Physics and Learning Based Computational Models for Breaking Bow Waves Based on New Boundary Immersion Approaches

by

Gabriel David Weymouth

B.S. Naval Arch. and Marine Eng., Webb Institute (2001)
M.S. Mechanical Eng., University of Iowa (2003)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Science in Ocean Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
[June 2008]
May 2008

Author .....
Department of Mechanical Engineering
May 13, 2008

Certified by .....
Dick K.P. Yue
Phillip J. Solondz Professor of Engineering,
Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by .....
Lallit Anand
Chairman, Department Committee on Graduate Students

Physics and Learning Based Computational Models for Breaking Bow Waves Based on New Boundary Immersion Approaches

by

Gabriel David Weymouth

Submitted to the Department of Mechanical Engineering
on May 13, 2008, in partial fulfillment of the
requirements for the degree of
Doctor of Science in Ocean Engineering

Abstract

A ship moving on the free surface produces energetic breaking bow waves which generate spray and air entrainment. Present experimental, analytic, and numerical studies of this problem are costly, inaccurate and not robust. This thesis presents new cost-effective and accurate computational tools for the design and analysis of such ocean systems through a combination of physics-based and learning-based models.

Methods which immerse physical boundaries on Cartesian background grids can model complex topologies and are well suited to study breaking bow waves. However, current methods such as Volume of Fluid and Immersed Boundary methods have numerical and modeling limitations. This thesis advances the state of the art in Cartesian-grid methods through development of a new conservative Volume-of-Fluid algorithm and the Boundary Data Immersion Method, a new approach to the formulation and implementation of immersed bodies. The new methods are simple, robust and shown to out perform existing approaches for a wide range of canonical test problems relevant to ship wave flows.

The new approach is used to study breaking bow waves through 2D+T and 3D simulations. The 2D+T computations compare well with experiments and breaking bow wave metrics are shown to be highly sensitive to the ship geometry. 2D+T breaking bow wave predictions are compared quantitatively to 3D computations and shown to be accurate only for certain flow features and very slender high speed vessels.

Finally the thesis formalizes the study and development of physics-based learning models (PBLM) for complex engineering systems. A new generalized PBLM architecture is developed based on combining fast simple physics-based models with available high-fidelity data. Models are developed and trained to accurately predict the wave field and breaking bow waves of a ship orders of magnitude faster than standard methods. Built on the new boundary immersion approaches, these computational tools are sufficiently cost-effective and robust for use in practical design and analysis.

Thesis Supervisor: Dick K.P. Yue
Title: Phillip J. Solondz Professor of Engineering,
Professor of Mechanical and Ocean Engineering

Acknowledgments

I would like to acknowledge the help of Dick Yue and my thesis committee advisors, of Kelli Hendrickson and the rest of the Vortical Flow Research Lab, and of Claudio Cairoli, Jason Dahl, the students in my "quals class" all of whom provided a willing ear and helpful suggestions.

I would also like to thank my friends family and Becca for their moral support and infinite patience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The three dimensional breaking waves associated with high-speed ships are a classic example of the highly complex flows found in typical ocean engineering settings. Such breaking waves flows are nonlinear and turbulent and include significant air entrainment and spray generation. The breaking process is crucial to the near bow flow which results in slamming and impact loads and bubbly wakes which can persist for kilometers behind the vessel. These physics must therefore be included in the design process of high-performance vessels. Yet experimental, analytic, and numerical approaches have not resulted in general design tool for highly complex breaking ship waves.

Design tools used must be accurate, applicable to general ocean engineering systems, and sufficiently fast and robust to enable hundreds or thousands of predictions to be made during the design process. Analytic studies of breaking bows waves such as [40] simplify the general problem by restricting the ship geometry, flow conditions, and admissible fluid motions. Such studies are useful for determining basic scaling laws but their inaccuracy and limited scope restricts their application to the very initial stages of design and analysis of ocean vessels. Experimental studies of breaking bow waves such as [27] suffer from the opposite problem; general geometries can be accurately modeled but the time and expense for each observations is very large. As such scientific experiments are performed on general hulls (like the large towed wedge referenced above), while ship specific tests are viable only after the design is

nearly complete. Additionally, obtaining accurate measurements of flow quantities in a breaking wave is difficult and often impossible.

Due to these problems, computational models are most often used in the analysis of breaking ship waves, but these tools have their own limitations. Nonlinear potential flow methods are suitable for modeling steep waves up to the point of breaking ([32] and [8]) but potential flow has a limited ability to capture the spray, bubbles, and highly rotational flows associated with the breaking process. Volume-based computational methods such as finite element and finite volume [42] are capable of solving the governing equations for rotational flows. However, the free surface topology for a general breaking wave is nonlinear and multiply connected. The topology of the ocean vessel is also typically complex, such as a ship with appendages and propellers. As such, methods which rely on grids whose boundaries align with the free surface and other boundaries of the fluid domain [39] are not suited for breaking bow wave analysis.

Cartesian-Grid (CG) approaches have the capability to simulate complex engineering flows such as breaking bow waves and are a topic of current research [64]. [6]. These physics-based methods solve the governing partial differential equations by immersing the effects of the free boundaries (such as the air-water interface) and active boundaries (such as a solid ship hull) onto a stationary background Cartesian-grid. The resulting solution methods are robust and orders of magnitude less complex than boundary-fitted methods [49]. This allows the fluid dynamics associated with complex surface topologies in breaking bow waves to be simulated with relative ease and low computational cost.

CG tools allow for detailed investigation of breaking bow wave flows. However, even these models become impractical when forced to simulate a full-scale breaking bow flow. The characteristic length scales for a ship hydrodynamic problem typically run over seven orders of magnitude [65]; from small-scale turbulence and contact line dynamics up to transverse waves in the wake of the vessel. This disparity makes direct simulation of full-scale flows impossible, even using modern computational resources. Using Reynolds-Averaged Navier-Stokes (RANS) or Large Eddy Simulation (LES)

models for the very small scale turbulence allows for reasonable simulations to be done, but each computational run still takes from days to weeks [21], limiting their usefulness in a design setting.

One solution to this problem is to adopt simplifying assumptions, such as in slender-body theory. A non-linear application of slender-body theory is 2D+T modeling, in which the 3D ship flow is mapped onto an unsteady 2D wavemaker flow [59]. The 2D+T model generates a nonlinear realization of bow waves, which may be measured experimentally with great fidelity [53] or simulated numerically in a matter of minutes. However, this model is only accurate in the limit of a very slender high-speed vessel and is not currently validated for breaking waves.

Learning-based models offer an alternative to physics-based simulations which are extremely fast; individual design evaluations taking from seconds to minutes [22],[10]. However, such models require example data to learn from and have limited accuracy when interpolating between this data. Regions of the design space for which there are few or no examples have a high risk of prediction errors [63]. Additionally, more complex physical systems require more elaborate learning models which must be trained with correspondingly larger sets of examples [26]. These models are custom-tailored for each learning problem and are not easily extended to new problems and designs.

This thesis develops fast and robust tools for the modeling and analysis of complex flows through a combination of physics-based methods and learning-based models. New comprehensive CG approaches are developed and applied to 3D and 2D+T ship bow waves. Combining the fast 2D+T computations with a simple learning model trained on a sparse set of 3D examples generates cost-effective design tools for ocean engineering systems. This process is summarized in greater detail in the following section.

1.1 Thesis Contributions and Outline

This thesis presents three key contributions on the topic of fast, robust and accurate computational modeling for complex engineering simulations such as breaking bow waves. The first contribution is to the technical advancement of Cartesian-grid approaches. The proposed advances to the immersion of free and active boundaries retain the inherent advantages of Cartesian-grid methods; speed simplicity and generality. The advancements are developed with general large-scale applications in mind and have benefits which allow comprehensive simulation of ship hydrodynamic flows on Cartesian grids.

Chapter 2 presents the contributions to the Volume of Fluid method for free surface flows. Volume of Fluid methods have been used and actively developed since the 1980s. A standing problem is that these methods do not conserve the volume of fluid during the simulation, leading to non-physical free surface results. The chapter demonstrates this problem is especially significant for energetic 3D flows such as breaking bow waves. A new simple method for the transport of immersed free-boundaries on Cartesian-grids is presented which exactly conserves the fluid volume in 2D and 3D flows.

Chapter 3 and 4 introduce the contributions to the immersion of active-surfaces onto Cartesian-grid simulations. Chapter 3 presents a new general approach for this problem though the use of the Boundary Data Immersion Method to construct the meta-equations governing both the solid and fluid behavior. This formulation produces equations which are a superset of those solved in boundary-fitted methods and could be used to derive adjustments to the discrete algebraic equations as in Cut-Cell methods. Instead of altering the discrete form of the equations we present a simple implementation akin to Immersed-Boundary methods which automatically maintains the solver's order of accuracy.

The method is studied in detail on sample fluid systems such as 1D channel flow and 2D potential flow before it is applied to the general case of fluid motion prediction in Chapter 4. In that chapter the general fluid/solid-body meta-equations are derived

and applied to canonical flow problems for stationary and dynamic bodies. The Boundary Data Immersion Method is used to derive meta-equations for a numerical exit boundary enabling simulation of external flows. The compatibility of the new method with free-surface flows is tested and the ability to measure accurate pressure forces is verified.

The second key contribution of the thesis is the quantified analysis of the validity of the nonlinear slender-body 2D+T model. Using the framework and tools developed earlier, chapter 5 investigates this problem in detail. The chapter presents the analytic background and limiting cases in which 2D+T is expected to be valid. 2D+T simulations of the bow flow are compared to recent experimental results to further validate the computational method and investigate the sensitivity of the flow to the ship geometry. The two governing parameters, ship speed and slenderness, are systematically varied over a set of 3D simulations and their effect on the performance of the 2D+T model is quantified. Limitations of the 2D+T model to the prediction of realistic breaking bow wave flows are discussed and supported with multiple examples.

Finally, the thesis contributes to the study and development of physics-based and learning-based models for physical systems. In chapter 6 an introduction to this theory is presented to further the goal of generalized classification and characterization of physical system models. In that chapter a number of metrics are introduced to characterize physical models which have been in use in information theory for many years [54]. The application of these metrics motivates the use of Cartesian-grid methods in a new and powerful way.

In chapter 7 the use of both physical-models and learning-models in the simulation of ocean systems is explored. Such physics-based learning models are common in the literature and after developing an advanced analytic framework both classical and modern examples are examined. A new type of physics-based learning model based on an intermediate mapping to a physics-based basis function is developed and tested on breaking bow wave flows. Predictions of the ship wave field are made using a fast potential flow code and limited experimental examples. Finally, predictions of

breaking bow waves are made using a sparse 3D set of examples and the CG 2D+T model.

1.2 Mathematical Description of the Fluid System

This section details the mathematical description of the general ocean engineering system. This is the basic description used through out the remainder of the thesis. The primary physical system of interest in this thesis is the general fluid flow near an air-water interface. It is required that the fluid velocity $\vec{u}$ satisfy the conservation of momentum for an incompressible inviscid fluid, given by the Euler equation as

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \vec{\nabla})\vec{u} = -\frac{1}{\rho}\vec{\nabla}p - \vec{g} \tag{1.1}$$

where p is the total pressure, $\vec{g}$ is the gravitational acceleration vector, and ρ is the local fluid density. For convenience, the vector $\vec{r}$ is defined as the combination of the convective and gravitational terms, such that 1.1 becomes

$$\frac{\partial \vec{u}}{\partial t} = -\frac{1}{\rho}\vec{\nabla}p + \vec{r}. \tag{1.2}$$

The effects of surface tension are ignored for the large-scale flows studied in this work. Note that the addition of viscous terms, Large Eddy Simulation Sub-Grid Scale (LES-SGS) models, and other modeling factors can be added to this right-hand-side vector $\vec{r}$ without altering the essence of the computational methods described in this thesis.

Taking the divergence of (1.2) results in a variable coefficient Poisson equation for the pressure of the form

$$\vec{\nabla} \cdot \left(\frac{1}{\rho}\vec{\nabla}p\right) = \vec{\nabla} \cdot \left(\vec{r} - \frac{\partial \vec{u}}{\partial t}\right). \tag{1.3}$$

The pressure field resulting from the solution of this equation is used to project the

velocity field onto one satisfying the divergence free constraint

$$\vec{\nabla} \cdot \vec{u} = 0. \tag{1.4}$$

Our basic implementation of these equations follows the method of the Numerical Flow Analysis (NFA) code [7]. The discrete forms of equations (1.2) and (1.3) are posed on a Cartesian grid covering the fluid domain. Staggered variable placement is used. The time derivatives are treated with an explicit low storage second-order Runge-Kutta method [7]. The pressure terms are treated conservatively using central differences and a preconditioned conjugate-gradient method is used to iteratively solve the Poisson equation. The convective terms are treated with a slope-limited QUICK scheme [31] for stability and accuracy.

The NFA code is well documented as providing excellent high fidelity turnkey solutions to ship flows. However, there are still a number of challenging issues which must be overcome for Cartesian-grid methods to optimally simulate complex flows such as breaking bow waves.

Chapter 2

Advancements to the Treatment of Free Boundaries in Cartesian-grid Methods

Free boundaries are those which are passively transported in the flow, such as the air-water interface. Volume-of-Fluid (VOF) methods allow topologically complex free surfaces to be treated generally by defining a volume-based color-function over a background grid instead of tracking the interface location explicitly. This background field defines the local physical properties of the fluid in the momentum equation, but because the field is discontinuous it is difficult to transport conservatively. This chapter presents a new, simple, and fully-conservative VOF method. The method is validated using canonical tests as well as a corner flow test which is demonstrated to be more relevant to general flows. The method is coupled with the flow solver and tested for linear and breaking waves.

2.1 Introduction

The free-boundary in multi-phase flows (systems with two or more fluid materials i.e. those with a liquid/gas interface) presents a serious mathematical modeling challenge. By definition, the position and forcing on a free boundary is unknown *a priori* and

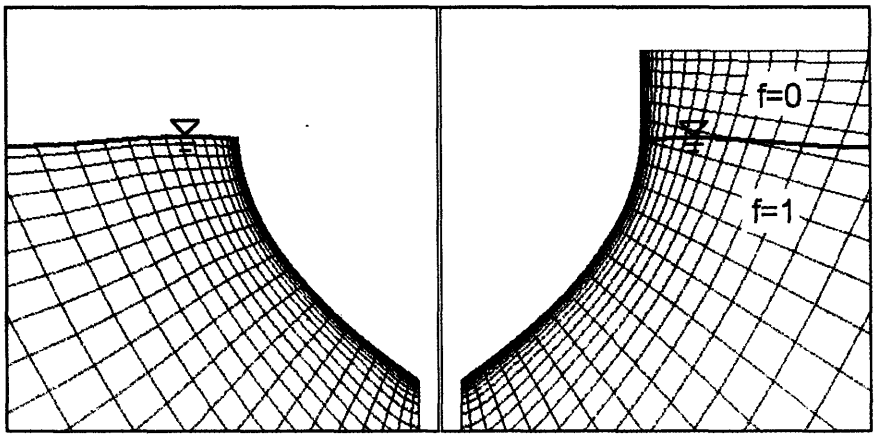


Figure 2-1: Surface capturing versus surface tracking. On the left, a boundary-conforming grid is used to 'track' the free surface interface (highlighted in blue). On the right, the same interface is 'captured' on a background grid whose boundaries do not line up with the free-boundary.

this greatly complicates the solution of the boundary value problem given in section 1.2. In the general case of energetic and topologically complex free-boundaries such as breaking waves, methods which rely on linearization or surface tracking (such as [39]) have limited ability to model such systems.

