf{T} dS = \int_V \nabla \cdot \mathbf{T} dV
\]

where \(S\) is the boundary of \(V\) and \(\mathbf{T}\) is a tensor or vector field.

**Algebra Identities**

\[
\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}
\]

\[
\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{a} \cdot \mathbf{b})\mathbf{c} \neq (\mathbf{a} \times \mathbf{b}) \times \mathbf{c}
\]

**Product Rule Identities**

\[
\nabla (\mathbf{a} \cdot \mathbf{b}) = \mathbf{b} \cdot \nabla \mathbf{a} + \mathbf{a} \cdot \nabla \mathbf{b} + \mathbf{b} \times (\nabla \times \mathbf{a}) + \mathbf{a} \times (\nabla \times \mathbf{b})
\]

\[
\nabla \cdot (\mathbf{a} \mathbf{b}) = \mathbf{a} \cdot \nabla \mathbf{b} + \mathbf{b} \nabla \cdot \mathbf{a}
\]

\[
\nabla \cdot (f\mathbf{a}) = (\nabla f) \cdot \mathbf{a} + f(\nabla \cdot \mathbf{a})
\]

\[
\nabla \cdot (\mathbf{a} \times \mathbf{b}) = \mathbf{b} \cdot (\nabla \times \mathbf{a}) - \mathbf{a} \cdot (\nabla \times \mathbf{b})
\]

\[
\nabla \times (\mathbf{a} \times \mathbf{b}) = \mathbf{b} \cdot \nabla \mathbf{a} - \mathbf{a} \cdot \nabla \mathbf{b} - (\nabla \cdot \mathbf{a})\mathbf{b} + (\nabla \cdot \mathbf{b})\mathbf{a}
\]
Second Derivative Identities

\[ \nabla \times \nabla f = 0 \]  \hspace{1cm} (A.1h)

\[ \nabla \cdot (\nabla \times a) = 0 \]  \hspace{1cm} (A.1i)

\[ \nabla \times (\nabla \times a) = \nabla (\nabla \cdot a) - \Delta a \]  \hspace{1cm} (A.1j)
Appendix B

Simulation Raw Data Tables

Below are the measured results in each of the 12 simulations. We have provided data from when the traveling wave is at an x-location of 40 meters, to the time it is 50 meters. Logic would suggest to measure from 35 meters to 45 meters (as this is when the waves would be further apart), however, in some of the simulations there are ripples which have not yet been absorbed into the wave. These ripples cause a bump which is slightly higher than the anticipated wave peak, and throw off the interpolation algorithm. Note that this ripples can easily be seen for the high epsilon simulations. For this reason, we have chosen to choose are data from when the wave is at a distance of 40 meters to a distance of around 50 meters. Some of the very high epsilon simulations start to merge together before the 50 meter mark, and so we have removed those data points from our measurements as well. Also, simulation 1 received a CFL error before the 50 meter mark, so we have measured from 35 meters to 45 meters in this simulation, because the value of epsilon is not high enough to cause the ripples which throw off the algorithm.
B.1 Simulation 1 Data

For this simulation we chose $h_0 = 1$, which yields $\epsilon \approx 0.053332$, $\delta^2 = 0.04$, $A \approx 0.053332$, and $v \approx 3.213973$. Our average amplitude measured $0.208165 \pm 0.000159$, and the average velocity measured $3.109099 \pm 0.009463$.

Table B.1: Simulation 1 Data

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Apparent Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.957200</td>
<td>35.192522</td>
<td>0.208419</td>
<td>3.094259</td>
</tr>
<tr>
<td>5.103000</td>
<td>35.644531</td>
<td>0.208394</td>
<td>3.100198</td>
</tr>
<tr>
<td>5.248800</td>
<td>36.096540</td>
<td>0.208370</td>
<td>3.100198</td>
</tr>
<tr>
<td>5.394600</td>
<td>36.551339</td>
<td>0.208344</td>
<td>3.119335</td>
</tr>
<tr>
<td>5.540400</td>
<td>37.003348</td>
<td>0.208319</td>
<td>3.119335</td>
</tr>
<tr>
<td>5.686200</td>
<td>37.456140</td>
<td>0.208295</td>
<td>3.105570</td>
</tr>
<tr>
<td>5.832000</td>
<td>37.909091</td>
<td>0.208268</td>
<td>3.106657</td>
</tr>
<tr>
<td>5.977800</td>
<td>38.362269</td>
<td>0.208243</td>
<td>3.108214</td>
</tr>
<tr>
<td>6.123600</td>
<td>38.815341</td>
<td>0.208218</td>
<td>3.107492</td>
</tr>
<tr>
<td>6.269400</td>
<td>39.268973</td>
<td>0.208193</td>
<td>3.111333</td>
</tr>
<tr>
<td>6.415200</td>
<td>39.722222</td>
<td>0.208166</td>
<td>3.108704</td>
</tr>
<tr>
<td>6.561000</td>
<td>40.176136</td>
<td>0.208141</td>
<td>3.113266</td>
</tr>
<tr>
<td>6.706800</td>
<td>40.627894</td>
<td>0.208115</td>
<td>3.098472</td>
</tr>
<tr>
<td>6.852600</td>
<td>41.084906</td>
<td>0.208088</td>
<td>3.134514</td>
</tr>
<tr>
<td>6.998400</td>
<td>41.536458</td>
<td>0.208063</td>
<td>3.097069</td>
</tr>
<tr>
<td>7.144200</td>
<td>41.990741</td>
<td>0.208037</td>
<td>3.115792</td>
</tr>
<tr>
<td>7.290000</td>
<td>42.445023</td>
<td>0.208011</td>
<td>3.115792</td>
</tr>
<tr>
<td>7.435800</td>
<td>42.899306</td>
<td>0.207985</td>
<td>3.115792</td>
</tr>
<tr>
<td>7.581600</td>
<td>43.352005</td>
<td>0.207958</td>
<td>3.104933</td>
</tr>
<tr>
<td>7.727400</td>
<td>43.806014</td>
<td>0.207932</td>
<td>3.113919</td>
</tr>
<tr>
<td>7.873200</td>
<td>44.260817</td>
<td>0.207906</td>
<td>3.119363</td>
</tr>
</tbody>
</table>
B.2 Simulation 2 Data

For this simulation we chose $h_0 = 1.25$, which yields $\epsilon \approx 0.083332$, $\delta^2 = 0.0625$, $A \approx 0.104165$, and $v \approx 3.645831$. Our average amplitude measured $0.249252 \pm 0.000191$, and the average velocity measured $4.126345 \pm 0.018055$.