The state-of-the-art for solving highly nonlinear multi-phase flows are the so-called 'surface capturing' methods, in which a characteristic function defining the location of the free surface is evolved in time. Figure 2-1 illustrates the difference between the surface capturing and tracking. On the left, the surface tracking method relies on a grid whose boundaries conform to the solid and fluid boundary interfaces. As the free-surface moves, the grid is deformed, setting limits on achievable topographies. On the right, the capturing method defines its function on a background grid which need not be boundary conforming. Because the grid does not follow the material interface the numerical problems of grid skew and connectivity are avoided.

There are more than one class of free-boundary capturing methods, notably the level-set method and the volume-of-fluid method. A second-order VOF method is chosen in this thesis, variations of which have been used in CFD simulations for more than 25 years [25]. However, level-set methods are a well established and vibrant research topic. A recent example [21] features direct numerical simulations of small scale turbulent breaking waves are performed using the level-set method.

The VOF technique is based on a color-function $\bar{f}(\vec{x})$ which is defined as

$$\bar{f}(\vec{x}) = \begin{cases} 1 \text{ if there is 'dark' fluid at point } \vec{x} \\ 0 \text{ if there is 'light' fluid at point } \vec{x} \end{cases} \tag{2.1}$$

In the physical simulations to follow the dark fluid is water and the light fluid is air but any appropriate two-phase flow (oil and water, etc) can be simulated using the VOF method. Figure 2-1 shows the value of $\bar{f}(\vec{x})$ in the lower and upper fluid domains for the capturing method. The integral version of the color-function f is defined as

$$f = \frac{\int_\Omega \bar{f}(\vec{x})\, \mathrm{d}v}{\Delta\Omega} \tag{2.2}$$

where $\Delta\Omega \equiv \int_\Omega \mathrm{d}v$ is the volume of region Ω. The field f is appropriately termed the volume fraction as it equals the fraction of volume taken up by the dark fluid. A set of bounds on f

$$0 \leq f \leq 1 \tag{2.3}$$

follow immediately and state that region Ω cannot be less than empty or more than full. In this thesis f is the discrete finite volume form of the color-function defined on each cell of the background grid.

Once the color-function is defined it is simple to construct the fluid property fields, ie

$$\rho(\vec{x}) = \rho_w \bar{f}(\vec{x}) + \rho_a[1 - \bar{f}(\vec{x})] \tag{2.4}$$

where the (assumed constant) density of water ρ_w and air ρ_a have been used with the color-function to define the global field $\rho(\vec{x})$. Using this field in the momentum equation allows one velocity field to describe the complete flow over both the air and water and explicit definition of the interface location is not required. This is what enables flows with arbitrarily complex topology to be simulated. An additional benefit of VOF is that conservative treatment of f conserves the volume of each fluid in the system by definition. This important property is lost when using level-set capturing methods.

However, the field $\bar{f}$ and fields based off of it, such as ρ, are discontinuous across the interface. This is important because the transport theorem used to derive the momentum equations is not applicable to discontinuous flows. The discontinuous form of the transport theorem results in the addition of jump conditions across the interface which are difficult to implement numerically. A related problem is that typical numerical schemes used to enforce conservation of mass and momentum require the resulting fields to be piecewise continuous.

It is common practice to ignore or otherwise sidestep this incompatibility of VOF with the typical formulation of the BVP. In this work we take the simple approach of [6] and smooth the discontinuity in the fluid properties over a few points using a triangular filter. The resulting fields do not capture the true jump at the free-boundary but both [6] and [21] show that good comparison with experimental results are still possible using a smoothed free-interface.

The second issue is with the transport of $\bar{f}$ itself. Unlike the fluid properties, the color-function cannot be smoothed, as discussed in [47]. Therefore the gradients in that field are not defined and the fluxes must be computed directly. This issue, along with condition 2.3 make proper transport of the volume fraction nontrivial. In fact, there are no conservative VOF transport methods for three dimensional flows in the literature. Volume conservation is a zeroth order condition (equivalent to mass conservation in the flow solver) and its violation results in many unphysical characteristics. Examples are spurious flotsam and jetsam (air/water mixing), stair-stepping of the interface along the background grid, and high frequency fluctuations in interface pressure.

Despite the long history of the method, it is clear that a conservative version of VOF is overdue. In the next section the details of VOF transport are presented, including the derivation of a new simple and fully conservative technique.

2.2 Volume Fraction Transport Equation

The transport equation for the color-function is based on the conservation of fluid type for any material fluid point. Therefore the color-function conservation equation is simply

$$\frac{D\bar{f}}{Dt} = \frac{\partial \bar{f}}{\partial t} + \vec{u} \cdot \vec{\nabla} \bar{f} = 0. \tag{2.5}$$

As stated in the previous section, the gradients of $\bar{f}$ are not defined across the interface (and are zero elsewhere) but such gradients can be avoided by using an integral form of the transport equation. This is derived by integrating equation 2.5 over each cell, integrating the second term by parts and using the divergence theorem to obtain

$$\frac{\partial}{\partial t} \int_\Omega \bar{f} \, \mathrm{d}v + \oint_{\partial\Omega} \bar{f} u_n \, \mathrm{d}s = \oint_\Omega \bar{f} \vec{\nabla} \cdot \vec{u} \, \mathrm{d}v. \tag{2.6}$$

which is a transport equation for the volume fraction f, written compactly as

$$\frac{\partial f}{\partial t} \Delta\Omega - F_{net} = \oint_\Omega \bar{f} \vec{\nabla} \cdot \vec{u} \, \mathrm{d}v \tag{2.7}$$

where F_{net} is the net dark fluid flux *into* the cell and the right hand side is the dilatation of dark volume. This dilatation is equal to zero in an incompressible flow.

The volume fraction is updated in time by solving transport equation 2.7 in two steps. First is the reconstruction step, in which the explicit interface location is locally approximated from the volume fraction field. Second is the advection step, in which the fluxes are computed and the equation is integrated in time.

2.2.1 Reconstruction Method

In order to accurately compute the fluxes in equation 2.7 the the free-interface must be locally reconstructed. A typical example is sketched in figure 2-2. On the left is the 'true' free-interface location, however the only information available is the value of the volume fraction f for each cell.

First-order assumptions are used in the original VOF methods which simplified

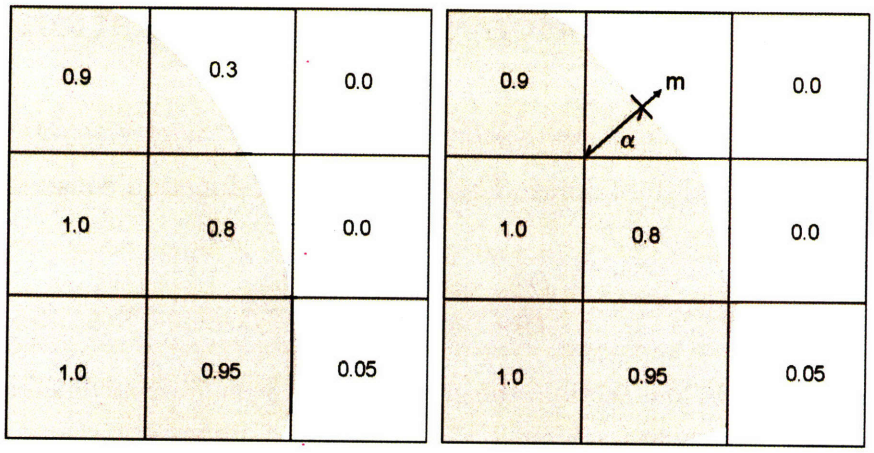


Figure 2-2: 2D diagram of a typical 3x3 block of cells. On the left is the exact interface and on the right is the linear reconstruction which maintains f.

the reconstruction and flux calculation process, but they result in extremely poor performance [45]. Instead a local linear reconstruction is required

$$\vec{m} \cdot \vec{x} = \alpha \tag{2.8}$$

where $\vec{m}$ is the surface normal vector and α is the intercept. Because the descriptions are local the surfaces do not generally match on the boundaries as shown in the figure. Determining the value of α and $\vec{m}$ from f_i for each cell on the interface ($0 < f < 1$) is the reconstruction problem.

Technical note [50] presents a set of analytic equations relating the intercept with the volume fraction once the normal is known. However, determining $\vec{m}$ is not trivial and in [45] Pilliod and Puckett show that poor estimation can limit a linear reconstruction to first-order convergence. That reference suggests two second-order methods for determining the normal, the so called LVIRA and ELVIRA methods. Both are based on generating a linear interface which minimizes the volume fraction error in the surrounding cells (3x3 in 2D and 5x5x5 in 3D) and has zero error in the central cell. That reference shows that both methods can reconstruct any linear interface to arbitrary precision. They also show that exact linear reconstruction results in $O(h^2)$ convergence of the method, where h is the grid spacing.

More specifically, LVIRA (Least-square Volume-of-fluid Interface Reconstruction

Algorithm) minimizes the squared error of a local 3x3 block of cells as a function of m under the constraint that the interface exactly reproduces the volume fraction in the center cell. This is done iteratively using a nonlinear minimization technique, Brent's algorithm. As such it involves an unknown number of inversions for the volume fraction in each of the cells in the local block. ELVIRA (Efficient Least-square Volume-of-fluid Interface Reconstruction Algorithm) is developed to reduce this computational cost. In ELVIRA m is chosen from a finite set of options (6 in 2D) to minimize the error in the local blocks of cells. These options are given by the forward, backward and central differences of the volume fraction in each direction. For instance, denoting $f_{i,j}$ as the central value of volume fraction we have

$$
\begin{aligned}
m_f^x &= \sum_{l=-1}^{1} f_{i+1,l} - f_{i,l} \\
m_c^x &= \frac{1}{2} \sum_{l=-1}^{1} f_{i+1,l} - f_{i-1,l} \\
m_b^x &= \sum_{l=-1}^{1} f_{i,l} - f_{i-1,l} \\
m_f^y &= \sum_{l=-1}^{1} f_{l,j+1} - f_{l,j} \\
m_c^y &= \frac{1}{2} \sum_{l=-1}^{1} f_{l,j+1} - f_{l,j-1} \\
m_b^y &= \sum_{l=-1}^{1} f_{l,j} - f_{l,j-1}
\end{aligned}
$$

where we note that the local cells are summed in the y-direction when taking differences in the x-direction and visa versa. Puckett shows that at least one of these differences reconstructs a linear interface exactly. However, extensions to 3D or nonuniform grids requires understanding how these odd finite differences are related to the interface normal.

The interface equation is the key to understanding the method. As the length of

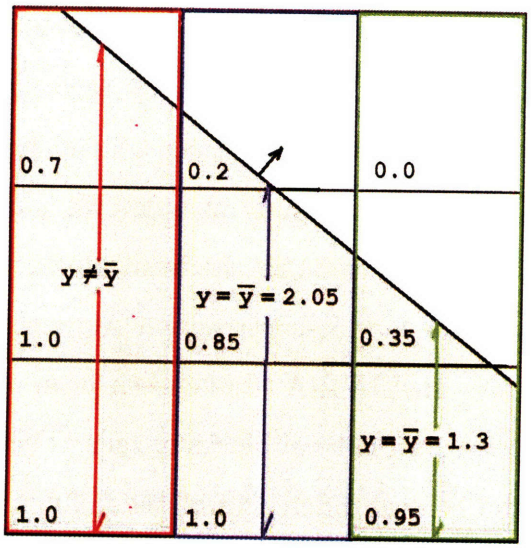


Figure 2-3: Illustration of the mean value of the interface height $\bar{y} = y(x_{i,j})$. When the interface does not cross the top or bottom of the cell the volume fraction f is related to $\bar{y}$. By concatenating cells vertically, the middle cell (blue) and right cell (green) meet this requirement.

m is arbitrary, equation 2.8 for 2D can be written as

$$y = \alpha - mx \tag{2.9}$$

where the y-component of the normal has been set to 1. If values of y are known the normal could be easily established as

$$m = -\frac{\partial y}{\partial x}. \tag{2.10}$$

In fact, ELVIRA is utilizing the mean value theorem to estimate y from the only information available, the finite volume f. Define the mean value of the interface as the integral of that function divided by the integration width,

$$\bar{y} = \frac{\int_a^b y(x)dx}{\int_a^b dx} \tag{2.11}$$

which, for a linear interface simplifies $\bar{y} = y(\frac{a+b}{2})$. Figure 2-3 shows that if the interface passes through opposite sides of the grid then equation 2.11 may be related

to the volume fraction by

$$\bar{y} = \frac{\int_\Omega \bar{f} dv}{\int_{\partial\Omega} dx} = f\Delta y \tag{2.12}$$

such that $y_{i,j} = f_{i,j}\Delta y$ if the interface does not pass through the upper or lower boundary of the cell. The summations in ELVIRA increase the chances of this by tripling the height of the cell, in figure 2-3 the cells are summed vertically resulting in the red green and blue cells. As shown in the figure this ensures that at least two of the 6 possible extended cells accurately estimate the interface position. Setting $\Delta x = \Delta y$ and taking finite differences results in the ELVIRA formulae above, at least one of which reproduce the normal exactly.

However, now that we have a greater understanding we do not need to limit ourselves to 2D uniform grids or inverting to find which difference formula to use. First, use the central differences of f given above to estimate m^x, m^y (and m^z in 3D). This does not generally lead to the correct value of m but it does give us an estimate of the interface orientation. Define the major axis as the one with the largest component of $\vec{m}$. For discussion, assume this is the z axis. Estimate the interface height by summing in that direction, i.e.

$$z_{i,j,k} = \sum_{l=-1}^{1} f_{i,j,k+l}\Delta z_{i,j,k+l} \tag{2.13}$$

As shown in the figure, the central value (blue) is always valid and if the central cell is more than half full the cell 'downwind' of the normal (green in the figure) is also valid. If the cell is less than half full the upwind cell is valid. (Imagine flipping the dark and light fluid in the figure along with the normal vector; the green cell is then upwind.) Thus we compute

$$\vec{m} = \frac{\partial z}{\partial \vec{x}} \tag{2.14}$$

using whichever finite difference method is appropriate. This direct ELVIRA-type method allows exact reconstruction of a linear interface in 2D or 3D using only the local 3x3(x3) block of cells and no inversions. Because it reproduces linear interface exactly the method is expected to be second-order, and this is verified in the following

sections. It is used for the remainder of the thesis.

2.2.2 Advection Method

Once the reconstruction step is complete (using ELVIRA or any other second-order method) the next step is to compute the fluxes in equation 2.7 and integrate the equation in time. Sources such as [45] and [46] presents many suggested unsplit advection algorithms to solve equation 2.7 for two-dimensional flows based on geometric flux integrations along approximated characteristics of the velocity field $\vec{u}$. However, such multidimensional flux integrations are algorithmically complex, particularly for three-dimensional flows. In addition, such methods often violate conservation drastically [20] due to overshooting condition 2.3.

The standard simplification is to use an operator-split advection method. As the name suggests, the transport equation 2.7 is split into sequential updates of the volume fraction in each of the $\mathcal{N}$ spatial dimensions

$$\frac{\partial f'_{ijk}}{\partial t}\Delta\Omega = F_{i+1/2} - F_{i-1/2} + \int_\Omega \bar{f}\frac{\partial u}{\partial x}\,\mathrm{d}v \tag{2.15}$$

$$\frac{\partial f''_{ijk}}{\partial t}\Delta\Omega = G_{j+1/2} - G_{j-1/2} + \int_\Omega \bar{f}\frac{\partial v}{\partial y}\,\mathrm{d}v \tag{2.16}$$

$$\frac{\partial f'''_{ijk}}{\partial t}\Delta\Omega = H_{k+1/2} - H_{k-1/2} + \int_\Omega \bar{f}\frac{\partial w}{\partial z}\,\mathrm{d}v \tag{2.17}$$

where G and H are the fluxed dark fluid in the second and third directions. In other words, the fluid is first transported along x, then along y and then (for 3D flows) along z. With a slight abuse of notation we summarize the above as

$$\frac{\partial f}{\partial t}\Delta\Omega = \Delta_d F_d + \int_\Omega \bar{f}\frac{\partial u_d}{\partial x_d}\,\mathrm{d}v \qquad \text{for } d = 1\ldots\mathcal{N} \tag{2.18}$$

where d is the Cartesian index and $\Delta_d F_d$ is the net flux in that direction. By treating only one velocity component at a time, the calculation of the fluxed dark volume can be analytically determined using the relations in [50]. A typical two-dimensional reconstruction is shown in Figure 2-4. In this figure the velocities are scaled by $\Delta t/\Delta x$

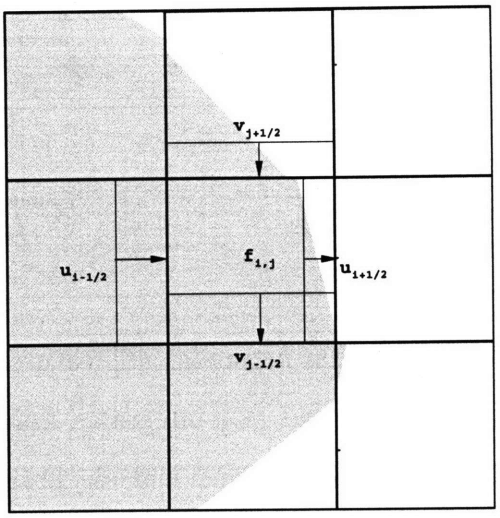


Figure 2-4: 2D diagram of a typical linear VOF surface reconstruction on a 3x3 block of cells with scaled velocity components. Using the standard operator-split treatment, the donating regions are right-cylinders upwind of each face.

making them local Courant numbers. With this scaling a donating region upwind of each face can be defined with width equal to the velocity magnitude. The dark fluid within each region is fluxed into the next cell. In figure 2-4, the fluid in the bottom-right corner of cell (i, j) is in two donating regions. To avoid the possibility of fluxing the same fluid into two different cells the surface must be reconstructed after each sweep of an operator-split method.

The dilatation term in equation 2.18 is required because at each step of an operator-split algorithm the volume fraction is advected in a one-dimensional flow *which is not divergence free.* Without accounting for the dilatation of dark volume there is no way to ensure that condition (2.3) is met after each step of the algorithm. For example, in figure 2-4 more fluid is being fluxed in from cell $(i-1, j)$ than space left after fluxing out to cell $(i+1, j)$, meaning that cell (i, j) overfills in the first sweep. A recent paper [51] investigates this effect in detail and develops a two-dimensional operator-split Eulerian implicit Lagrangian explicit (EI-LE) method which conserves the volume fraction exactly. In that method, (2.18) is integrated implicitly for $d = 1$ and explicitly with Lagrangian flux treatment for $d = 2$. Using the same velocity scaling as above, the discrete form of the implicit and explicit integrations are given

by

$$f_{ij}^{n+1/2} = \frac{f_{ij}^{n} - \left(F_{i+1/2}^{n} - F_{i-1/2}^{n}\right)}{\left[1 - u_{i+1/2}^{n} - u_{i-1/2}^{n}\right]} \tag{2.19}$$

$$f_{ij}^{n+1} = f_{ij}^{n+1/2}\left[1 + v_{j+1/2}^{n} - v_{j-1/2}^{n}\right] - \left(G_{j+1/2}^{n+1/2} - G_{j-1/2}^{n+1/2}\right) \tag{2.20}$$

where n is the time step counter.

Earlier reference propose similar methods; [46] features the same splitting without the Lagrangian flux calculation and [47] integrates first explicitly, then implicitly. However, [2] demonstrates by two-dimensional algebraic mapping that the EI-LE scheme is the only one of the three that is volume conserving. Unfortunately, this is dependent upon a two-dimensional divergence-free velocity field and the mathematical analysis in that work does not naturally extend to 3D.

Breaking the problem of conservation into a short list of requirements clarifies the issues. Given that

1. The flux terms are conservative, *and*

2. The divergence term sums to zero *and*

3. No clipping or filling of a cell is needed due to violation of (2.3) at any stage

then the algorithm *must* conserve f to machine precision. The first requirement ensures that any fluid going into one cell is coming out of another and the second ensures that there is no net source term added to the advection equation. Along with the third requirement, it is guaranteed that there is no net change in the dark fluid volume regardless of the dimensionality of the system. Multiplying the divergence term by f^n in the explicit step and then f^{n+1} in the implicit step, [47] creates a net source term, violating requirement 2. Implicit integration violates requirement 1 by scaling the fluxes by a local stretching term, bracketed in (2.19). This scaling term is not consistent on both sides of the cell face, meaning the effective flux is not conservative. The EI-LE scheme maintains global conservation despite this by using the Lagrangian flux calculations and relying on the bracketed term in (2.19) to be

exactly canceled by the bracketed term in (2.20) due to two-dimensional continuity. This cancelation is not extendable to three dimensions.

However, a simple operator-split method can be designed which meets these requirements for two or three-dimensional flows. Starting from 2.18 the only flexibility we have is the treatment of the dilation term. The term must be treated explicitly but there is the matter of estimating the integral. All the methods above estimate the integral using the volume fraction f, but the cell center value of the old color-function $\bar{f}^n$

$$\bar{f}_c^n = \begin{cases} 1 & \text{if} \quad f^n > 1/2 \\ 0 & \text{else} \end{cases} \tag{2.21}$$

follows from simple geometry and is a better choice. Indeed, an advection method which treats the dilatation in this way

$$\Delta f \frac{\Delta\Omega}{\Delta t} = \Delta_d F_d + \bar{f}_c^n \frac{\partial u_d}{\partial x_d} \Delta\Omega \qquad \text{for } d = 1 \ldots \mathcal{N} \tag{2.22}$$

along with the restriction

$$|u_d| < \frac{1}{2\mathcal{N} - 2} \qquad \text{for all } d \tag{2.23}$$

for cells on the interface is fully conservative for general flows. The key features of this scheme are the use of conservative fluxes and the constant field multiplying the divergence term. Thus requirements 1 and 2 are met, leaving only requirement 3 which is proven in appendix A. Note that in addition to handling 3D flows, the algorithm is also applicable to both uniform and clustered Cartesian-grids. Thus, the additional work put into understanding the true form and role of the dilatation term has resulted in a new simple and general method.

2.2.3 Transport Method Verification

There are a standard set of tests in the literature for the numerical verification of VOF transport algorithms. The general method involves starting with an analytic

interface and transporting the resulting volume fraction in a given constant velocity field. The result is then compared to the known analytic solution. Reference [45] presents a set of such tests for the uniform flow with an arbitrary orientation and rotational flow for shapes such as circles, crosses and notched circles.

This paper is an excellent reference but has a major shortcoming; the sample flows have no stretching. This simplifies the calculation of the error metrics but artificially ensures that the methods conserve mass. Note that all the advection methods above (equations 2.20 2.19 2.22) collapse to the simple update

$$\Delta f \frac{\Delta \Omega}{\Delta t} = \Delta_d F_d \qquad \text{for } d = 1 \ldots \mathcal{N} \tag{2.24}$$

for flows without stretching. Thus the current method performs identical to method of [45] for translation and rotation cases. To verify this (and the second order convergence of the method) the translation and rotation tests for a circle geometry are repeated and the results are shown in table 2.3. The error metric for these tests is simply

$$E = \sum_{i,j,k} |f_{i,j,k} - \tilde{f}_{i,j,k}| \tag{2.25}$$

where $\tilde{f}$ is the exact volume fraction solution. Reference [45] uses a more advanced metric based on analytic integrations of the color-function. While this is necessary for establishing reconstruction errors (as $f = \tilde{f}$ for any valid reconstruction) it is not required when measuring the overall transport error. In the table $h = R/\Delta x$ where R is the circle radius. As table 2.3 shows, the current method is second-order for the translation and rotation cases for CFL numbers $\Delta x/\Delta t = 1/2, 1/32$. Again this is expected because the method is effectively identical to the previous results for flows with no stretching.

To compare the relative mass conservation properties of the methods requires the use of a background flow with stretching such as the corner flow, defined by

$$u = x \qquad ; \qquad v = -y \tag{2.26}$$

Table 2.1: Error of 100 random translations, $\Delta x/\Delta t = 1/2$

h	8	16	32	64
E	1.5324e-2	4.7428e-3	1.5086e-3	4.2818e-4

Table 2.2: Error of 100 random translations, $\Delta x/\Delta t = 1/32$

h	8	16	32
E	1.9208e-2	7.9863e-3	2.6550e-3

Table 2.3: Error of one rotation, $\Delta x/\Delta t = \pi/6$

h	8	16	32	64
E	1.5756e-2	3.7770e-3	9.7135e-4	1.6705e-4

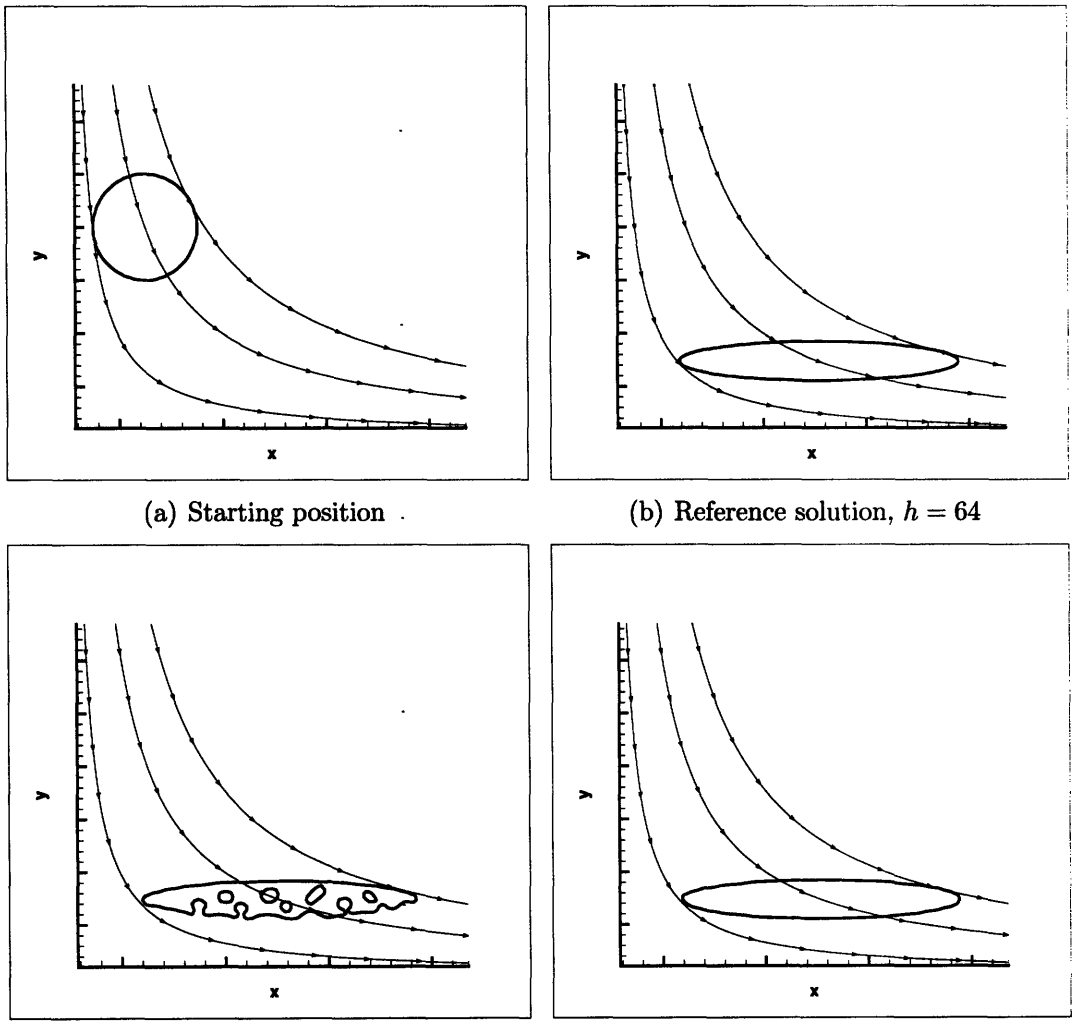
Table 2.4: Error and mass loss of the corner flow test

method	h	L_1	% Change	E
Baseline	8	3.0079e-4	14.85	2.0397e-1
Baseline	16	1.4734e-4	14.15	1.9188e-1
Baseline	32	7.6070e-4	14.62	1.9631e-1
Pilliod 04	8	1.4549e-5	0.084	3.8764e-2
Pilliod 04	16	2.7978e-6	-0.036	1.1548e-2
Pilliod 04	32	5.3006e-7	0.013	2.5441e-3
Current	8	0.	0.	4.0853e-2
Current	16	0.	0.	1.1261e-2
Current	32	0.	0.	2.4661e-3

which has uniform stretching throughout the flow and is divergence free. Figure 2-5 and table 2.4 show the results for three advection algorithms. The first, labeled 'baseline' is the advection method with no stretching term, given by equation 2.24. The second is the 2D split method of Pilliod 04, given by equations 2.20 and 2.19. The third is the current method, given by equation 2.22.

Figure (a) shows the starting position of the circle centered at $(x/R, y/R) = (3/2, 4)$. Figure (b) shows the reference solution, computed using the current method with a fine grid, $h = 64$. Figures (c) and (d) show solutions using the baseline and current method with $h = 32$. Clearly, the baseline method which does not account for the 1D stretching is completely invalid, resulting in a flow full of flotsam.

Table 2.4 further verifies that result. In this table, E is calculated as above and compared to the reference solution. The L_1 metric is the mean dark fluid loss at each



(a) Starting position

(b) Reference solution, $h = 64$

(c) Baseline solution, $h = 32$

(d) Current solution, $h = 32$

Figure 2-5: Contours of $f = 1/2$ and streamlines for the corner flow test case. Figure (a) shows the starting position, and (b) shows the reference solution obtained on a fine grid ($h = 64$). Figure (c) shows the result when 1D stretching is not accounted for. Figure (d) shows the result when stretching is accounted for using the current method with $h = 32$.

step of the transport algorithm. Defining the volume of dark fluid as V we have

$$L_1 = \frac{1}{T}\sum_{t=1}^{T}|V_t - V_{t-1}|. \tag{2.27}$$

The second loss metric is the total percent change of volume after transport

$$Change = \frac{V_0 - V_T}{V_0}. \tag{2.28}$$

The table shows that the loss of volume for the baseline case is around 15%, and not mitigated with increased resolution. The Pilliod 04 method fares much better, but still features volume losses of around 0.01% even on the finest grid. The current method conserves volume exactly for every resolution. The corner flow test thus highlights the need for VOF methods to be tested in flows with stretching. The performance of the second method is probably adequate for the corner flow, however this velocity field is 2D and the stretching is very mild $\frac{\partial u}{\partial x} = 1$. In a more violent flow, such as the flow associated with breaking waves, the conservation problem is more severe. This is demonstrated in the following section.

2.3 Two-phase Flow Solver Verification

In this section the VOF transport method developed and tested in the previous section is incorporated into the general flow solver described in section 1.2, enabling the simulation of general two-phase flows.

As mentioned in section 2.1 the sole adjustment to the momentum and mass conservation equations is the inclusion of spatially variable fluid properties based on the color-function, such as equation 2.4. That section also discusses that smoothing the jump in values across the free-interface enables a standard flow solver to be utilized. Therefore, the density is smoothed using a triangular filter over the 5x5(x5) block of cells surrounding the interface.

Two standing wave tests are presented verify the proper inclusion of the VOF method; a small amplitude linear 2D case and a high amplitude breaking 3D case.