Table B.2: Simulation 2 Data

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.243800</td>
<td>35.207674</td>
<td>0.251343</td>
<td>3.939899</td>
</tr>
<tr>
<td>4.372400</td>
<td>35.717147</td>
<td>0.251101</td>
<td>3.961690</td>
</tr>
<tr>
<td>4.501000</td>
<td>36.229968</td>
<td>0.250876</td>
<td>3.987718</td>
</tr>
<tr>
<td>4.629600</td>
<td>36.745477</td>
<td>0.250668</td>
<td>4.008624</td>
</tr>
<tr>
<td>4.758200</td>
<td>37.262500</td>
<td>0.250475</td>
<td>4.020397</td>
</tr>
<tr>
<td>4.886800</td>
<td>37.782939</td>
<td>0.250301</td>
<td>4.046961</td>
</tr>
<tr>
<td>5.015400</td>
<td>38.304795</td>
<td>0.250143</td>
<td>4.057973</td>
</tr>
<tr>
<td>5.144000</td>
<td>38.829195</td>
<td>0.250003</td>
<td>4.077766</td>
</tr>
<tr>
<td>5.272600</td>
<td>39.353299</td>
<td>0.249878</td>
<td>4.075454</td>
</tr>
<tr>
<td>5.401200</td>
<td>39.879464</td>
<td>0.249768</td>
<td>4.091490</td>
</tr>
<tr>
<td>5.529800</td>
<td>40.407130</td>
<td>0.249674</td>
<td>4.103157</td>
</tr>
<tr>
<td>5.658400</td>
<td>40.935299</td>
<td>0.249591</td>
<td>4.107069</td>
</tr>
<tr>
<td>5.787000</td>
<td>41.465127</td>
<td>0.249518</td>
<td>4.119965</td>
</tr>
<tr>
<td>5.915600</td>
<td>41.995018</td>
<td>0.249457</td>
<td>4.120461</td>
</tr>
<tr>
<td>6.044200</td>
<td>42.523321</td>
<td>0.249401</td>
<td>4.108109</td>
</tr>
<tr>
<td>6.172800</td>
<td>43.053768</td>
<td>0.249355</td>
<td>4.124786</td>
</tr>
<tr>
<td>6.301400</td>
<td>43.584559</td>
<td>0.249313</td>
<td>4.127453</td>
</tr>
<tr>
<td>6.430000</td>
<td>44.113806</td>
<td>0.249275</td>
<td>4.115452</td>
</tr>
<tr>
<td>6.558600</td>
<td>44.643842</td>
<td>0.249241</td>
<td>4.121586</td>
</tr>
<tr>
<td>6.687200</td>
<td>45.174632</td>
<td>0.249211</td>
<td>4.127453</td>
</tr>
<tr>
<td>6.815800</td>
<td>45.704291</td>
<td>0.249182</td>
<td>4.118652</td>
</tr>
<tr>
<td>6.944400</td>
<td>46.233915</td>
<td>0.249157</td>
<td>4.118386</td>
</tr>
<tr>
<td>7.073000</td>
<td>46.764706</td>
<td>0.249132</td>
<td>4.127453</td>
</tr>
<tr>
<td>7.201600</td>
<td>47.294776</td>
<td>0.249108</td>
<td>4.121853</td>
</tr>
<tr>
<td>7.330200</td>
<td>47.824337</td>
<td>0.249087</td>
<td>4.117893</td>
</tr>
<tr>
<td>7.458800</td>
<td>48.355877</td>
<td>0.249070</td>
<td>4.133280</td>
</tr>
<tr>
<td>7.587400</td>
<td>48.887311</td>
<td>0.249057</td>
<td>4.132455</td>
</tr>
<tr>
<td>7.716000</td>
<td>49.419981</td>
<td>0.249052</td>
<td>4.142072</td>
</tr>
<tr>
<td>7.844600</td>
<td>49.954327</td>
<td>0.249061</td>
<td>4.155100</td>
</tr>
<tr>
<td>7.973200</td>
<td>50.492424</td>
<td>0.249095</td>
<td>4.184272</td>
</tr>
</tbody>
</table>
### B.3 Simulation 3 Data

For this simulation we chose $h_0 = 1.5$, which yields $\epsilon \approx 0.1199988$, $\delta^2 = 0.09$, $A \approx 0.179998$, and $v \approx 4.064099$. Our average amplitude measured $0.344389 \pm 0.003791$, and the average velocity measured $4.064553 \pm 0.023055$.

Table B.3: Simulation 3 Data

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.593580</td>
<td>40.043851</td>
<td>0.337566</td>
<td>4.025484</td>
</tr>
<tr>
<td>4.689680</td>
<td>40.430069</td>
<td>0.338204</td>
<td>4.018922</td>
</tr>
<tr>
<td>4.785780</td>
<td>40.819664</td>
<td>0.338826</td>
<td>4.025964</td>
</tr>
<tr>
<td>4.881880</td>
<td>41.204427</td>
<td>0.339429</td>
<td>4.031871</td>
</tr>
<tr>
<td>4.977980</td>
<td>41.592773</td>
<td>0.340019</td>
<td>4.041065</td>
</tr>
<tr>
<td>5.074080</td>
<td>41.981309</td>
<td>0.340591</td>
<td>4.043037</td>
</tr>
<tr>
<td>5.170180</td>
<td>42.370192</td>
<td>0.341152</td>
<td>4.046649</td>
</tr>
<tr>
<td>5.266280</td>
<td>42.759123</td>
<td>0.341696</td>
<td>4.047143</td>
</tr>
<tr>
<td>5.362380</td>
<td>43.149547</td>
<td>0.342230</td>
<td>4.062686</td>
</tr>
<tr>
<td>5.458480</td>
<td>43.538800</td>
<td>0.342747</td>
<td>4.050498</td>
</tr>
<tr>
<td>5.554580</td>
<td>43.929571</td>
<td>0.343256</td>
<td>4.066298</td>
</tr>
<tr>
<td>5.650680</td>
<td>44.319770</td>
<td>0.343749</td>
<td>4.060340</td>
</tr>
<tr>
<td>5.746780</td>
<td>44.710546</td>
<td>0.344235</td>
<td>4.066349</td>
</tr>
<tr>
<td>5.842880</td>
<td>45.101376</td>
<td>0.344706</td>
<td>4.066918</td>
</tr>
<tr>
<td>5.938980</td>
<td>45.492507</td>
<td>0.345169</td>
<td>4.070040</td>
</tr>
<tr>
<td>6.035080</td>
<td>45.884034</td>
<td>0.345619</td>
<td>4.074158</td>
</tr>
<tr>
<td>6.131180</td>
<td>46.275741</td>
<td>0.346062</td>
<td>4.076032</td>
</tr>
<tr>
<td>6.227280</td>
<td>46.667424</td>
<td>0.346494</td>
<td>4.075787</td>
</tr>
<tr>
<td>6.323380</td>
<td>47.059456</td>
<td>0.346919</td>
<td>4.079416</td>
</tr>
<tr>
<td>6.419480</td>
<td>47.451509</td>
<td>0.347334</td>
<td>4.079636</td>
</tr>
<tr>
<td>6.515580</td>
<td>47.844370</td>
<td>0.347742</td>
<td>4.088044</td>
</tr>
<tr>
<td>6.611680</td>
<td>48.236734</td>
<td>0.348143</td>
<td>4.082879</td>
</tr>
<tr>
<td>6.707780</td>
<td>48.629401</td>
<td>0.348539</td>
<td>4.086026</td>
</tr>
<tr>
<td>6.803880</td>
<td>49.022447</td>
<td>0.348930</td>
<td>4.089966</td>
</tr>
<tr>
<td>6.899980</td>
<td>49.415799</td>
<td>0.349321</td>
<td>4.093147</td>
</tr>
<tr>
<td>6.996080</td>
<td>49.809611</td>
<td>0.349709</td>
<td>4.097947</td>
</tr>
<tr>
<td>7.092180</td>
<td>50.203298</td>
<td>0.350105</td>
<td>4.096632</td>
</tr>
</tbody>
</table>
B.4 Simulation 4 Data

For this simulation we chose $h_0 = 1.75$, which yields $\epsilon \approx 0.163332$, $\delta^2 = 0.1225$, $A \approx 0.285831$, and $v \approx 4.479456$. Our average amplitude measured $0.445222 \pm 0.000529$, and the average velocity measured $4.349239 \pm 0.021997$.