Table 2.5: Error norms for the linear wave test

h	25	50	100	200
E	1.145e-2	4.399e-3	1.115e-3	9.521e-4
E_{ref}	1.162e-2	3.277e-3	5.921e-4	-

2.3.1 2D Linear Wave Test

The first test of the two-phase solver is a simple low amplitude standing wave. The interface elevation η is given by

$$\eta(x,t) = A\cos(kx + \omega t) \tag{2.29}$$

where A is the wave amplitude, k is the wave-number, and ω is the frequency. The dispersion relationship for deep water waves is

$$\omega^2 = kg \tag{2.30}$$

where g is the acceleration of gravity.

This analytic benchmark is used to assess the quality of the numerical method but it is important to note that the correspondence is not exact. Firstly, the VOF method is a general two-phase flow solution method, whereas the analytic solution assumes the effect of the air is negligible and the free surface BCs may be linearized around the mean free surface. Additionally, smoothing the density transition introduces a modeling error into the VOF method. These factors mean the the VOF solution does not converge to the analytic solution exactly, but the difference is very small for flows with high density ratios (such as air and water) and a fine numerical grid.

For the simulations the computational domain chosen is a rectangle with width equal to the wavelength λ and extended above and below the mean free surface by a wavelength. The grid spacing is varied from $h = \lambda/\Delta x = 25, 50, 100, 200$, and $\Delta x/\Delta t = 0.5$ and reflection boundary conditions set on all walls. The analytic solution is used as the initial condition with $A = 0.01\lambda$.

Table 2.5 shows the error E in the volume fraction compared to the analytic

solution. The convergence rate is less than second order because of the modeling errors discussed above, but the error is small and monotonically decreasing with increased resolution. To measure the convergence rate more precisely, we compare the error at each refinement level with the finest solution, labeled E_{ref} in the table. The results are approximately second-order and as such the VOF flow solver is numerically verified and analytically validated.

2.3.2 3D Nonlinear Wave Test

A high amplitude 3D standing air-water wave is used for the next test. Because this wave is nonlinear, there is no benchmark data against which the finial solutions can be compared. However, the convergence properties of the solver can be estimated and compared to the results from existing VOF methods in the literature.

The same Cartesian-grid solver described in the previous section is run on a cubic computational domain, with $\Delta x/\lambda = 100$, $\Delta x/\Delta t = 0.1$ and reflection boundary conditions set on all walls. A sinusoidal free surface with $A = 0.3\lambda$ and an average elevation of $z = 0$ is used as the initial condition. The slope is well above the Stokes limit and results in a highly nonlinear fully three-dimensional wave with overturning as seen in Figure 2-6.

Four VOF transport methods are tested, the first being a baseline test, equation 2.24. The second two are extensions of methods such as that of [47], and [46] and [45] to three-dimensional flows. The first (E-E-E) uses all explicit integrations of the form of equation 2.20 and the second (I-I-E) uses implicit integration of the form of equation 2.19 for the first two steps. These are all compared to the current method defined by equation 2.22 and the results are shown in table 2.6. The first two columns give the L_1 and L_∞ norms of the global volume loss of water in the domain after each use of the advection method. The third column gives the percentage change in total water volume in the domain after one wave period. The baseline gives by far the worst performance, with a net mass loss of 12% after only one wave period. The second and third rows show around an order of magnitude decrease in error compared to neglecting the stretching term altogether but still leave room for improvement. The

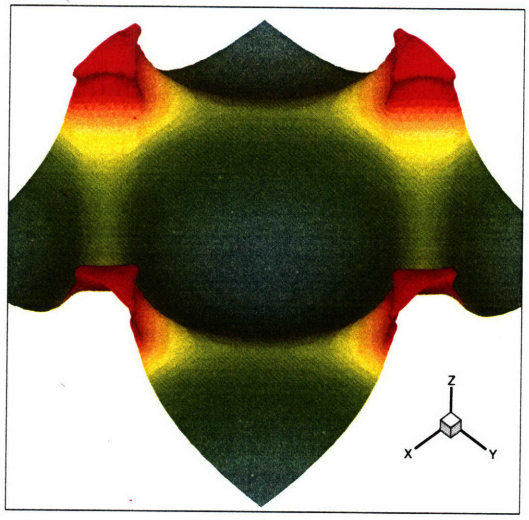


Figure 2-6: Visualization of the free surface for the 3D wave test at $t = 0.5$. Contours indicate elevation. The nonlinear wave has nearly impinged on the top of the computational domain in this image.

Algorithm	L_1 norm	L_∞ norm	% Change
Baseline	2.53e-3	1.40e-2	-12.63%
E-E-E	3.03e-4	2.51e-3	-1.42%
I-I-E	1.22e-4	1.34e-3	+0.81%
Current	1.34e-12	6.83e-12	-0.00%

Table 2.6: Global mass loss measurements for a high slope 3D standing wave test case. The first two columns refer to the volume loss at each step, and the final column gives the total change after one wave period.

current method demonstrates conservation of water-volume after each step and after one period to machine precision.

Note that the results in table 2.6 are around two orders of magnitude worse than the corner flow test. There are two reasons for this decreased performance. First, the violent breaking has more extreme stretching than the corner flow. Instead of $\frac{\partial u}{\partial x} = 1$ the local stretching could be as large as $u_{max}/\Delta x$. For this breaking wave test that value is around 100, two orders larger than the corner flow. The second reason is that the flow is 3D. As detailed in section 2.2.2, the bracketed terms in equations 2.20 and 2.19 cancel in a 2D flow. Lagrangian flux calculation (as in the EI-LE scheme of [2]) is required for the cancelation to be exact but partial cancelation still mitigates the error in 2D. In 3D the additional advection step makes this impossible.

In light of these problems it may seem somewhat surprising that the E-E-E and

I-I-E schemes are an order more conservative than the baseline case. Recall that the baseline case has no stretching term and as discussed in section 2.2.2 this means that the method overfills during advection and requires excess volume to be 'clipped' at each iteration. This accounts for the drastic mass loss in both 2D and 3D tests. On the other hand, while both the E-E-E and I-I-E methods overfill, *improper inclusion of the dilatation term also allows them to overempty.* The process could be crudely modeled as a random walk. By allowing steps to be taken in both directions, the 'average distance' from the starting point is decreased but still grows with the iteration number. The only way to avoid this situation is through exact conservation at each step, as achieved by the current method.

This chapter demonstrated the development, verification, and validation of a new VOF method for 2D and 3D flows. The new method is simple and accurate and exactly conserves the volume of fluid throughout the domain. It is therefore a significant contribution to the state of the art in Cartesian-grid methods for free surface flows, enabling accurate and physically consistent predictions of ocean systems such as breaking waves.

Chapter 3

Advancements to the Treatment of Active Boundaries in Cartesian-grid Methods

Active boundaries are those which actively influence the fluid system in which they are immersed. These are the natural complement to free boundaries and examples range from exit-planes to fixed or prescribed-motion material interfaces. Such boundaries are ubiquitous in ocean systems, and in the case of breaking ship waves they are the driving force behind the fluid dynamics.

In this the immersion of active boundaries and bodies are examined in detail. A new general framework called the boundary data immersion method (BDIM) is developed to formulate equations of motion governing the complete system. Analogous to VOF's role in free-boundary simulations this method allows background-grid methods to accurately simulate complex flows around active boundaries. A form of these governing equations is presented which, unlike the current state-of-art methods, is simple to implement and automatically maintains the order-of-accuracy of the general flow solver.

After deriving a complete set of general equations, two example systems are studied. First a simple 1D channel example which allows explicit demonstration of consistency and order-of-accuracy. Second BDIM is applied to the problem of potential

flow and a spherical immersed geometry.

3.1 Introduction

Treatment of body boundaries are of particular concern in Cartesian-grid methods. Because they do not require complex boundary fitted grids to be created, Cartesian-grid solvers have the potential to generate solutions to more complicated problems orders of magnitude faster than conventional fitted-grid solvers. Yet, this speed and generality must not result in a lack of accuracy in the flow around bodies immersed in the fluid domain. As this section shows, the methods currently available have limitations in applications or accuracy or have complicated implementations.

Thus far, two primary methods exists in the literature to enforce the effects of solid bodies in Cartesian-grid simulations: Immersed-Boundary methods and Cut-Cell methods. The Immersed-Boundary method is developed for use in fluid-structure interaction problems, specifically biological fluid dynamics. [43] gives a detailed review. Generally, the elastic body equations are solved on an explicitly defined surface mesh, while the fluid equations are solved on a Cartesian grid. The two simulations are linked by applying the reaction force of the body on the fluid and advecting the body in the resulting flow. The localized forces and body velocities are calculated using a smoothed approximation of the Dirac delta function. There are many physical systems which are naturally modeled in this way, such as flapping filament in a cross flow [68]. However, in Immersed Boundary methods the body is pseudo-passively advected by the flow restricting applications to flexible bodies and systems that are not mathematically stiff. High-stiffness bodies generate larger reaction forces, leading to instability in simulating rigid bodies.

The other prominent choice is the so called Cut-Cell method. The general process in this family of methods is to alter the discrete form of the fluid equations of motion near the body to account for its presence. There are a great variety of these methods, and the body may either be explicitly defined by a mesh [61] or implicitly defined by a field [62]. The changes to the discrete equations often involve interpolating

boundary values such as in Chimera methods and altering the local grid metrics such as with non-orthogonal grids [34]. Other alterations have also been researched such as locally changing the grid from staggered to collocated [17] and setting up extrapolated "ghost-cells" [58] within the domain. Such methods have also been used model ship flows using level-set [66] and VOF [6] for the free surface. The difficulties with these methods are their complexity, computational expense and maintaining the flow solver's order of accuracy and stability [67].

Dommermuth ([5],[7]) developed an alternate approach to avoid the difficulties of both immersed-boundary and cut-cell methods for applications to rigid no-slip bodies such as ships. Given a body with a no-slip boundary condition, ie

$$\vec{u} = \vec{U} \tag{3.1}$$

where $\vec{U}$ is the body velocity. The method simply adds a body force term proportional to the velocity boundary condition error

$$\vec{f_b} = \alpha(\vec{u} - \vec{U})$$

to the momentum equations 1.2 for all points within the ship body

$$\frac{\partial \vec{u}}{\partial t} = \vec{\nabla} p + \vec{r} + \vec{f_b}.$$

If f_b is the dominate term on the RHS the the system behavior is approximately modeled by

$$\frac{\partial \vec{u}}{\partial t} \approx \alpha(\vec{u} - \vec{U})$$

and the velocity field within the body is driven to the prescribed body velocity $\vec{U}$ exponentially in time. However, this method suffers from fairly serious modeling errors and limitations. First, the boundary condition is treated as a goal-state rather than an instantaneous restriction on admissible velocity fields. Depending on the system and value of α the internal field could react in an underdamped or overdamped

fashion. This is unacceptable for unsteady simulations and could unduely influence the evolved solution for a steady-state calculations. Also, the method is restricted to modeling a 'no-slip' type boundary condition while a free-slip condition

$$\vec{u} \cdot \hat{n} = U_n, \tag{3.2}$$

where $\hat{n}$ is the surface normal, is often more efficient for large scale Cartesian-grid simulations as discussed below. Despite these limitations the method has provided good 3D ship wave field results.

Expanding upon that work, a new method is presented to constrain the flow to the known instantaneous velocity conditions on a general, dynamic body for two and three-dimensional flows. Similar to the body force or immersed-boundary method, the analytic form of the governing equations are altered to account for the body. This is in contrast with Cut-Cell methods which change the discrete form of the equations near the body incurring computational difficulties. The boundary constraints are directly applied to the velocity field unlike the body-force or immersed-boundary method. This avoids goal-seeking and allows the body to drive the flow instead of being advected by it, enabling the simulation of unsteady flows around high-stiffness bodies such as ship hulls.

3.2 Boundary Data Immersion Method

Cartesian-grid free-surface methods use variable density and viscosity fields to avoid explicitly defining the free-surface interface and boundary conditions. This allows the entire flow field to be governed by one general set of equations, greatly increasing the simplicity, speed, and robustness of the solution algorithm. The fundamental goal of a boundary data immersion method is to extend this solution philosophy to flows with solid bodies.

3.2.1 The Meta-Equation Concept

With a transition between two fluids the density and other physical properties may change but the form of the governing equations are consistent. The additional difficulty when a solid is immersed in the fluid domain is that the form of the equations of motion themselves are not uniform. Therefore the boundary data immersion method must develop a single "meta-equation" which describes the dynamics governing the fluid as well as the body immersed within it.

Equations governing the fluid and solid body behavior such as 1.2 and 3.1 can generally be cast in the form

$$\mathcal{F}(\vec{x}_f, t) = 0 \quad \forall \quad \vec{x}_f \in \Omega_f \tag{3.3}$$

$$\mathcal{B}(\vec{x}_b, t) = 0 \quad \forall \quad \vec{x}_b \in \Omega_b \tag{3.4}$$

where $\mathcal{F}$ is the fluid equation defined on points $\vec{x}_f$ in the fluid domain Ω_f and $\mathcal{B}$ is the body equation defined on points $\vec{x}_b$ in the body domain Ω_b. This section does not deal with the time dependance of the functions and so this dependance is omitted below. Time varying systems are considered in chapter 4. The meta equation for the system is simply a mixture of those two systems of the form

$$\mathcal{M}_\epsilon(\vec{x}) = \mathcal{P}_\epsilon\{\mathcal{F}, \vec{x}\} + \frac{1}{c}\mathcal{Q}_\epsilon\{\mathcal{B}, \vec{x}\} = 0 \tag{3.5}$$

where $\mathcal{Q}_\epsilon$ and $\mathcal{P}_\epsilon$ are interpolating functions which are used to generate the single meta equation at the general point $\vec{x}$ in the full domain $\Omega = \Omega_f \bigcup \Omega_b$. The parameter ϵ is the interpolation width and c is a scaling factor to balance the influence of the two equations. Figure 3-1 sketches the desired spacial variation of equation 3.5, which can be summarized as:

1. Inside the solid domain Ω_b we require $\mathcal{P}_\epsilon \to 0$, $\mathcal{Q}_\epsilon \to \mathcal{B}$, and therefore $\mathcal{M}_\epsilon \to \frac{1}{c}\mathcal{B}$.

2. Inside the fluid domain Ω_f we require $\mathcal{P}_\epsilon \to \mathcal{F}$, $\mathcal{Q}_\epsilon \to 0$, and therefore $\mathcal{M}_\epsilon \to \mathcal{F}$.

3. The transition between the equations should be smooth and located within a

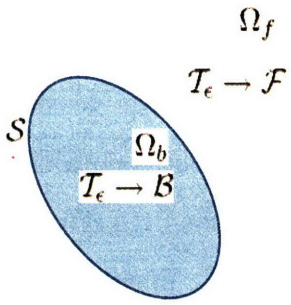


Figure 3-1: Behavior of the total meta-function $\mathcal{M}_\epsilon$ for a general body immersed in the fluid domain.

small distance of the solid/fluid interface $\mathcal{S}$.

At this point it is useful to address possible concerns over the validity of equation 3.5. Using interpolation to transition between two sets of functions is a standard mathematical process. However, the idea of interpolating governing equations may seem slightly more foreign because of the involvement of unknowns (such as the velocity field) and their derivatives. A simple conceptual trick to overcome this problem is to think of the field as a infinite sum of weighted basis functions $\mathbf{B}(x)$, ie

$$\phi(x) = \sum_{i=1}^{\infty} \phi_i B_i(x).$$

Substitution of such a field into the meta-equation allows all of the differential and interpolation operations to be applied to the known basis function set. As such the result is simply a new equation for the unknowns.