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.602400</td>
<td>40.371568</td>
<td>0.444309</td>
<td>4.311769</td>
</tr>
<tr>
<td>4.707000</td>
<td>40.823119</td>
<td>0.444422</td>
<td>4.316928</td>
</tr>
<tr>
<td>4.811600</td>
<td>41.275156</td>
<td>0.444529</td>
<td>4.321581</td>
</tr>
<tr>
<td>4.916200</td>
<td>41.727651</td>
<td>0.444628</td>
<td>4.325951</td>
</tr>
<tr>
<td>5.020800</td>
<td>42.180462</td>
<td>0.444722</td>
<td>4.328975</td>
</tr>
<tr>
<td>5.125400</td>
<td>42.633630</td>
<td>0.444809</td>
<td>4.332397</td>
</tr>
<tr>
<td>5.230000</td>
<td>43.087164</td>
<td>0.444891</td>
<td>4.335884</td>
</tr>
<tr>
<td>5.334600</td>
<td>43.541034</td>
<td>0.444968</td>
<td>4.339107</td>
</tr>
<tr>
<td>5.439200</td>
<td>43.995153</td>
<td>0.445040</td>
<td>4.341481</td>
</tr>
<tr>
<td>5.543800</td>
<td>44.449546</td>
<td>0.445107</td>
<td>4.344100</td>
</tr>
<tr>
<td>5.648400</td>
<td>44.904184</td>
<td>0.445171</td>
<td>4.346438</td>
</tr>
<tr>
<td>5.753000</td>
<td>45.359038</td>
<td>0.445230</td>
<td>4.348517</td>
</tr>
<tr>
<td>5.857600</td>
<td>45.814122</td>
<td>0.445287</td>
<td>4.350700</td>
</tr>
<tr>
<td>5.962200</td>
<td>46.269415</td>
<td>0.445342</td>
<td>4.352710</td>
</tr>
<tr>
<td>6.066800</td>
<td>46.724920</td>
<td>0.445397</td>
<td>4.354728</td>
</tr>
<tr>
<td>6.171400</td>
<td>47.180685</td>
<td>0.445453</td>
<td>4.357223</td>
</tr>
<tr>
<td>6.276000</td>
<td>47.636688</td>
<td>0.445512</td>
<td>4.359492</td>
</tr>
<tr>
<td>6.380600</td>
<td>48.093063</td>
<td>0.445579</td>
<td>4.363050</td>
</tr>
<tr>
<td>6.485200</td>
<td>48.549865</td>
<td>0.445660</td>
<td>4.367126</td>
</tr>
<tr>
<td>6.589800</td>
<td>49.007142</td>
<td>0.445761</td>
<td>4.371678</td>
</tr>
<tr>
<td>6.694400</td>
<td>49.465114</td>
<td>0.445895</td>
<td>4.378317</td>
</tr>
<tr>
<td>6.799000</td>
<td>49.923965</td>
<td>0.446076</td>
<td>4.386723</td>
</tr>
<tr>
<td>6.903600</td>
<td>50.383957</td>
<td>0.446328</td>
<td>4.397624</td>
</tr>
</tbody>
</table>
B.5 Simulation 5 Data

For this simulation we chose $h_0 = 2$, which yields $\epsilon \approx 0.213332$, $\delta^2 = 0.16$, $A \approx 0.426664$, and $v \approx 4.899420$. Our average amplitude measured $0.526579 \pm 0.005797$, and the average velocity measured $5.068991 \pm 0.081395$.

Table B.5: Simulation 5 Data

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.529800</td>
<td>40.021734</td>
<td>0.538785</td>
<td>4.989616</td>
</tr>
<tr>
<td>4.625500</td>
<td>40.516633</td>
<td>0.536759</td>
<td>5.171359</td>
</tr>
<tr>
<td>4.721200</td>
<td>41.016287</td>
<td>0.534970</td>
<td>5.221045</td>
</tr>
<tr>
<td>4.816900</td>
<td>41.515625</td>
<td>0.533388</td>
<td>5.217742</td>
</tr>
<tr>
<td>4.912600</td>
<td>42.012500</td>
<td>0.531976</td>
<td>5.192006</td>
</tr>
<tr>
<td>5.008300</td>
<td>42.506665</td>
<td>0.530701</td>
<td>5.163685</td>
</tr>
<tr>
<td>5.104000</td>
<td>42.998371</td>
<td>0.529545</td>
<td>5.137995</td>
</tr>
<tr>
<td>5.199700</td>
<td>43.485809</td>
<td>0.528480</td>
<td>5.093399</td>
</tr>
<tr>
<td>5.295400</td>
<td>43.970722</td>
<td>0.527496</td>
<td>5.067011</td>
</tr>
<tr>
<td>5.391100</td>
<td>44.453510</td>
<td>0.526574</td>
<td>5.044804</td>
</tr>
<tr>
<td>5.486800</td>
<td>44.934575</td>
<td>0.525711</td>
<td>5.026805</td>
</tr>
<tr>
<td>5.582500</td>
<td>45.414335</td>
<td>0.524896</td>
<td>5.013160</td>
</tr>
<tr>
<td>5.678200</td>
<td>45.892745</td>
<td>0.524128</td>
<td>4.999064</td>
</tr>
<tr>
<td>5.773900</td>
<td>46.370739</td>
<td>0.523404</td>
<td>4.994709</td>
</tr>
<tr>
<td>5.869600</td>
<td>46.848693</td>
<td>0.522723</td>
<td>4.994294</td>
</tr>
<tr>
<td>5.965300</td>
<td>47.326031</td>
<td>0.522088</td>
<td>4.987861</td>
</tr>
<tr>
<td>6.061000</td>
<td>47.802835</td>
<td>0.521506</td>
<td>4.982279</td>
</tr>
<tr>
<td>6.156700</td>
<td>48.281250</td>
<td>0.520983</td>
<td>4.999111</td>
</tr>
<tr>
<td>6.252400</td>
<td>48.760980</td>
<td>0.520534</td>
<td>5.012850</td>
</tr>
<tr>
<td>6.348100</td>
<td>49.242527</td>
<td>0.520186</td>
<td>5.031844</td>
</tr>
<tr>
<td>6.443800</td>
<td>49.726563</td>
<td>0.519969</td>
<td>5.057840</td>
</tr>
<tr>
<td>6.539500</td>
<td>50.216480</td>
<td>0.519937</td>
<td>5.119310</td>
</tr>
</tbody>
</table>
B.6 Simulation 6 Data

For this simulation we chose $h_0 = 2.25$, which yields $\epsilon \approx 0.269999$, $\delta^2 = 0.2025$, $A \approx 0.607497$, and $v \approx 5.329665$. Our average amplitude measured $0.677150 \pm 0.003966$, and the average velocity measured $5.014648 \pm 0.074582$.

Table B.6: Simulation 6 Data

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.224000</td>
<td>40.118024</td>
<td>0.684160</td>
<td>4.930093</td>
</tr>
<tr>
<td>4.312000</td>
<td>40.552404</td>
<td>0.683542</td>
<td>4.936138</td>
</tr>
<tr>
<td>4.400000</td>
<td>40.987481</td>
<td>0.682901</td>
<td>4.944054</td>
</tr>
<tr>
<td>4.488000</td>
<td>41.422904</td>
<td>0.682245</td>
<td>4.947989</td>
</tr>
<tr>
<td>4.576000</td>
<td>41.859472</td>
<td>0.681580</td>
<td>4.961005</td>
</tr>
<tr>
<td>4.664000</td>
<td>42.297072</td>
<td>0.680907</td>
<td>4.972721</td>
</tr>
<tr>
<td>4.752000</td>
<td>42.734869</td>
<td>0.680235</td>
<td>4.974976</td>
</tr>
<tr>
<td>4.840000</td>
<td>43.172771</td>
<td>0.679563</td>
<td>4.976150</td>
</tr>
<tr>
<td>4.928000</td>
<td>43.610832</td>
<td>0.678893</td>
<td>4.977969</td>
</tr>
<tr>
<td>5.016000</td>
<td>44.049022</td>
<td>0.678228</td>
<td>4.979427</td>
</tr>
<tr>
<td>5.104000</td>
<td>44.487903</td>
<td>0.677567</td>
<td>4.987292</td>
</tr>
<tr>
<td>5.192000</td>
<td>44.926948</td>
<td>0.676916</td>
<td>4.989146</td>
</tr>
<tr>
<td>5.280000</td>
<td>45.366291</td>
<td>0.676274</td>
<td>4.992533</td>
</tr>
<tr>
<td>5.368000</td>
<td>45.806084</td>
<td>0.675649</td>
<td>4.997653</td>
</tr>
<tr>
<td>5.456000</td>
<td>46.245902</td>
<td>0.675045</td>
<td>4.997923</td>
</tr>
<tr>
<td>5.544000</td>
<td>46.687190</td>
<td>0.674467</td>
<td>5.014636</td>
</tr>
<tr>
<td>5.632000</td>
<td>47.128972</td>
<td>0.673925</td>
<td>5.020250</td>
</tr>
<tr>
<td>5.720000</td>
<td>47.571791</td>
<td>0.673431</td>
<td>5.032034</td>
</tr>
<tr>
<td>5.808000</td>
<td>48.015678</td>
<td>0.673002</td>
<td>5.044179</td>
</tr>
<tr>
<td>5.896000</td>
<td>48.461125</td>
<td>0.672662</td>
<td>5.061899</td>
</tr>
<tr>
<td>5.984000</td>
<td>48.908963</td>
<td>0.672443</td>
<td>5.089060</td>
</tr>
<tr>
<td>6.072000</td>
<td>49.359321</td>
<td>0.672388</td>
<td>5.117708</td>
</tr>
<tr>
<td>6.160000</td>
<td>49.814693</td>
<td>0.672553</td>
<td>5.174682</td>
</tr>
<tr>
<td>6.248000</td>
<td>50.275112</td>
<td>0.673019</td>
<td>5.232030</td>
</tr>
</tbody>
</table>
B.7 Simulation 7 Data

For this simulation we chose \( h_0 = 2.5 \), which yields \( \epsilon \approx 0.333332, \delta^2 = 0.25, \ A \approx 0.833330, \) and \( v \approx 5.77470 \). Our average amplitude measured \( 0.846063 \pm 0.008690 \), and the average velocity measured \( 5.131061 \pm 0.140857 \).

Table B.7: Simulation 7 Data

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.096590</td>
<td>40.017244</td>
<td>0.861758</td>
<td>4.995641</td>
</tr>
<tr>
<td>4.180770</td>
<td>40.421699</td>
<td>0.860343</td>
<td>4.982208</td>
</tr>
<tr>
<td>4.261950</td>
<td>40.828053</td>
<td>0.858906</td>
<td>5.005587</td>
</tr>
<tr>
<td>4.343130</td>
<td>41.235665</td>
<td>0.857470</td>
<td>5.021091</td>
</tr>
<tr>
<td>4.424310</td>
<td>41.644085</td>
<td>0.856030</td>
<td>5.031041</td>
</tr>
<tr>
<td>4.505490</td>
<td>42.052776</td>
<td>0.854588</td>
<td>5.034381</td>
</tr>
<tr>
<td>4.586670</td>
<td>42.461953</td>
<td>0.853151</td>
<td>5.040369</td>
</tr>
<tr>
<td>4.667850</td>
<td>42.872221</td>
<td>0.851714</td>
<td>5.053797</td>
</tr>
<tr>
<td>4.749030</td>
<td>43.282014</td>
<td>0.850285</td>
<td>5.047961</td>
</tr>
<tr>
<td>4.830210</td>
<td>43.693355</td>
<td>0.848864</td>
<td>5.067019</td>
</tr>
<tr>
<td>4.911390</td>
<td>44.104552</td>
<td>0.847451</td>
<td>5.065260</td>
</tr>
<tr>
<td>4.992570</td>
<td>44.516241</td>
<td>0.846055</td>
<td>5.071302</td>
</tr>
<tr>
<td>5.073750</td>
<td>44.929511</td>
<td>0.844678</td>
<td>5.090792</td>
</tr>
<tr>
<td>5.154930</td>
<td>45.342319</td>
<td>0.843330</td>
<td>5.085088</td>
</tr>
<tr>
<td>5.236110</td>
<td>45.756600</td>
<td>0.842019</td>
<td>5.103242</td>
</tr>
<tr>
<td>5.317290</td>
<td>46.171472</td>
<td>0.840758</td>
<td>5.110525</td>
</tr>
<tr>
<td>5.398470</td>
<td>46.587171</td>
<td>0.839563</td>
<td>5.120704</td>
</tr>
<tr>
<td>5.479650</td>
<td>47.005072</td>
<td>0.838464</td>
<td>5.147831</td>
</tr>
<tr>
<td>5.560830</td>
<td>47.424368</td>
<td>0.837485</td>
<td>5.165020</td>
</tr>
<tr>
<td>5.642010</td>
<td>47.845528</td>
<td>0.836670</td>
<td>5.187979</td>
</tr>
<tr>
<td>5.723190</td>
<td>48.270693</td>
<td>0.836080</td>
<td>5.237301</td>
</tr>
<tr>
<td>5.804370</td>
<td>48.699638</td>
<td>0.835783</td>
<td>5.283886</td>
</tr>
<tr>
<td>5.885550</td>
<td>49.133803</td>
<td>0.835875</td>
<td>5.348169</td>
</tr>
<tr>
<td>5.966730</td>
<td>49.575054</td>
<td>0.836487</td>
<td>5.435469</td>
</tr>
<tr>
<td>6.047910</td>
<td>50.025187</td>
<td>0.837782</td>
<td>5.544868</td>
</tr>
</tbody>
</table>
B.8 Simulation 8 Data

For this simulation we chose $h_0 = 2.75$, which yields $\epsilon \approx 0.403332$, $\delta^2 = 0.3025$, $A \approx 1.109163$, and $v \approx 6.23826$. Our average amplitude measured $1.019603 \pm 0.013139$, and the average velocity measured $5.703958 \pm 0.242199$.