It is important to note that the interpolating functions are extending the range of each domain's governing equations by the small amount ϵ into the neighboring domain. As this width goes to zero, the switch from $\mathcal{F}$ to $\mathcal{B}$ becomes a shock across the surface of the immersed body. This is analogous to the transition in fluid properties across the free surface interface discussed in the previous chapter and similar continuity issues arise. When ϵ is too small to be resolved by the numerical grid special methods for calculating fluxes are required near this transition. This leads to the cut-cell methods introduced in the previous section. However, if the width ϵ is small but resolved by the grid then no alterations are required. This allows the governing

Domain Type	Volume	Area	Line	Point
Dimensions	3	2	1	0
α	$\epsilon^3(\frac{2\pi}{3} - \frac{4}{\pi})$	$\epsilon^2(\frac{\pi}{2} - \frac{2}{\pi})$	ϵ	1

Table 3.1: Kernel normalization coefficients for different domain types

meta-equation to be solved independent of the space tessellation. Thus a trade-off exists between computationally complex sharp interface methods and approximate smoothed interface methods.

3.2.2 Interpolation Functions

Before equation 3.5 can be applied to fluid/solid interaction problems the form of the interpolation functions must be determined. One suitable choice of P_ϵ and Q_ϵ with a rich mathematical background is an interpolating kernel, ie

$$\mathcal{Q}_\epsilon\{\mathcal{B}, \vec{x}\} = \int_{\Omega_b} \mathcal{B}(\vec{x}_b) K_\epsilon(\vec{x}, \vec{x}_b)\, \mathrm{d}\vec{x}_b \tag{3.6}$$

$$\mathcal{P}_\epsilon\{\mathcal{F}, \vec{x}\} = \int_{\Omega_f} \mathcal{F}(\vec{x}_f) K_\epsilon(\vec{x}, \vec{x}_f)\, \mathrm{d}\vec{x}_f \tag{3.7}$$

where K_ϵ is an interpolating kernel function of finite support ϵ. Choosing K_ϵ as a nascent delta function such as,

$$K_\epsilon(\vec{x}, \vec{x}') \equiv \begin{cases} \frac{1}{2\alpha}\left[1 + \cos\left(\pi \frac{|\vec{x}-\vec{x}'|}{\epsilon}\right)\right] & \text{for} \quad \frac{|\vec{x}-\vec{x}'|}{\epsilon} < 1 \\ 0 & \text{else} \end{cases} \tag{3.8}$$

ensures the desired regional behavior is met as $\epsilon \to 0$. The normalization coefficient α is set such that the zeroth moment of the kernel for any point $\vec{x}$ fully within the domain equals one. This coefficient is only dependent on the number of spacial dimensions the domain is defined over [1] and table 3.1 lists the results.

The form of equations 3.6 and 3.7 are very general and a system of linear algebra equations could be produced from them using a Galerkin approach, among other

[1]This simple result does not hold if an immersed manifold comes within ϵ distance of folding back on itself. Such surfaces are not explored in this work and so the point is relegated to footnote status.

options. However in many cases a more simplified approach is possible. Instead of integrating the functions over their domains the functions may be simply scaled by the integral of the interpolation functions, ie,

$$\begin{aligned}
\mathcal{Q}_\epsilon\{\mathcal{B},\vec{x}\} &= \mathcal{B}(\vec{x})\int_{\Omega_b} K_\epsilon(\vec{x},\vec{x}_b)\,\mathrm{d}\vec{x}_b \\
&= \delta_\epsilon(\vec{x})\mathcal{B}(\vec{x}) \qquad (3.9)\\
\mathcal{P}_\epsilon\{\mathcal{F},\vec{x}\} &= \mathcal{F}(\vec{x})\int_{\Omega_f} K_\epsilon(\vec{x},\vec{x}_f)\,\mathrm{d}\vec{x}_f \\
&= \mathcal{F}(\vec{x})\left[\int_{\Omega} K_\epsilon(\vec{x},\vec{x}')\,\mathrm{d}\vec{x}' - \int_{\Omega_b} K_\epsilon(\vec{x},\vec{x}_b)\,\mathrm{d}\vec{x}_b\right] \\
&= [1-\delta_\epsilon(\vec{x})]\mathcal{F}(\vec{x}) \qquad (3.10)
\end{aligned}$$

where δ_ϵ is the zeroth moment of the kernel over the body;

$$\delta_\epsilon(\vec{x}) = \int_{\Omega_b} K_\epsilon(\vec{x},\vec{x}_b)\,\mathrm{d}\vec{x}_b. \qquad (3.11)$$

Using this approach the meta-equation simply becomes a mixing function

$$\mathcal{M}(\vec{x}) = [1-\delta_\epsilon(\vec{x})]\mathcal{F}(\vec{x}) + \frac{1}{c}\delta_\epsilon(\vec{x})\mathcal{B}(\vec{x}) = 0 \qquad (3.12)$$

which ensures the regional behavior of figure 3-1 and is easily implemented into general PDE solution methods as shown in the following sections.

3.2.3 Interpolating Function Formulation

The gross properties of δ_ϵ for a 2D body immersed in a 2D fluid are sketched in figure 3-2. In the figure, $\vec{x}$ is confined to a curve passing through a the body. The figure shows $\delta_\epsilon = 0$ exterior to the body and $\delta_\epsilon = 1$ on the interior, which is in accordance with the desired behavior. The figure also shows that there is a 2ϵ wide transition region centered on the solid/fluid boundary and that $\delta_\epsilon \simeq \frac{1}{2}$ for points on that boundary. The exact value of the interpolating function δ_ϵ could be determined by evaluating the integral in equation 3.11 over the specific body geometry for every point in space. However, the general characteristics of δ_ϵ match those of a nascent

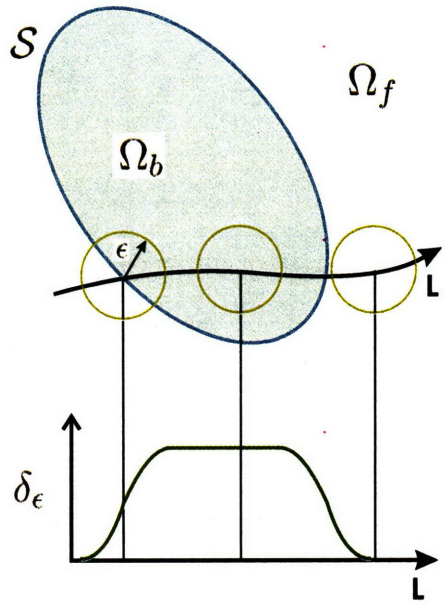


Figure 3-2: Value of the kernel zeroth moment over the body δ_ϵ given on a line L passing through a 2D body B. The integrated value is 1 within the body and 0 on the exterior with a smooth transition over 2ϵ.

heavy-side function, leading to the more direct definition

$$\delta_\epsilon^B(d) = \begin{cases} \frac{1}{2}\left[1+\sin\left(\frac{\pi}{2}\frac{d}{\epsilon}\right)\right] & \text{for} \quad \frac{|d|}{\epsilon} < 1 \\ 1 & \text{for} \quad \frac{d}{\epsilon} < -1 \\ 0 & \text{else} \end{cases} \tag{3.13}$$

where $d(\vec{x})$ is a signed distance function to nearest point on the fluid/solid interface chosen negative within the solid. Thus if a signed distance function is available equation 3.12 may be used without any integrations using $\delta_\epsilon(\vec{x}) = \delta_\epsilon^B(d(\vec{x}))$. This simplifies the numerical implementation and saves on computational cost.

The zeroth kernel moment of a 1D body immersed in a 2D space is shown in figure 3-3 (compared to figure 3-2). In both cases the maximum (magnitude 1 by definition) occurs on the body. The difference is that the immersed surface has zero width and the transition occurs on either side of the surface over a width ϵ.

This is similar to the situation when modeling surface tension using the Level-Set method or immersing a flexible filament with the Immersed Boundary method. In both of those methods the influence of the zero-thickness surface is spread over a

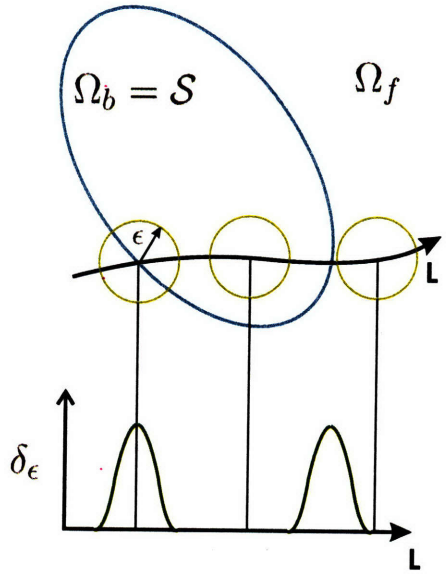


Figure 3-3: Value of the zeroth kernel moment over the body on a line L passing through a 1D body $\Omega_b = \mathcal{S}$. The integrated value is 1 on the immersed surface and 0 off the surface with a nonzero width of 2ϵ.

finite width so that it can be felt on a background grid. However, in those methods the effect is modeled as a body force added to the momentum equation. In the meta-equation formulation there is instead a transition between the fluid and solid governing equations which requires that the momentum equation 'deactivate' on the surface as seen in equation 3.12.

There is a complication to resolve before equation 3.9 can be used in the case of an immersed surface. Unlike before, $\mathcal{B}$ is now a function of a reduced set of spacial dimensions. For example if $\mathcal{B} = \mathcal{B}(\{\sigma, \tau\})$ then $\mathcal{B}(\{x, y, z\})$ has no inherent meaning unless the point $\{x, y, z\}$ lies on the surface. However, this is easily remedied. Define a correlation function $\vec{S}(\vec{x})$ such that for any point $\vec{x}$ the result is the coordinates of the closest point on the surface. Thus, the change of variables from $\{\sigma, \tau\}$ to $\{x, y, z\}$ accomplished through integration in equation 3.6 is managed directly by the function $\vec{S}$. This allows equation 3.9 to model an immersed surface using $\mathcal{B}(\vec{x}) = \mathcal{B}(\vec{S}(\vec{x}))$ and where $\delta_\epsilon^{\mathcal{S}}$ for an immersed surface is of the form

$$\delta_\epsilon^{\mathcal{S}}(d) \equiv \begin{cases} \frac{1}{2}\left[1 + \cos\left(\pi \frac{d}{\epsilon}\right)\right] & \text{for} \quad \frac{|d|}{\epsilon} < 1 \\ 0 & \text{else} \end{cases} \tag{3.14}$$

and $d(\vec{x})$ is the distance from the general point $\vec{x}$ to the nearest point on the surface. Further details on the construction of $\vec{S}$ and d for general immersed surfaces is given in section 5.2.

3.2.4 Vector Governing Equations

The final generalization step is to develop a method of handling vector governing equations such as the Navier-Stokes equations. This is accomplished by decomposing the vector equation into its scalar basis, ie

$$\vec{\mathcal{F}}(\vec{x}) = \sum_i \mathcal{F}_i(\vec{x})\hat{e}_i$$

where $\hat{e}_i$ are the unit basis vectors. The interpolation functions are applied to the component equations, ie

$$\vec{\mathcal{P}}_\epsilon\left\{\vec{\mathcal{F}},\vec{x}\right\} = \sum_i \mathcal{P}_\epsilon\{\mathcal{F}_i(\vec{x})\}\hat{e}_i$$

resulting in a meta equation with an identical form to the previous two sections but with vector fluid and solid governing equations, ie

$$\vec{\mathcal{M}}(\vec{x}) = [1-\delta_\epsilon(\vec{x})]\vec{\mathcal{F}}(\vec{x}) + \frac{1}{c}\delta_\epsilon(\vec{x})\vec{\mathcal{B}}(\vec{x}) = 0. \tag{3.15}$$

It is commonly the case that the vector body equations are most simply written using a local normal-tangential basis instead of the global basis, especially in the case of an immersed surface. In that case a local rotation matrix $\mathbf{C}$ must be used to perform a coordinate transformation on the equation components before they are mixed with the fluid equations, ie

$$\begin{aligned}\vec{\mathcal{Q}}_\epsilon\left\{\vec{\mathcal{B}},\vec{x}\right\} &= \sum_i \hat{e}_i\left(\int_{\Omega_b}\left(\sum_j C_{ij}(\vec{x}_b)\mathcal{B}_j(\vec{x}_b)\right)K_\epsilon(\vec{x},\vec{x}_b)\,\mathrm{d}\vec{x}_b\right)\\ &\simeq \delta_\epsilon(\vec{x})\mathbf{C}(\vec{x})\vec{\mathcal{B}}(\vec{x})\end{aligned} \tag{3.16}$$

where the rotated function has been pulled out of the integral using $\mathbf{C}(\vec{x}) = \mathbf{C}(\vec{S}(\vec{x}))$ as in the previous sections.

This meta-equation formulation also allows for the possibility of only changing individual components of the governing equations. For example if the tangential components of the governing equations are equal ($\mathcal{B}_\sigma = \mathcal{F}_\sigma, \mathcal{B}_\tau = \mathcal{F}_\tau$) but the normal components are not then the meta-equation takes on the form

$$\begin{aligned}
\vec{\mathcal{M}}(\vec{x}) &= \sum_i \hat{e}_i \left([1 - \delta_\epsilon(\vec{x})] \mathcal{F}_i(\vec{x}) + \frac{1}{c} \delta_\epsilon(\vec{x}) \mathcal{B}_i(\vec{x}) \right) \\
&= \sum_i \hat{e}_i \sum_j e_i e_j \left([1 - \delta_\epsilon(\vec{x})] \mathcal{F}_j(\vec{x}) + \frac{1}{c} \delta_\epsilon(\vec{x}) \mathcal{B}_j(\vec{x}) \right) \\
&= \sum_i \hat{e}_i \sum_j [n_i(\vec{x}) n_j(\vec{x}) + \sigma_i(\vec{x}) \sigma_j(\vec{x}) + \tau_i(\vec{x}) \tau_j(\vec{x})] \left([1 - \delta_\epsilon(\vec{x})] \mathcal{F}_j(\vec{x}) + \frac{1}{c} \delta_\epsilon(\vec{x}) \mathcal{B}_j(\vec{x}) \right) \\
&= \sum_i \hat{e}_i \sum_j \left([e_i e_j - \delta_\epsilon(\vec{x}) n_i(\vec{x}) n_j(\vec{x})] \mathcal{F}_j(\vec{x}) + \frac{1}{c} \delta_\epsilon(\vec{x}) n_i(\vec{x}) n_j(\vec{x}) \mathcal{B}_j(\vec{x}) \right) \\
&= [\mathbf{I} - \delta_\epsilon(\vec{x}) \mathbf{N}(\vec{x})] \vec{\mathcal{F}}(\vec{x}) + \frac{1}{c} \delta_\epsilon(\vec{x}) \mathbf{N}(\vec{x}) \vec{\mathcal{B}}(\vec{x}) = 0
\end{aligned} \tag{3.17}$$

In the first line, the meta equation is written out for each component. In the second line, the equations are taken inside the summation over j multiplied by the Cartesian basis matrix $e_i e_j$. In the third line the Cartesian matrix is split into normal and tangential components where $\hat{n}$ is the unit normal vector. In the next line the coefficient $\frac{1}{c}$ is set to unity for the tangential equations, allowing them to cancel. In the final line the component equations are written back in tensor notation using the identity tensor $\mathbf{I}$ and the normal dyad $\mathbf{N}$. The final result, equation 3.17, is simply a formal way of expressing that over the body $\mathcal{F}_n$ deactivates and $\mathcal{B}_n$ activates.

3.3 Application of BDIM to Canonical Example Systems

Before applying the boundary data immersion method of the previous section to the Navier-Stokes equations it is applied to a set of simple sample problems; channel flow and the potential flow around a sphere. These cases are instructive and establish the

method's numerical properties.