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.130500</td>
<td>40.181494</td>
<td>1.047012</td>
<td>6.275796</td>
</tr>
<tr>
<td>4.205600</td>
<td>40.636131</td>
<td>1.043420</td>
<td>6.053762</td>
</tr>
<tr>
<td>4.280700</td>
<td>41.076793</td>
<td>1.040142</td>
<td>5.867661</td>
</tr>
<tr>
<td>4.355800</td>
<td>41.512877</td>
<td>1.037111</td>
<td>5.806720</td>
</tr>
<tr>
<td>4.430900</td>
<td>41.936849</td>
<td>1.034281</td>
<td>5.645428</td>
</tr>
<tr>
<td>4.506000</td>
<td>42.357694</td>
<td>1.031584</td>
<td>5.603790</td>
</tr>
<tr>
<td>4.581100</td>
<td>42.776659</td>
<td>1.029012</td>
<td>5.578769</td>
</tr>
<tr>
<td>4.656200</td>
<td>43.192426</td>
<td>1.026541</td>
<td>5.536173</td>
</tr>
<tr>
<td>4.731300</td>
<td>43.606530</td>
<td>1.024151</td>
<td>5.514043</td>
</tr>
<tr>
<td>4.806400</td>
<td>44.019439</td>
<td>1.021849</td>
<td>5.498118</td>
</tr>
<tr>
<td>4.881500</td>
<td>44.433594</td>
<td>1.019635</td>
<td>5.514711</td>
</tr>
<tr>
<td>4.956600</td>
<td>44.844177</td>
<td>1.017504</td>
<td>5.467153</td>
</tr>
<tr>
<td>5.031700</td>
<td>45.259052</td>
<td>1.015473</td>
<td>5.524297</td>
</tr>
<tr>
<td>5.106800</td>
<td>45.671517</td>
<td>1.013545</td>
<td>5.492214</td>
</tr>
<tr>
<td>5.181900</td>
<td>46.085524</td>
<td>1.011744</td>
<td>5.512746</td>
</tr>
<tr>
<td>5.257000</td>
<td>46.501865</td>
<td>1.010087</td>
<td>5.543821</td>
</tr>
<tr>
<td>5.332100</td>
<td>46.918209</td>
<td>1.008605</td>
<td>5.543867</td>
</tr>
<tr>
<td>5.407200</td>
<td>47.337155</td>
<td>1.007351</td>
<td>5.578509</td>
</tr>
<tr>
<td>5.482300</td>
<td>47.760217</td>
<td>1.006369</td>
<td>5.633308</td>
</tr>
<tr>
<td>5.557400</td>
<td>48.184841</td>
<td>1.005730</td>
<td>5.654122</td>
</tr>
<tr>
<td>5.632500</td>
<td>48.615048</td>
<td>1.005521</td>
<td>5.728451</td>
</tr>
<tr>
<td>5.707600</td>
<td>49.048653</td>
<td>1.005853</td>
<td>5.773704</td>
</tr>
<tr>
<td>5.782700</td>
<td>49.492793</td>
<td>1.006881</td>
<td>5.913979</td>
</tr>
<tr>
<td>5.857800</td>
<td>49.947951</td>
<td>1.008798</td>
<td>6.060692</td>
</tr>
<tr>
<td>5.932900</td>
<td>50.419363</td>
<td>1.011877</td>
<td>6.277121</td>
</tr>
</tbody>
</table>
B.9 Simulation 9 Data

For this simulation we chose $h_0 = 3$, which yields $\epsilon \approx 0.479999$, $\delta^2 = 0.36$, $A \approx 1.439996$, and $v \approx 6.723496$. Our average amplitude measured $1.232878 \pm 0.016665$, and the average velocity measured $5.728892 \pm 0.255181$.

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.985660</td>
<td>40.029109</td>
<td>1.266207</td>
<td>5.619009</td>
</tr>
<tr>
<td>4.055380</td>
<td>40.417425</td>
<td>1.262745</td>
<td>5.569650</td>
</tr>
<tr>
<td>4.125100</td>
<td>40.805904</td>
<td>1.259325</td>
<td>5.571986</td>
</tr>
<tr>
<td>4.194820</td>
<td>41.190869</td>
<td>1.255975</td>
<td>5.521600</td>
</tr>
<tr>
<td>4.264540</td>
<td>41.577785</td>
<td>1.252668</td>
<td>5.549568</td>
</tr>
<tr>
<td>4.334260</td>
<td>41.964529</td>
<td>1.249426</td>
<td>5.547096</td>
</tr>
<tr>
<td>4.403980</td>
<td>42.351058</td>
<td>1.246248</td>
<td>5.544019</td>
</tr>
<tr>
<td>4.473700</td>
<td>42.738869</td>
<td>1.243138</td>
<td>5.562407</td>
</tr>
<tr>
<td>4.543420</td>
<td>43.127126</td>
<td>1.240096</td>
<td>5.568803</td>
</tr>
<tr>
<td>4.613140</td>
<td>43.514545</td>
<td>1.237128</td>
<td>5.556785</td>
</tr>
<tr>
<td>4.682860</td>
<td>43.904616</td>
<td>1.234260</td>
<td>5.594830</td>
</tr>
<tr>
<td>4.752580</td>
<td>44.295158</td>
<td>1.231483</td>
<td>5.601565</td>
</tr>
<tr>
<td>4.822300</td>
<td>44.687280</td>
<td>1.228827</td>
<td>5.624244</td>
</tr>
<tr>
<td>4.892020</td>
<td>45.080610</td>
<td>1.226299</td>
<td>5.641563</td>
</tr>
<tr>
<td>4.961740</td>
<td>45.474803</td>
<td>1.223932</td>
<td>5.653956</td>
</tr>
<tr>
<td>5.031460</td>
<td>45.873379</td>
<td>1.221757</td>
<td>5.716805</td>
</tr>
<tr>
<td>5.101180</td>
<td>46.278019</td>
<td>1.219831</td>
<td>5.803791</td>
</tr>
<tr>
<td>5.170900</td>
<td>46.683735</td>
<td>1.218256</td>
<td>5.819213</td>
</tr>
<tr>
<td>5.240620</td>
<td>47.062044</td>
<td>1.216794</td>
<td>5.426117</td>
</tr>
<tr>
<td>5.310340</td>
<td>47.463768</td>
<td>1.215649</td>
<td>5.761967</td>
</tr>
<tr>
<td>5.380060</td>
<td>47.869662</td>
<td>1.214944</td>
<td>5.826077</td>
</tr>
<tr>
<td>5.449780</td>
<td>48.284062</td>
<td>1.214826</td>
<td>5.939465</td>
</tr>
<tr>
<td>5.519500</td>
<td>48.705013</td>
<td>1.215422</td>
<td>6.037739</td>
</tr>
<tr>
<td>5.589220</td>
<td>49.135417</td>
<td>1.216897</td>
<td>6.173319</td>
</tr>
<tr>
<td>5.658940</td>
<td>49.569909</td>
<td>1.219462</td>
<td>6.231967</td>
</tr>
<tr>
<td>5.728660</td>
<td>50.022229</td>
<td>1.223229</td>
<td>6.487663</td>
</tr>
</tbody>
</table>
B.10 Simulation 10 Data

For this simulation we chose $h_0 = 3.25$, which yields $\epsilon \approx 0.563332$, $\delta^2 = 0.4225$, $A \approx 1.830829$, and $v \approx 7.233186$. Our average amplitude measured $1.459080 \pm 0.026201$, and the average velocity measured $5.805843 \pm 0.805085$.

Table B.10: Simulation 10 Data

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.952800</td>
<td>40.235002</td>
<td>1.509692</td>
<td>4.766720</td>
</tr>
<tr>
<td>4.017600</td>
<td>40.523516</td>
<td>1.506313</td>
<td>4.452382</td>
</tr>
<tr>
<td>4.082400</td>
<td>40.809842</td>
<td>1.502299</td>
<td>4.418613</td>
</tr>
<tr>
<td>4.147200</td>
<td>41.103329</td>
<td>1.497728</td>
<td>4.529124</td>
</tr>
<tr>
<td>4.212000</td>
<td>41.408763</td>
<td>1.492721</td>
<td>4.713479</td>
</tr>
<tr>
<td>4.276800</td>
<td>41.728265</td>
<td>1.487403</td>
<td>4.930593</td>
</tr>
<tr>
<td>4.341600</td>
<td>42.064824</td>
<td>1.481967</td>
<td>5.193814</td>
</tr>
<tr>
<td>4.406400</td>
<td>42.414949</td>
<td>1.476489</td>
<td>5.403158</td>
</tr>
<tr>
<td>4.471200</td>
<td>42.774884</td>
<td>1.471112</td>
<td>5.554556</td>
</tr>
<tr>
<td>4.536000</td>
<td>43.145461</td>
<td>1.465921</td>
<td>5.718782</td>
</tr>
<tr>
<td>4.600800</td>
<td>43.524922</td>
<td>1.460939</td>
<td>5.855870</td>
</tr>
<tr>
<td>4.665600</td>
<td>43.904619</td>
<td>1.456244</td>
<td>5.859526</td>
</tr>
<tr>
<td>4.730400</td>
<td>44.293471</td>
<td>1.451861</td>
<td>6.000802</td>
</tr>
<tr>
<td>4.795200</td>
<td>44.683080</td>
<td>1.447777</td>
<td>6.012488</td>
</tr>
<tr>
<td>4.860000</td>
<td>45.072158</td>
<td>1.444050</td>
<td>6.004282</td>
</tr>
<tr>
<td>4.924800</td>
<td>45.465153</td>
<td>1.440713</td>
<td>6.064736</td>
</tr>
<tr>
<td>4.989600</td>
<td>45.875207</td>
<td>1.437848</td>
<td>6.328006</td>
</tr>
<tr>
<td>5.054400</td>
<td>46.244202</td>
<td>1.435521</td>
<td>5.694366</td>
</tr>
<tr>
<td>5.119200</td>
<td>46.639038</td>
<td>1.433401</td>
<td>6.093142</td>
</tr>
<tr>
<td>5.184000</td>
<td>47.033142</td>
<td>1.431887</td>
<td>6.081849</td>
</tr>
<tr>
<td>5.248800</td>
<td>47.439156</td>
<td>1.431125</td>
<td>6.265655</td>
</tr>
<tr>
<td>5.313600</td>
<td>47.847977</td>
<td>1.431189</td>
<td>6.308965</td>
</tr>
<tr>
<td>5.378400</td>
<td>48.270089</td>
<td>1.432290</td>
<td>6.514078</td>
</tr>
<tr>
<td>5.443200</td>
<td>48.701146</td>
<td>1.434652</td>
<td>6.652103</td>
</tr>
<tr>
<td>5.508000</td>
<td>49.135098</td>
<td>1.438506</td>
<td>6.696789</td>
</tr>
<tr>
<td>5.572800</td>
<td>49.594373</td>
<td>1.443973</td>
<td>7.087581</td>
</tr>
<tr>
<td>5.637600</td>
<td>50.084022</td>
<td>1.451537</td>
<td>7.556311</td>
</tr>
</tbody>
</table>
B.11 Simulation 11 Data

For this simulation we chose $h_0 = 3.5$, which yields $\epsilon \approx 0.653332$, $\delta^2 = 0.49$, $A \approx 2.286663$, and $v \approx 7.769780$. Our average amplitude measured $1.690076 \pm 0.027785$, and the average velocity measured $6.232373 \pm 0.214223$.

Table B.11: Simulation 11 Data

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.979800</td>
<td>40.022969</td>
<td>1.746571</td>
<td>6.259966</td>
</tr>
<tr>
<td>4.040100</td>
<td>40.401012</td>
<td>1.739151</td>
<td>6.269362</td>
</tr>
<tr>
<td>4.100400</td>
<td>40.775810</td>
<td>1.732127</td>
<td>6.215562</td>
</tr>
<tr>
<td>4.160700</td>
<td>41.149465</td>
<td>1.725458</td>
<td>6.196595</td>
</tr>
<tr>
<td>4.221000</td>
<td>41.520625</td>
<td>1.719111</td>
<td>6.155230</td>
</tr>
<tr>
<td>4.281300</td>
<td>41.887781</td>
<td>1.713052</td>
<td>6.088826</td>
</tr>
<tr>
<td>4.341600</td>
<td>42.255379</td>
<td>1.707257</td>
<td>6.096151</td>
</tr>
<tr>
<td>4.401900</td>
<td>42.619017</td>
<td>1.701711</td>
<td>6.030484</td>
</tr>
<tr>
<td>4.462200</td>
<td>42.984440</td>
<td>1.696409</td>
<td>6.060084</td>
</tr>
<tr>
<td>4.522500</td>
<td>43.347838</td>
<td>1.691353</td>
<td>6.026487</td>
</tr>
<tr>
<td>4.582800</td>
<td>43.713255</td>
<td>1.686560</td>
<td>6.059987</td>
</tr>
<tr>
<td>4.643100</td>
<td>44.077413</td>
<td>1.682047</td>
<td>6.039112</td>
</tr>
<tr>
<td>4.703400</td>
<td>44.444483</td>
<td>1.677827</td>
<td>6.087390</td>
</tr>
<tr>
<td>4.763700</td>
<td>44.811551</td>
<td>1.673975</td>
<td>6.087365</td>
</tr>
<tr>
<td>4.824000</td>
<td>45.182868</td>
<td>1.670475</td>
<td>6.157823</td>
</tr>
<tr>
<td>4.884300</td>
<td>45.554434</td>
<td>1.667438</td>
<td>6.161967</td>
</tr>
<tr>
<td>4.944600</td>
<td>45.929573</td>
<td>1.664907</td>
<td>6.221201</td>
</tr>
<tr>
<td>5.004900</td>
<td>46.309451</td>
<td>1.662946</td>
<td>6.299800</td>
</tr>
<tr>
<td>5.065200</td>
<td>46.692517</td>
<td>1.661666</td>
<td>6.352682</td>
</tr>
<tr>
<td>5.125500</td>
<td>47.081935</td>
<td>1.661186</td>
<td>6.458008</td>
</tr>
<tr>
<td>5.185800</td>
<td>47.473580</td>
<td>1.661613</td>
<td>6.494933</td>
</tr>
<tr>
<td>5.246100</td>
<td>47.874409</td>
<td>1.663088</td>
<td>6.647248</td>
</tr>
<tr>
<td>5.306400</td>
<td>48.289171</td>
<td>1.665828</td>
<td>6.878307</td>
</tr>
</tbody>
</table>
B.12 Simulation 12 Data

For this simulation we chose \( h_0 = 3.75 \), which yields \( \epsilon \approx 0.7499990 \), \( \delta^2 = 0.5625 \), \( A \approx 2.812495 \), and \( v \approx 8.335491 \). Our average amplitude measured \( 1.439186 \pm 0.027592 \), and the average velocity measured \( 6.737920 \pm 0.317559 \).