3.3.1 Poiseuille Flow

For the first simple example, consider the steady flow in a channel between two parallel plates; the Poiseuille flow. As the flow is steady and uniform in the x-direction, the system is described by the simple boundary value problem

$$\mathcal{F} = f_x + \nu \frac{d^2u}{dy^2} = 0 \tag{3.18}$$

$$\mathcal{B} = u(0) = u(L) = 0 \tag{3.19}$$

where f_x is the applied horizontal body force, $u = u(y)$ is the velocity component, ν is the kinematic viscosity, and L is the width of the channel. Equation 3.18 is the conservation of x-momentum and equation 3.19 are the no-slip boundary conditions on the walls at $y = 0$ and $y = L$. The solution of this second order ordinary differential equation is simply

$$u = \frac{f_x}{2\nu} y(L - y) \tag{3.20}$$

Equation 3.12 is applied to this boundary value problem, resulting in the meta-equation

$$\left\{ \frac{1}{c}\delta_\epsilon(y) - [1 - \delta_\epsilon(y)]\frac{d^2}{dy^2} \right\} u_\epsilon = [1 - \delta_\epsilon(y)]\frac{f_x}{\nu} + \frac{1}{c}\delta_\epsilon(y)U \tag{3.21}$$

where u_ϵ is the solution to this smoothed version of the governing equations and U is the boundary value, zero in this case.

Before moving on it is important to show that the equations are consistent. In other words, as $\epsilon \to 0$ we recover $u_\epsilon \to u$ for all y in the channel and $u_\epsilon \to U$ for all other values of y. Posing equation 3.21 on a boundary fitted grid and letting $\epsilon \to 0$ verifies this property.

First, tessellate the line into N pieces of length Δy and set $y_1 = 0$ and $y_N = L$.

Then form the vector $\mathbf{u}$ with components u_1 to u_N corresponding to the values of the velocity field at $u(y_1)$ to $u(y_N)$. The second derivative is required, and is estimated using the second order difference matrix $\mathbf{D}^2$ with components

$$D^2_{ij} \equiv \begin{cases} \frac{-2}{\Delta y^2} & \text{for} \quad i = j \\ \frac{1}{\Delta y^2} & \text{for} \quad i = j \pm 1 \\ 0 & \text{else} \end{cases} . \tag{3.22}$$

Finally, taking $c = \Delta y^2$ to balance the solid and fluid influence and setting $\epsilon = 0$ results in the linear algebra system

$$\frac{1}{\Delta y^2} \begin{pmatrix} 1 & & & & & \\ -1 & 2 & -1 & & & \\ & & \ddots & & & \\ & & -1 & 2 & -1 & \\ & & & & 1 & \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \frac{f}{\nu} \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 1 \\ 0 \end{pmatrix} \tag{3.23}$$

for $\mathbf{u}$. The solution of system 3.23 is the exact solution at every point in the fluid domain. Additionally, the exact solution (zero) is achieved outside the fluid domain. Thus consistency is established.

The above example used a boundary-fitted grid but equation 3.21 can just as easily be posed on a background grid whose topography does not match the solid body boundaries. The only change to the above system is to use the smoothed form of the interpolation function δ_ϵ given by equation 3.13 where the distance function is given by

$$d(y) = \frac{L}{2} - \left| \frac{L}{2} - y \right| \tag{3.24}$$

for this geometry.

As mentioned in the previous section the interpolation functions must be resolved in order for the solution to be independent of the background grid, meaning that $\epsilon > \Delta y$. A number of simple tests to determine the influence of the parameter ϵ

ϵ/L	$\|E\|_1/u_{max}$	$\|E\|_2/u_{max}$	$\|E\|_\infty/u_{max}$
0.02	0.08306	0.08369	0.08524
0.01	0.02922	0.02930	0.02953
0.005	0.00791	0.00792	0.00797

Table 3.2: Error metrics of the numerical solution to the Poiseuille flow. The errors $E = u_\epsilon - u$ are normalized by the maximum analytic velocity $u_{max} = \frac{fL^2}{8\nu}$. The convergence rate is second order for all metrics.

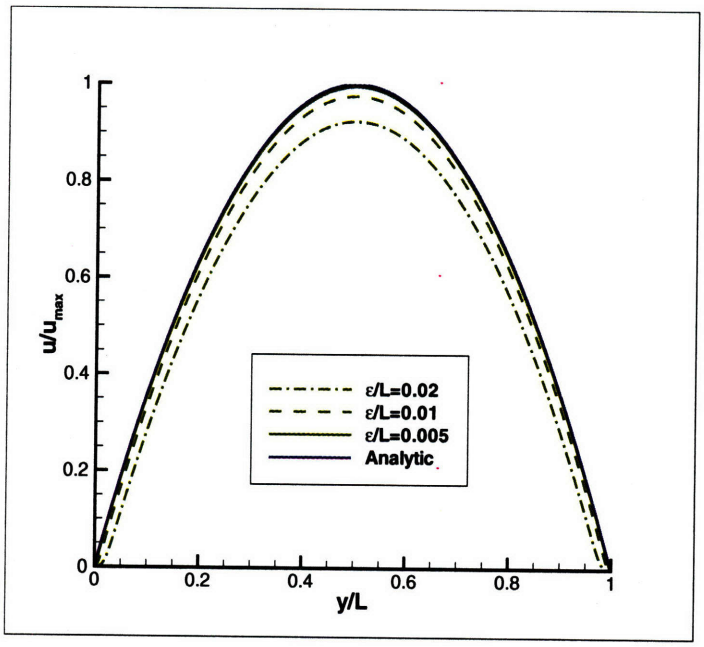


Figure 3-4: Numerical solution to the Poiseuille flow for different values of ϵ.

are run and the results are shown in table 3.2 and figure 3-4. In these tests the grid spacing is set to $\Delta y/L = 0.001$ and the grid points are set such that no two points lined up with the edges of the channel. Both the table and figure show the second order convergence of the numerical solution to the analytic solution as $\epsilon \to 0$. Additionally, a study on the variation of the solution due to shifts of the grid found that the error metrics changed less than 0.001% even for the smallest value of ϵ. This supports the conclusion that 5 points within the interpolated region is sufficient to produce a solution which is shift-independent.

As is typical for preliminary examples, this section is meant to be more instructive than practical. For a 1D example the analytic meta-equation is hardly worth the effort because defining a boundary-fitted grid is a simple exercise. The next section

ϵ/L	$\|E_I\|_1/u_{max}$	$\|E_I\|_2/u_{max}$	$\|E_p\|_1/u_{max}$	$\|E_p\|_2/u_{max}$
0.05	0.09715	0.09875	0.41828	0.43741
0.025	0.03392	0.03414	0.12516	0.12886
0.0125	0.00717	0.00714	0.05727	0.05733
0.006125	0.00725	0.00752	0.01204	0.01205

Table 3.3: Error metrics of the numerical solution to the Poiseuille flow. The E_I values are for the solution to the full integral equations, while the E_p values are for the point-scaling equations.

demonstrates the usefulness of the method in a more complex 2D example.

3.3.2 Integral Approximation Validation

The Poiseuille flow offers one more opportunity: to quantify the magnitude of the error induced by approximating the interpolating kernel integrals of equation 3.6 and 3.7 with the point-scaling of equations 3.9 and 3.10. As mentioned above, the integral form of the equations is directly applicable, but its implementation requires major modification to the existing 3D fluid solver. This simple 1D test case allows us to test the relative accuracy of the more naturally implemented point-scaling meta-equations.

Solutions for both the integral and point equations are generated using the same background grid as the previous section and $\epsilon/L = 0.05, 0.025, 0.0125, 0.006125$. The results, shown in figure 3-5 and table , demonstrate that the error reduces by a factor of 4 when using the full integral equation for the larger values of ϵ. This indicates that the point-scaling approximation pays a toll in accuracy on the order of 400% for all but the smallest ϵ.

The smallest value $\epsilon/L = 0.006125$ actually demonstrated an increased error in the integral solution, probably due to insufficient resolution of the smoothing region. This indicates that the required resolution of the kernel width is slightly larger than when using the point-scaling equations; say $\epsilon/\Delta x$ 10. Based on this rough value, the matrix for the integral equations is around 10 times less sparse than the point-scaling equations. This is because while the point-scaling equations only reference their nearest neighbor, the kernel integrals reference every point within ϵ radius. Scaling this effect up to 3D and assuming an efficient conjugate gradient method is

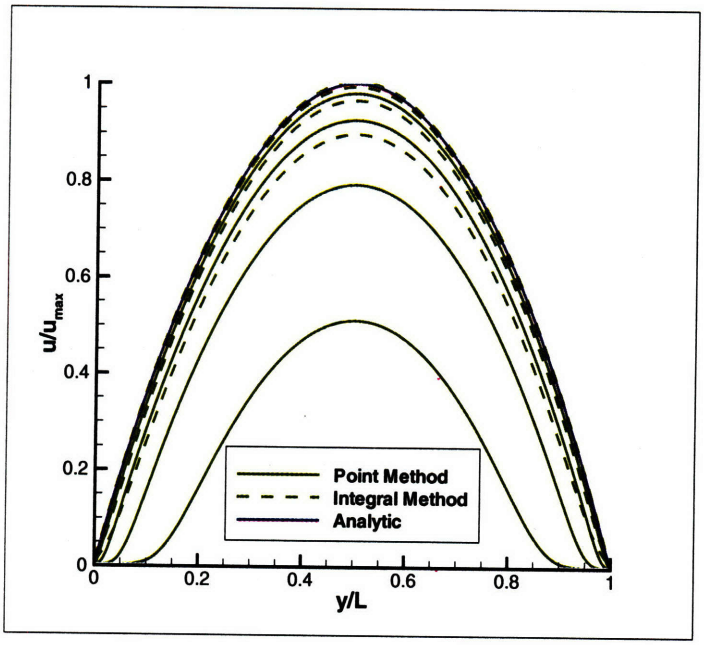


Figure 3-5: Numerical solution to the Poiseuille flow for different values of ϵ. The solid lines are the solutions of the point-scaling equations, while the dashed lines are for the full integral equations.

used for inversion, a cost increase of O(100) is quite conservative. Therefore, while evaluating the kernel functions directly is more accurate, it is less cost-effective than the point-integral method.

3.3.3 Potential Flow

Next, the boundary data immersion method is applied to a general 2D potential flow system for the second canonical example. Define a 2D velocity field $\vec{u}$ with the governing equations

$$\mathcal{F} = \vec{u}(\vec{x}_f) - \vec{\nabla}\phi(\vec{x}_f) = 0 \tag{3.25}$$

$$\mathcal{B} = \vec{u}(\vec{x}_b) - \vec{U}(\vec{x}_b) = 0 \tag{3.26}$$

where the first equation is the velocity field definition for all points $\vec{x}$ in the fluid domain and the second equation is the boundary condition requiring that the velocity match the prescribed velocity $\vec{U}$ for all points $\vec{x}_b$ on the body. Next, a meta-equation for the velocity field is constructed which is valid over both the fluid and the body

using equation 3.15

$$\vec{u}_\epsilon(\vec{x}) = [1 - \delta_\epsilon(\vec{x})]\vec{\nabla}\phi_\epsilon(\vec{x}) + \delta_\epsilon(\vec{x})\vec{U}(\vec{x}) \tag{3.27}$$

where again, $\vec{u}_\epsilon$ is the velocity solution of this smoothed system, ϕ_ϵ is the smoothed velocity potential and the factor $\frac{1}{c}$ has been set to unity.

This equation has one distinct difference from the previous examples, namely the unknown field ϕ_ϵ. As in standard potential flow methods the divergence-free constraint

$$\vec{\nabla} \cdot \vec{u}(\vec{x}) = \nabla^2\phi(\vec{x}) = 0 \tag{3.28}$$

is required to solve for the potential, and therefore the velocity field. The governing equation for the potential ϕ_ϵ is derived by taking the divergence of equation 3.27 to write

$$\vec{\nabla} \cdot [1 - \delta_\epsilon(\vec{x})]\vec{\nabla}\phi_\epsilon(\vec{x}) = -\vec{\nabla} \cdot \delta_\epsilon(\vec{x})\vec{U}(\vec{x}). \tag{3.29}$$

As in the previous section, when the width $\epsilon \to 0$ we recover $\vec{u}_\epsilon \to U$ on the body, and the exact solutions $\phi_\epsilon \to \phi$ and $\vec{u}_\epsilon \to \vec{u}$ in the fluid. In this case, the field ϕ_ϵ has no analog inside the body, and is only required to remain finite.

In this 2D example the advantage of the analytic meta-equation becomes clear. The burden is shifted from solving a Laplace equation over a potentially highly complex domain to that of solving a variable coefficient Poisson equation over the most convenient background space. The strategy is roughly analogous to the shift from a constrained optimization problem to an unconstrained optimization problem using Lagrange multipliers. The boundary 'constraints' have been incorporated into a new governing equation.

Note that the coefficients of the Poisson equation are zero on the body which corresponds to the proper Neumann boundary condition for the potential on the body interface. To illustrate this, let us quickly pose the 1D form of equation 3.29 on a uniform variable-staggard boundary-fitted grid with spacing Δx and $\epsilon = 0$. The

grid point x_1 is placed on the boundary and thus the boundary condition applies, ie $u(x_1) = u_1 = U_1$. The potential ϕ_i is staggard halfway between u_i and u_{i+1}. The poisson equation at point i is written as

$$\left(p_{i+1}\frac{\phi_{i+1}-\phi_i}{\Delta x} - p_i\frac{\phi_i-\phi_{i-1}}{\Delta x}(\vec{x})\right)/\Delta x = -\frac{q_{i+1}U_{i+1}-q_iU_i}{\Delta x} \tag{3.30}$$

which, for $i = 1$ simplifies to

$$\left(\frac{\phi_2-\phi_1}{\Delta x} - U_1\right)/\Delta x = 0 \tag{3.31}$$

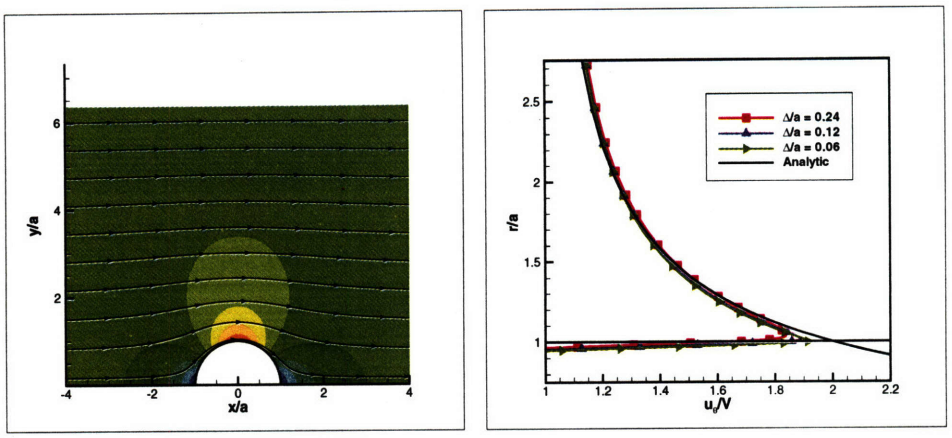
which is exactly the form of the Laplace equation on the boundary after substitution of the Nuemann boundary condition. As is well known, if all the boundaries have Nuemann conditions applied on them the resulting system is degenerate, and this is also the case for the meta equation. In addition, the meta equation is degenerate within the body because ϕ_ϵ is not strictly defined there. Both issues are trivially circumvented by treating the system as a minimization problem and solving with the conjugate gradient method. This is equivalent to minimizing the energy of the system as in variational methods.