Table B.12: Simulation 12 Data

<table>
<thead>
<tr>
<th>Actual Time (in seconds)</th>
<th>Approximate Position of Apex (meters)</th>
<th>Approximate Height of Apex (meters)</th>
<th>Measured Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.710520</td>
<td>40.022969</td>
<td>1.496571</td>
<td>6.714265</td>
</tr>
<tr>
<td>3.776110</td>
<td>40.463084</td>
<td>1.487955</td>
<td>6.710099</td>
</tr>
<tr>
<td>3.841700</td>
<td>40.901042</td>
<td>1.479866</td>
<td>6.677195</td>
</tr>
<tr>
<td>3.907290</td>
<td>41.335142</td>
<td>1.472246</td>
<td>6.618395</td>
</tr>
<tr>
<td>3.972880</td>
<td>41.765878</td>
<td>1.465043</td>
<td>6.567096</td>
</tr>
<tr>
<td>4.038470</td>
<td>42.193943</td>
<td>1.458204</td>
<td>6.526380</td>
</tr>
<tr>
<td>4.104060</td>
<td>42.619017</td>
<td>1.451711</td>
<td>6.480775</td>
</tr>
<tr>
<td>4.169650</td>
<td>43.045254</td>
<td>1.445552</td>
<td>6.498503</td>
</tr>
<tr>
<td>4.235240</td>
<td>43.469483</td>
<td>1.439722</td>
<td>6.467888</td>
</tr>
<tr>
<td>4.300830</td>
<td>43.895349</td>
<td>1.434270</td>
<td>6.492848</td>
</tr>
<tr>
<td>4.366420</td>
<td>44.321215</td>
<td>1.429198</td>
<td>6.492846</td>
</tr>
<tr>
<td>4.432010</td>
<td>44.751436</td>
<td>1.424592</td>
<td>6.559253</td>
</tr>
<tr>
<td>4.497600</td>
<td>45.182868</td>
<td>1.420475</td>
<td>6.577706</td>
</tr>
<tr>
<td>4.563190</td>
<td>45.616452</td>
<td>1.416979</td>
<td>6.610524</td>
</tr>
<tr>
<td>4.628780</td>
<td>46.055584</td>
<td>1.414183</td>
<td>6.695106</td>
</tr>
<tr>
<td>4.694370</td>
<td>46.500053</td>
<td>1.412215</td>
<td>6.776473</td>
</tr>
<tr>
<td>4.759960</td>
<td>46.952501</td>
<td>1.411250</td>
<td>6.898125</td>
</tr>
<tr>
<td>4.825550</td>
<td>47.408299</td>
<td>1.411473</td>
<td>6.949193</td>
</tr>
<tr>
<td>4.891140</td>
<td>47.874409</td>
<td>1.413088</td>
<td>7.106423</td>
</tr>
<tr>
<td>4.956730</td>
<td>48.359397</td>
<td>1.416428</td>
<td>7.394233</td>
</tr>
<tr>
<td>5.022320</td>
<td>48.863324</td>
<td>1.421893</td>
<td>7.682995</td>
</tr>
<tr>
<td>5.087910</td>
<td>49.392633</td>
<td>1.429921</td>
<td>8.069966</td>
</tr>
<tr>
<td>5.153500</td>
<td>49.958092</td>
<td>1.441117</td>
<td>8.621111</td>
</tr>
</tbody>
</table>
Appendix C

Source Code

C.1 Sample PARIS Input file

Below is one of the PARIS input files we used in our simulation.

```
! Test file for VOF test case
!
! 4 grid points per dimension per domain
! 8 mpi processes
!
! The input file for running the PARIS code
! Parameters are read using a namelist statement.
! Blank lines and commented lines are ignored.
! Location of the parameters in the list is not important.

&PARENTERALS

! name of the namelist
!===============================================================================
!
! General parameters

TWOPHASE = T    ! TwoPhase: Is the flow two-phase?
DoVOF         = T
DoFront       = F
GetPropertiesFromFront = F ! T: uses Front-Tracking data to compute mu, rho and Surface Tension.
                           ! F: uses VOF data to compute mu and rho.

ZeroReynolds = T
Implicit      = F
hpyre         = T  ! T: uses hyprepackage, F: uses SOR solver
restart      = F   ! T: start the domain from a previous simulation
restartFront = F   ! T: start the front from a previous simulation
```

46
restartAverages = F
nBackup = 200       ! number of time steps between backups are kept.

NSTEP  = 10000000   ! maximum number of time steps
EndTime = 20        ! When to stop simulation

MaxDt  = 1e-1       ! maximum size of time step
dtFlag = 1          ! 1: fixed dt; 2: fixed CFL
dt     = 0.00001d0   ! dt in case of dtFlag=1
CFL    = 0.25d0
MAXERROR= 1d-6      ! Residual for Poisson solver
MAXERRORVOL = 1d-4

! Numerical parameters

ITIME_SCHEME = 1
               ! time scheme: 1:first order Euler, 2: second order quasi
               ! Crank-Nicolson

MAXIT  = 2000
BETA   = 1.2

! parameters for linear solver
U_init = 0

! Output parameters
termout = 1
ICOut  = T          ! output initial condition
NOUT   = 1922        ! write the solution to file every nout time steps
output_format = 2    ! 1: tecplot 2: vtk
out_path = 'out'     ! name of the output directory

nstats = 1         ! number of time steps between flow statistics
calculations

! Grid parameters

npx    = 4
npy    = 4
npz    = 4

! number of processors in x,y,z direction

Nx     = 512
Ny     = 32
Nz     = 64
Ng     = 2

! grid size in x,y,z direction and number of ghost cells

XLENGTH = 80.0d0
YLENGTH = 5.0d0
ZLENGTH = 10.0d0

! domain size in x,y,z direction

read_x = F ! read the grid file for x-grid; If true xLength and
read_y = F
read_z = F

x_file = 'xh.dat' ! input file for xh (Nx+1 points)
y_file = 'yh.dat'
z_file = 'zh.dat'

xform = 0.0 !1.0
yform = 0.0 !1.0
zform = 0.0 ! non-uniformity of the grid if not reading an input
! file
! 0:uniform; +:clustered at center; -:clustered near ends

!=================================================================================
! Flow parameters
GX = 0.0
GY = 0.0
GZ = -9.8d0

! Components of the gravity in x,y,z direction

BDRY_COND = 1 1 0 1 1 0
!Type of boundary condition in x,y,z direction: 0:wall
! 1:periodic 2:shear
! x- y- z- x+ y+ z+

dPdX = 0.0
dPdY = 0.0
dPdZ = 0.0

! Px, Py, Pz: pressure gradients in case of pressure
! driven channel flow
! Px = (P(xLength)-P(0))/xLength

RH01 = 1.165d0
MU1 = 0.0d0

! rho1, mu1 : density and viscosity of the matrix phase

RH02 = 995.7d0
MU2 = 0.0d0

! rho2, mu2 : properties of the drop
SIGMA = .0712d0
NumBubble = 1
   ! number of bubbles

xyzrad(1, 1) = 40.0
xyzrad(2, 1) = 0.0
xyzrad(3, 1) = 5.0
xyzrad(4, 1) = 3.0
   ! Initial bubble size and location : x,y,z,radius

MaxPoint = 2000000
MaxElem = 4000000
MaxFront = 100
amin = 0.32
amax = 0.96
aspmx = 1.54
smooth = T   !smooth the interface
nsmooth = 10  !every nsmooth time steps
nregrid = 10  !regrid the front every nregrid time steps

BUOYANCYCASE = 1
   ! BuoyancyCase : determines what density will be
   ! subtracted from the gravity
   ! body force.
   ! 0: rro=0, 1: rro=rho1, 2: rro=rho2, 3: rro=average(rho)

/  
! end of the namelist

C.2 VTK2Array

The VTK2Array tool is used to pool all of the VOF data from each separate process and bring them into a single 3D array. PARIS outputs $n$ VTK(Visual Toolkit) VOF files, where $n$ is the number of processes that the computation has been split on, for each desired output timestep. A sample name for a VTK file is "VOF00000-00000.vtk", which is located in the output folder of the VOF test. The first five digits represent the outputted time step, and the last five digits represent which process was outputted. Dr. Zaleski has added this tool to the stable PARIS download so that all users may make use of it.

function A = vtk2array(folderloc, time, numprocesses)

%vtk2array
% Returns a three dimensional array of the color function at a specific
% point in time. To use, simply input the VTK folder, desired
% timestep, and number of processes. This should work for variables
% other than the color
% function, but only if the value is described at the center of the grid
% cell. Please feel free to contact me with any comments/concerns.
%
% Note: numprocesses MUST be a perfect cube!
%
% Sample Call:
% folderloc = '/home/jakedynes/paris-stable-8proc-ex/Tests/VOF/out/VTK/';
% time = '00000';
% numprocesses = 8;
% A = vtk2array(folderloc, time, numprocesses)
%
% Jake Dynes, March 2015 (dynesj1@montclair.edu)
%
% GPL Licence
%
% This file is part of PARIS.
%
% PARIS is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation, either version 3 of the License, or
% (at your option) any later version.
%
% PARIS is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with PARIS. If not, see <http://www.gnu.org/licenses/>.