3.3.4 Potential Flow: General Scaler BCs

Note that equation 3.26 actually supplies more information than is required for determining the velocity potential. While a vector boundary condition such as 3.26 is required for flows with vorticity (and is used below) scaler boundary conditions are appropriate here. The possible Dirichlet and Neumann boundary conditions are

$$\mathcal{B}_1 = \phi(\vec{x}_{b1}) - \Phi(\vec{x}_{b1}) = 0 \quad \forall \quad \vec{x}_{b1} \in \Omega_{b1} \tag{3.32}$$

$$\mathcal{B}_2 = \frac{\partial\phi}{\partial n}(\vec{x}_{b2}) - \frac{\partial\Phi}{\partial n}(\vec{x}_{b2}) = 0 \quad \forall \quad \vec{x}_{b2} \in \Omega_{b2} \tag{3.33}$$



(a) Potential ϕ_ϵ and stream lines for $\epsilon = 0.06$

(b) Velocity profile u_ϵ on $\theta = \pi/2$

Figure 3-6: Solution of the no-penetration meta-equation for the potential flow past a circular cylinder

where the body has been split into two parts Ω_{b1} and Ω_{b2} to remain as general as possible. These solid equations can be mixed directly with fluid equation 3.28

$$\mathcal{F} = \nabla^2\phi(\vec{x}_f) = 0 \quad \forall \quad \vec{x}_f \in \Omega_f \tag{3.34}$$

leading to a general potential flow meta equation

$$\mathcal{M} = \vec{\nabla} \cdot [\mathbf{I} - \delta_{\epsilon 2}\mathbf{N}]\,\vec{\nabla}\phi_\epsilon - \frac{\partial}{\partial n}\delta_{\epsilon 2}\frac{\partial \Phi}{\partial n} = 0 \tag{3.35}$$

where $\delta_{\epsilon 1}$ and $\delta_{\epsilon 2}$ are the interpolation functions over the two bodies. Equation 3.35 thus demonstrates how simple it is to construct a general-meta equation even when bodies with different types of governing equations are immersed in the system. Note that the interpolation coefficient in front of the Laplacian operator is bounded by definition ($0 \leq 1 - \delta_{\epsilon 1} - \delta_{\epsilon 2} \leq 1$) and the numerical implementation must honor that requirement.

As with a channel flow, a series of numerical experiments are performed to quantify the effect of the finite interpolation width in the meta-equations. A circular cylinder benchmark case is used for these tests. The exact solution for this flow is given in

cylindrical coordinates by

$$\phi = Vr\left(1 + \frac{a^2}{r}\right)\cos\theta \tag{3.36}$$

where V is the free stream velocity and a is the circle radius. The Dirichlet body is set to be one half of the circle $\{0 < r \leq a, -\pi/2 < \theta < pi/2\}$ and the Neumann body is set to be the other half of the circle. The normal is the radial coordinate for this geometry $\hat{n} = \hat{r}$ and for any point $\{r, \theta\}$ the closest point on the solid/fluid interface is simply $\{a, \theta\}$.

Using these relations, equation 3.35 is posed on a fine Cartesian-grid using $c1 = \Delta x^2$, $c2 = \Delta x$ and varying $\epsilon/a = 0.06, 0.12, 0 : 24$. The resulting system is inverted using a preconditioned conjugate gradient method.

Figure 3-6(a) shows the computed potential and streamlines, which match the expected dipole solution. Figure 3-6(b) plots the velocity profiles for all three solutions along with the analytic solution. As expected, we see the velocity $u_\epsilon \rightarrow u$ in the flow as $\epsilon \rightarrow 0$.

This chapter introduced the concept of meta-equations and the Boundary Data Immersion Method (BDIM) to generate those meta-equations for general fluid-body systems. BDIM

Chapter 4

Application of the Boundary Data Immersion Method to General Flows

With the Boundary Data Immersion Method (BDIM) well verified by test problems, it is now extended to prediction of systems with bodies immersed in general 3D fluid flows. First, this extension requires derivation of the proper meta-equations for a variety of possible boundary conditions. Next, BDIM is tested for compatibility against the VOF free surface method of chapter 2 by running a series of 2D internal flows. These tests also provide an example of computing forces on immersed surfaces, and the challenges this presents on Cartesian-grids is discussed.

A numerical exit is critical for examining external flows and a new method of dealing with this difficult boundary is derived and tested based on the BDIM framework. The exit capability allows the BDIM and VOF methods to be validated against experimental and numerical studies of external flows. The unsteady flow of a 2D wave-maker and a 3D semi-submerged sphere are chosen as test cases.

4.1 Meta-Equation for General Flows

In this section meta-equations are derived governing the immersion of a solid body into a general fluid. Jumping directly to the completely general case is possible, but examining each case individually is insightful.

4.1.1 No-slip Bodies

Starting with the immersion of a no-slip body we have

$$\vec{\mathcal{F}} = \vec{u}(\vec{x}_f,t) - \vec{u}(\vec{x}_f,t_0) - \int_{t_0}^{t}\left[\vec{r}(\vec{x}_f,t') - \frac{1}{\rho(\vec{x}_f,t')}\vec{\nabla}p(\vec{x}_f,t')\right]\,\mathrm{d}t' = 0 \quad (4.1)$$

$$\vec{\mathcal{B}} = \vec{u}(\vec{x}_b,t) - \vec{U}(\vec{x}_b,t) = 0 \quad (4.2)$$

where the time integral of equation 1.2 has been taken resulting in an explicit update equation. These governing equations are substituted into equation 3.15 resulting in the vector meta-equation

$$\begin{aligned}[1-\delta_\epsilon(\vec{x},t)] \quad & \left(\vec{u}_\epsilon(\vec{x},t) - \vec{u}_\epsilon(\vec{x},t_0) - \int_{t_0}^{t}\left[\vec{r}(\vec{x},t') - \frac{1}{\rho(\vec{x},t')}\vec{\nabla}p_\epsilon(\vec{x},t')\right]\,\mathrm{d}t'\right)\\ +\delta_\epsilon(\vec{x},t) \quad & \left(\vec{u}_\epsilon(\vec{x},t) - \vec{U}(\vec{x},t)\right) = 0\end{aligned}$$

which, by combining terms and omitting the dependent variables, can be written as

$$\vec{u}_\epsilon - \vec{u}_\epsilon^0 = \delta_\epsilon(\vec{U} - \vec{u}_\epsilon^0) + [1-\delta_\epsilon]\int_{t_0}\vec{r} - \frac{1}{\rho}\vec{\nabla}p_\epsilon\,\mathrm{d}t \quad (4.3)$$

where $\phi^0 = \phi(t_0)$. As with the meta-equations in chapter 3 equation 4.3 is merely a blended version of the governing equations valid over the full domain. The interpolation function δ_ϵ is used to extend the influence of the fluid and solid equations and transition between them over the width ϵ at the boundary of Ω_f and Ω_b. As the blending width $\epsilon \to 0$ we recover $\vec{u}_\epsilon \to \vec{u}$ in Ω_f and $\vec{u}_\epsilon \to \vec{U}$ in Ω_b. This analytic transition from the field equations to the boundary conditions differs significantly from other methods; Body-Force, Immersed-Boundary or Cut-Cell. Additionally, the arguments

on smooth vs. sharp interface raised in chapter 3 apply here automatically. If the user sets $\epsilon = 0$ and uses a boundary-fitted grid, the standard equations are recovered. If $\epsilon = 0$ on a background grid then special numerical methods are needed to compute the fluxes, resulting in a Cut-Cell method. As long as ϵ is resolved by the background grid, no such terms are required to maintain the order of accuracy.

A meta-equation for the pressure is also required, and in an incompressible flow it is derived by taking the divergence of 4.3 and setting $\vec{\nabla} \cdot \vec{u}_\epsilon(\vec{x}) = 0$, ie

$$-\vec{\nabla} \cdot \vec{u}_\epsilon^0 = \vec{\nabla} \cdot \left(\delta_\epsilon (\vec{U} - \vec{u}_\epsilon^0) + [1 - \delta_\epsilon] \int_{t_0} \vec{r} - \frac{1}{\rho} \vec{\nabla} p_\epsilon \, dt \right) \tag{4.4}$$

which eliminates the unknown $\vec{u}_\epsilon$ from the equation. This equation can be inverted for the pressure as soon as a time integral quadrature is choosen, as done below. The Neumann boundary condition on the pressure is

$$\frac{\partial p}{\partial n} = \rho \left(r_n - \frac{\partial U_n}{\partial t} \right) \tag{4.5}$$

obtained from direct substitution of the no-penetration condition into the momentum equation. In most fitted-grid methods the pressure on the boundary is updated with this equation as the pressure is inverted. However, this is not required. Recall that the mathematical role of the pressure is a Lagrange-multiplier, enforcing the divergence free constraint on the admissible velocity fields. On the body, this constraint is not active, and thus the pressure gradient has no mathematical influence on the equations.

In a more rigorous sense, there are always two options for completing the rank of a linear system based off of a boundary value problem. One can either add equations to the system for points on the boundary or directly substitute the boundary equations into the linear system, eliminating references to the boundary points. The meta-equation approach does both automatically. Equation 4.3 adds equations for the velocity on the body, completing the momentum equation. This automatically substitutes boundary condition 4.5 into equation 4.4, eliminating references to the fictitious pressure on the body and completing the pressure system. [1]

[1]This detailed explanation may comfort readers familiar with linear systems, but I find the

To solve equations 4.3 and 4.4 a method of estimating the time integral is required. The second-order Runge-Kutta (or Huen's) method uses an explicit-Euler predictor step followed by an explicit trapezoidal corrector step. For example, to estimate the integral

$$I(t) \equiv I(t_0) + \int_{t_0}^{t} h(I(t'), t')\ \mathrm{d}t'$$

we have

$$I' - I(t_0) = \Delta t\, h(I(t_0), t_0) = \Delta t\, h^0 + O(1)$$

$$I(t_0 + \Delta t) - I(t_0) = \frac{\Delta t}{2}(h^0 + h') + O(2)$$

where Δt is the time step and $\dot{\phi}'$ are the values after the predictor step. This approach is commonly used for unsteady flows and [56] demonstrates its low storage and stability properties. Applying this method to equations 4.3 and 4.4 results in an update sequence of the form

$$\begin{aligned}
&\texttt{predictor} \\
\vec{\nabla} \cdot \frac{1-\delta_\epsilon}{\rho^0} \vec{\nabla} p_\epsilon^0 &= \vec{\nabla} \cdot \left(\vec{r_{ns}}^{\,0} + \frac{\vec{u}_\epsilon^0}{\Delta t} \right) \\
\vec{u}_\epsilon' &= \vec{u}_\epsilon^0 + \Delta t \vec{f}^0 \\
&\texttt{corrector} \\
\vec{\nabla} \cdot \frac{1-\delta_\epsilon}{\rho'} \vec{\nabla} p_\epsilon' &= \vec{\nabla} \cdot \left(\vec{r_{ns}}' + \frac{\vec{u}_\epsilon' + \vec{u}_\epsilon^0}{\Delta t} \right) \\
\vec{u_\epsilon} &= \frac{1}{2}(\vec{u}_\epsilon' + \vec{u}_\epsilon^0 + \Delta t \vec{f}')
\end{aligned}$$

where $\vec{r_{ns}}$ is the new RHS vector for flows with immersed no-slip bodies and $\vec{f}$ is the total forcing, given by

$$\begin{aligned}
\vec{r_{ns}} &= [1-\delta_\epsilon]\vec{r} + \delta_\epsilon \frac{\vec{U} - \vec{u}_\epsilon^0}{\Delta t} \\
\vec{f} &= \vec{r_{ns}} - \frac{1-\delta_\epsilon}{\rho} \vec{\nabla} p_\epsilon.
\end{aligned}$$

This process is identical to the original solver algorithm with the substitution of $\vec{r_{ns}}$

Lagrange-multiplier argument more physically insightful.

for $\vec{r}$ and $\frac{1-\delta_\epsilon}{\rho}$ for $\frac{1}{\rho}$. As such, the alteration to the main solver is trivial, and the sole difficulty in computing flows with immersed solids lies in determining the values of δ_ϵ and $\vec{U}$ for the given problem. The above equations can be formulated for any number of arbitrary bodies with any given velocity boundary conditions. Also, Ω_b does not need to be a material surface, the inflow and outflow boundaries can be modeled with this formulation by setting $\vec{U}$ appropriately, as done in section 4.4. Thus the method is completely modular and general, allowing for rapid inclusion into the existing software.

4.1.2 No-Penetration Surfaces

The meta-equations above are based off of the 'true' no-slip boundary condition for solids immersed in a flow but this is not always the best choice for simulation purposes. Especially in Cartesian-grid methods it can be very difficult to fully resolve the near-wall viscous flow. The scales of such flows scale inversely with the Reynolds number and modeling the body with a slip-condition of some kind is more computationally practical. A first attempt at that goal is to develop a meta-equation which only enforces the no-penetration condition across the immersed body, given by

$$\mathcal{B}_n = \vec{u}(\vec{x}_b) \cdot \hat{n}(\vec{x}_b) - U_n(\vec{x}_b) = 0 \tag{4.6}$$

where again, $\hat{n}$ is the surface normal. Applying the result from equation 3.17 immediately gives the meta-equation for the momentum

$$\vec{u}_\epsilon - \vec{u}_\epsilon^0 = \delta_\epsilon(U_n\hat{n} - \mathbf{N}\vec{u}_\epsilon^0) + [\mathbf{I} - \delta_\epsilon\mathbf{N}]\int_{t_0} \vec{r} - \frac{1}{\rho}\vec{\nabla}p_\epsilon \, \mathrm{d}t \tag{4.7}$$

and pressure

$$-\vec{\nabla}\cdot\vec{u}_\epsilon^0 = \vec{\nabla}\cdot\left(\delta_\epsilon(U_n\hat{n} - \mathbf{N}\vec{u}_\epsilon^0) + [\mathbf{I} - \delta_\epsilon\mathbf{N}]\int_{t_0} \vec{r} - \frac{1}{\rho}\vec{\nabla}p_\epsilon \, \mathrm{d}t\right) \tag{4.8}$$

where $\mathbf{I}$ is the identity tensor and $\mathbf{N} \equiv \hat{n}\hat{n}$ is the normal dyad. As detailed in section 3.2.4 these meta-equations enforce the fluid equations everywhere except on the on

the normal component of the velocity near the body, where the body condition is enforced. The predictor-corrector algorithm above is adjusted by changing $\frac{1-\delta_\epsilon}{\rho}$ to $\frac{\mathbf{I}-\delta_\epsilon\mathbf{N}}{\rho}$ and and $\vec{r}_{ns}$ to $\vec{r}_{np}$

$$\vec{r}_{np} = [I - \delta_\epsilon \mathbf{N}]\vec{r} + \delta_\epsilon \frac{U_n \hat{n} - \mathbf{N}\vec{u}_\epsilon^0}{\Delta t}.$$

Although similar in form to the no-slip equations, the no-penetration equations feature tensor products instead of simple multiplication which make numerical solution of the no-penetration equations more laborious. The source term of equation 4.8 is easily constructed, but the Poisson coefficients give rise to cross derivatives. These terms arise because of the conflict of the global (Cartesian) and local (normal-tangential) coordinates and results in a non-symmetric matrix which is less sparse than the original.

However, the modeling concerns resulting from equation 4.7 are more serious than the numerical problems of inverting equation 4.8. The no-penetration equations do allow the tangential velocity on the body to vary from zero, but at the cost of keeping the tangential momentum equation active across the body. For an immersed solid this is clearly nonphysical, but it is also incorrect for an immersed surface because the flows on either side of the surface can interact.