for p=0:numprocesses-1
    % import data from the current processor
    if p<10
        intProcessor=strcat('0000',int2str(p));
    elseif p<100 & p>9
        intProcessor=strcat('000',int2str(p));
    elseif p<1000 & p>99
        intProcessor=strcat('00',int2str(p));
    elseif p<10000 & p>999
        intProcessor=strcat('0',int2str(p));
    elseif p<100000 & p>9999
        intProcessor=strcat('',int2str(p));
    else
        intProcessor=strcat('',int2str(p));
    end
%
end
strFileloc=strcat(folderloc, 'VOF', time, '-', intProcessor, '.vtk');
matData=dlmread(strFileloc);
proccube=nthroot(numprocesses,3);

% some setup from first processor
if p==0

    % get grid size (specified in 'input')
    intXgriddim=(matData(5,2)-1)*proccube;
    intYgriddim=(matData(5,3)-1)*proccube;
    intZgriddim=(matData(5,4)-1)*proccube;
    procXsize=intXgriddim/proccube;
    procYsize=intYgriddim/proccube;
    procZsize=intZgriddim/proccube;

    % create VOF array (dimensions)
    matVOF=zeros(intZgriddim,intXgriddim,intYgriddim);
end

% create raw data matrices (points and values)
intTotalCellswithGhost=matData(5,2)*matData(5,3)*matData(5,4);
matPoints=zeros(intTotalCellswithGhost,3);
for i=1:intTotalCellswithGhost
    for j=1:3
        matPoints(i,j)=matData(i+6,j);
    end
end
matValues=zeros(intTotalCellswithGhost,1);
intStartCell=6+intTotalCellswithGhost+3;
for i=1:intTotalCellswithGhost
    matValues(i,1)=matData(intStartCell+i,1);
end

% more setup if on first processor
if p==0

    % find domain dimensions
    xDomain=intXgriddim*((matPoints(1,1)*2)/proccube);
    yDomain=intYgriddim*((matPoints(1,2)*2)/proccube);
    zDomain=intZgriddim*((matPoints(1,3)*2)/proccube);
end
% sort out ghost values
intNumInVOF=(intXgriddim/proccube)*(intYgriddim/proccube)*(intZgriddim/proccube);
matCorrectVOF=zeros(intNumInVOF,1);
xzero=(p-mod(p,proccube^2))/proccube^2;
yzero=floor(mod(p,proccube^2)/proccube);
zzero=mod(p,proccube);
j=1;

for i=1:intTotalCellswithGhost
    xOK=0;
yOK=0;
zOK=0;
    if and(matPoints(i,1)>xzero*xDomain,matPoints(i,1)<(xzero+1)*xDomain)
        xOK=1;
    end
    if and(matPoints(i,2)>yzero*yDomain,matPoints(i,2)<(yzero+1)*yDomain)
        yOK=1;
    end
    if and(matPoints(i,3)>zzero*zDomain,matPoints(i,3)<(zzero+1)*zDomain)
        zOK=1;
    end
    A=[xOK,yOK,zOK];
    if all(A)
        matCorrectVOF(j)=matValues(i,1);
j=j+1;
    end
end

% create vof array for current processor
matprocVOF=zeros(intZgriddim/proccube,intXgriddim/proccube,intYgriddim/proccube);
l=1;

for c=intZgriddim/proccube:-1:1
    for b=1:intYgriddim/proccube
        for a=1:intXgriddim/proccube
            matprocVOF(c,a,b)=matCorrectVOF(l);
l=l+1;
        end
    end
end
% stick the processor VOF array in the right spot in the overall VOF array
zzero=proccube-mod(p,proccube);
yzero=floor(mod(p,proccube^2)/proccube);
xzero=floor(p/proccube^2);

matVOF(procZsize*(zzero-l)+l:procZsize*zzero,...
    (xzero*procXsize)+1:(xzero+l)*procXsize,...
    (yzero*procYsize)+1:(yzero+l)*procYsize)=matprocVOF;

end

A=matVOF;
end
C.3 GetApex

GetApex is used to track the height of the desired wave.

```matlab
function apexcell = GetApex(vtkarray, xmin, xmax)
%GetApex
% Returns a 3x3 matrix of the location and VOF value of
% the apex and it's two nearest neighbors.
% Jake Dynes August, 2014

[xdim,ydim,zdim]=size(vtkarray);
A=vtkarray(:,xmin:xmax,1);

for i=1:xdim
  if sum(A(i,:))>0
    break
  end
end

waverow=A(i,:);
B=waverow<1 & waverow>0;
wavevof=max(waverow(B));
waveboxlength=find(waverow==wavevof);
vofleft=waverow(waveboxlength-1);
vofright=waverow(waveboxlength+1);

lefttotal=[waveboxlength+xmin-2,xdim-i+1,vofleft];
centertotal=[waveboxlength+xmin-1,xdim-i+1,wavevof];
righttotal=[waveboxlength+xmin,xdim-i+1,vofright];

if or(vofleft==0,vofright==0)
  waverowunder=A(i+1,:);
  vofleft=waverowunder(waveboxlength-1);
  vofright=waverowunder(waveboxlength+1);
  if or(vofleft==1,vofright==1)
    'error!!!'
  end
  lefttotal=[waveboxlength+xmin-2,xdim-i+2,vofleft];
  centertotal=[waveboxlength+xmin-1,xdim-i+2,1+wavevof];
  righttotal=[waveboxlength+xmin,xdim-i+2,vofright];
end

apexcell=[lefttotal;centertotal;righttotal];
end
```
C.4 ParaApex

The ParaApex function is intended to use the output from GetApex and fit a parabola to it in order to more accurately measure the apex.

```
function apexcell = ParaApex(apexmat,h,k)

leftxbox=apexmat(1,1);
leftzbox=apexmat(1,2);
v1=apexmat(1,3);
centerxbox=apexmat(2,1);
centerzbox=apexmat(2,2);
v2=apexmat(2,3);
rightxbox=apexmat(3,1);
rightzbox=apexmat(3,2);
v3=apexmat(3,3);

x1=(leftxbox-1)*h;
x2=leftxbox*h;
x3=(leftxbox+1)*h;
x4=(leftxbox+2)*h;

apexloc=((h*(2*v1-3*v2+v3))/(v1-2*v2+v3))+x1;
apexheight=-(k*(v1^2+13*v2^2-7*v2*v3+v3^2-v1*(7*v2+v3))/(6*(v1-2*v2+v3)))+((centerzbox) *
% or centerzbox-1 here?

apexcell=[apexloc,apexheight];

end
```
C.5 PARIS Debugging Tool

% Snippet of code used to output whatever the desired parameter is.
% If file already exists, it will append to it, so make sure to delete
% previos log.
if (rank==0) then
  inquire(file="jaketest.txt", exist=exist)
if (exist) then
  open(17, file="jaketest.txt", status="old", position="append",
       action="write")
else
  open(17, file="jaketest.txt", status="new", action="write")
end if
write(17, *) "vmax:" , vmax, "h:" , h , "deltaT:" , deltaT,
       "inbox_cfl:" , inbox_cfl , "get_cfl_and_check:" , get_cfl_and_check
close(17)
end if
Bibliography