4.1.3 General Slip-Model Meta-Equations

The additional complexity arises in the no-penetration case because equation 4.6 is a scalar equality and can therefore only make a rank-one adjustment to the governing equations. The adjustment is made through the rank-one dyad $\mathbf{N}$, leaving the $(\mathcal{N}-1)$ tangential components of the momentum equation unchanged by the presence of the body. However, these tangential equations are free to carry any user-defined slip model and the standard choice in CFD is a Neumann condition

$$\frac{\partial \phi}{\partial n} = 0$$

but some caution is required in the more general setting of immersed body equations. The Neumann condition does not mean the normal derivatives are zero *across* the surface, rather that they approach zero *from either side.* Therefore, in this thesis

$$\left.\frac{\partial \phi}{\partial n}\right|_{\vec{x}} = \lim_{d \to 0^{\pm}} \frac{\phi(\vec{x}_{\pm} + d\,\hat{n}) - \phi(\vec{x}_{\pm})}{d} = 0 \tag{4.9}$$

which allows the function to be discontinuous across the interface. With that technicality understood, the tangential body equations are written as

$$\mathcal{B}_{\sigma} = \frac{\partial u_{\sigma}(\vec{x}_b)}{\partial n} = 0 \tag{4.10}$$

$$\mathcal{B}_{\tau} = \frac{\partial u_{\tau}(\vec{x}_b)}{\partial n} = 0 \tag{4.11}$$

where $\hat{\sigma}$ and $\hat{\tau}$ are the local tangential vectors. These equations along with equation 4.6 and equation 4.1 are substituted into equation 3.16 to give

$$\vec{u}_{\epsilon} + \vec{\mathcal{L}}(\vec{u}_{\epsilon}) - \vec{u}_{\epsilon}^{0} = \delta_{\epsilon}(U_n \hat{n} - \vec{u}_{\epsilon}^{0}) + [1 - \delta_{\epsilon}] \int_{t_0} \vec{r} - \frac{1}{\rho}\vec{\nabla} p_{\epsilon}\, \mathrm{d}t \tag{4.12}$$

where $\mathcal{L}$ is the left hand side operator, defined as

$$\vec{\mathcal{L}}(\vec{u}_{\epsilon}) = \delta_{\epsilon}[\mathbf{I} - \mathbf{N}] \left(\frac{1}{c} \frac{\partial \vec{u}_{\epsilon}}{\partial n} - \vec{u}_{\epsilon} \right) \tag{4.13}$$

for the Neumann condition. Apart from $\vec{U} \to U_n \hat{n}$ this LHS term is the only difference between the Neumann meta-equation and the no-slip equation. The blending factor $\frac{1}{c}$ is set to Δx to equate the order of magnitude of the two operators. The pressure equation

$$\vec{\nabla} \cdot \left(\vec{\mathcal{L}}(\vec{u}_{\epsilon}) - \vec{u}_{\epsilon}^{0} \right) = \vec{\nabla} \cdot \left(\delta_{\epsilon}(U_n \hat{n} - \vec{u}_{\epsilon}^{0}) + [1 - \delta_{\epsilon}] \int_{t_0} \vec{r} - \frac{1}{\rho}\vec{\nabla} p_{\epsilon}\, \mathrm{d}t \right) \tag{4.14}$$

is also similar to the no-slip version but the addition of the left hand side operator keeps the unknown velocity $\vec{u}_{\epsilon}$ from canceling fully. Luckily, a simple order of magnitude analysis shows that the term is very small, on the order of ϵ. It can be safely

modeled explicitly or, as demonstrated in the next sections, ignored entirely.

Moving on to the case of a general Robin condition on the tangential velocity, ie

$$\mathcal{B}_\sigma = \alpha \frac{\partial u_\sigma(\vec{x}_b)}{\partial n} + (1-\alpha)\left(\vec{u}_\sigma(\vec{x}_b) - \vec{U}_\sigma(\vec{x}_b)\right) = 0 \tag{4.15}$$

the predictor-corrector method can be generally stated as

predictor (4.16)

$$\begin{aligned} \vec{\nabla}\cdot\frac{1-\delta_\epsilon}{\rho^0}\vec{\nabla}p_\epsilon^0 &= \vec{\nabla}\cdot\left(\vec{r_\alpha}^0 + \frac{\vec{u}_\epsilon^0}{\Delta t}\right) \\ \vec{u}_\epsilon' + \vec{\mathcal{L}}_\alpha(\vec{u}_\epsilon') &= \vec{u}_\epsilon^0 + \Delta t\vec{f}^0 \end{aligned}$$

corrector (4.17)

$$\begin{aligned} \vec{\nabla}\cdot\frac{1-\delta_\epsilon}{\rho'}\vec{\nabla}p_\epsilon' &= \vec{\nabla}\cdot\left(\vec{r_\alpha'} + \frac{\vec{u}_\epsilon' + \vec{\mathcal{L}}_\alpha(\vec{u}_\epsilon') + \vec{u}_\epsilon^0}{\Delta t}\right) \\ \vec{u}_\epsilon + \vec{\mathcal{L}}_\alpha(\vec{u}_\epsilon) &= \frac{1}{2}(\vec{u}_\epsilon' + \vec{\mathcal{L}}_\alpha(\vec{u}_\epsilon') + \vec{u}_\epsilon^0 + \Delta t\vec{f}') \end{aligned}$$

where

$$\begin{aligned} \vec{r_\alpha} &= [1-\delta_\epsilon]\vec{r} + \delta_\epsilon\frac{\alpha U\hat{n} + (1-\alpha)\vec{U} - \vec{u}_\epsilon^0}{\Delta t} \\ \vec{\mathcal{L}}_\alpha(\vec{u}_\epsilon') &= \alpha\delta_\epsilon[\mathbf{I}-\mathbf{N}]\left(\frac{1}{c}\frac{\partial\vec{u}_\epsilon}{\partial n} - \vec{u}_\epsilon\right) \end{aligned}$$

which recovers the no-slip or Neumann conditions by setting α appropriately. Also note that unlike the no-penetration meta-equation of the previous section, using a Neumann (or Robin) slip model fully deactivates the fluid equations across the surface. This allows no direct communication between the flows on either side of the surface, the correct physical behavior. This deactivation also allows the velocity field to model sharp discontinuities across the surface. This may initially seem at odds with a smoothed immersion approach, but it is not. The blending width ϵ must still be resolved by the background grid allowing the fluid equations to transition smoothly into the body equations. However, the body equations themselves allow a discontinuous solution as shown by equation 4.9. Because the fluid equations are

fully deactivated on the surface this discontinuity does not lead to problems in the calculations of the fluid terms, unlike in the modeling of a sharp free surface.

4.1.4 Inversion of the $\mathcal{L}$ operator

The algorithm above requires four inversions per time step; two for the pressure and two for the velocity. However, as shown in this section, inversion of the $\mathcal{L}$ operator is trivial. First, consider the fitted-grid case, setting $\epsilon = 0$. The Neumann conditions do not effect the computation of the velocity in the field, those values are updated exclusively with the momentum equation. Once that is done, the Neumann boundary equations are satisfied by setting the boundary values of the tangential velocity appropriately. If the grid is locally normal-tangential this is quite simple, but generally it requires interpolation, coordinate rotation, and algebraic equation inversions.

The velocity inversions in algorithms 4.16 and 4.17 are a generalization of this exact process. In the fluid domain $\delta_\epsilon = \mathcal{L} = 0$ meaning that the velocity components are given directly by the RHS. The values in the smoothing region are set based on the field values in the fluid and the body governing equation. The flow of information is entirely from the fluid to the surface, making the inversion simple.

Conceptually, the equivalent linear matrix operator $[\mathbf{I} + \mathbf{L}]\{u\} \doteq \vec{u} + \vec{\mathcal{L}}(\vec{u})$ could be inverted in two steps. First order the equations by the value of δ_ϵ from smallest to largest, thus starting with the fluid points and working deeper into the smoothing region. The second and final step is direct substitution for the solution since the unidirectional information flow results in a completely lower-triangular matrix.

However, explicit construction of the matrix $\mathbf{L}$ and equation ordering are still more computationally expensive than required. A simple iterative method can be used instead, such as a Jacobi-like inversion

$$\begin{aligned} \texttt{given} \quad & [\mathbf{I} + \mathbf{L}]\{u\} = \{r\} \\ \texttt{let} \quad & \{u\}^0 = \{r\} \end{aligned}$$

$$\texttt{then loop} \quad \{u\}^n = \{r\} - \mathbf{L}\{u\}^{n-1},$$

such that the operator never need be constructed explicitly.

This method is validated with a toy example problem. Given an immersed circle of diameter D and a flow field defined by

$$\begin{aligned} U_r &= 1 \\ U_\theta &= \begin{cases} 1 \forall r < D/2 \\ -1 \forall r > D/2 \end{cases} \end{aligned}$$

such that the tangential component is discontinuous across the surface. Plugging this into the left-hand-side of 4.12 results in a smoothed version of this field

$$\vec{u}_\epsilon = [1 - \delta_\epsilon]\vec{U} + \delta_\epsilon(U_r\hat{r} - \vec{U}).$$

Figure 4-1 depicts the streamlines of these velocity fields, the discontinuous U in figure (a) and the blended U_ϵ in figure (b). A contour plot of the distance function $d = r - R$ from $-\epsilon < d < \epsilon$ is shown in the background for $\epsilon = D/8$. The sharp $\pi/4$ direction change in the streamlines across the surface in figure (a) has been greatly smoothed in figure (b).

To test the simple Jacobi-like inversion method we apply it to equation

$$\vec{u}_\epsilon + \delta_\epsilon\vec{\mathcal{L}}(\vec{u}_\epsilon) = \vec{u}_\epsilon$$

and compare $\vec{u}_\epsilon^n$ to the exact solution $\vec{U}$. Figure (c) shows the streamline results for $n = 2$ which are already much closer to the sharp corners of figure(a). Figure(d) and table 4.1 show the field values and error norms as a function of iteration number. The background grid did not line up values exactly with $x = 0$, so the values along $x/D = 0.035$ are given which is close enough to a radial line to illustrate the inversion process. The figure shows the values of $\vec{u}_\epsilon^n$ and $\delta_\epsilon\vec{\mathcal{L}}(\vec{u}_\epsilon^n)$ for $n = 0, 1, 2, 3, 4$ along with $\vec{U}$ and $\vec{u}_\epsilon - \vec{U} = \delta_\epsilon\vec{\mathcal{L}}(\vec{U})$. The figure and table both show clear second-order

n	L_1	L_2	L_∞
0	0.201E-01	0.776E-03	0.976E+00
1	0.102E-01	0.494E-03	0.841E+00
2	0.364E-02	0.225E-03	0.536E+00
3	0.600E-03	0.469E-04	0.171E+00
4	0.150E-03	0.109E-04	0.308E-01

Table 4.1: First second and ∞ norms of the error $\vec{u}_\epsilon - \vec{U}$ as a function of iteration number n.

convergence to the exact discontinuous solution with iteration number which validates the inversion method. In the remainder of the thesis this method is used with $n < 10$ such that the total inversion cost is proportional to the number of points in the smoothing region making it inconsequential compared to the pressure inversion cost.

4.1.5 Oscillating Cylinder Test

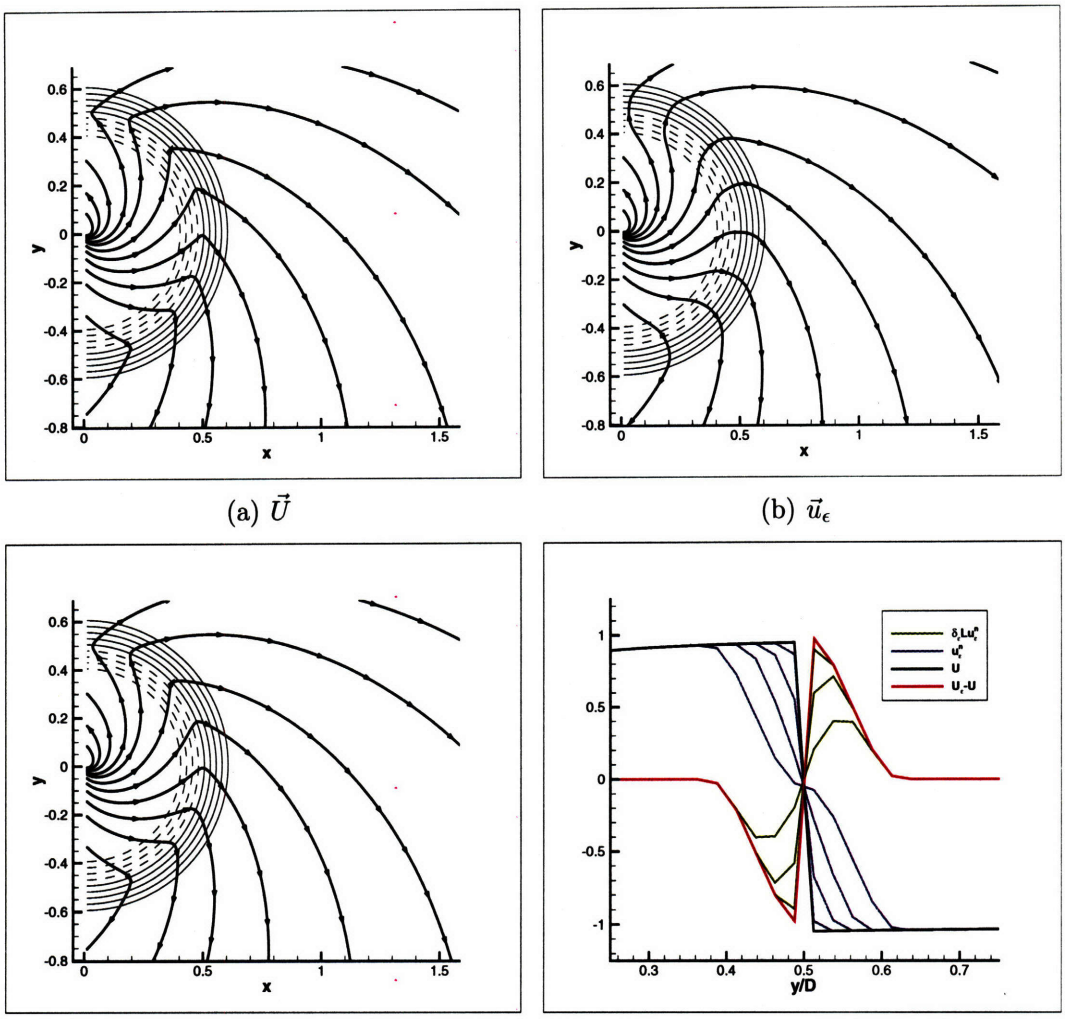
This sections determines the qualitative effect of the boundary equations on the solution for the test problem of a vertically oscillating cylinder of diameter D. The center of the body traces out the path

$$y_c(t) = -A\cos(2\pi t/T)$$

where $A = D/4$ and T is the characteristic period. Thus the maximum velocity is $U_{max} = \frac{\pi D}{2T}$ achieved when $y_c = 0$.

This system is simulated using a Cartesian background grid with $\Delta x = 0.025D$ and $\epsilon = D/8$. The meta-equations for the no-slip, no-penetration and Neumann BC are all tested. The fairly large smoothing width relative to the geometry accentuates the differences in the flows, as shown in figure 4-2. Figure (a) shows the vertical component of velocity along the $y = 0$ axis at $t = 3T/4$ for all the boundary treatments. Figure (b) shows level curves of the vorticity in the field at the same time instant.

Qualitative assessment of the condition's characteristics can be made from this figure. The no-slip boundary condition results in $v = -U_{max}$ on the body with a gradual transition to the 'exterior' flow velocity over the smoothing width ϵ. This



(a) $\vec{U}$ (b) $\vec{u}_\epsilon$

(c) $\vec{u}_\epsilon^2$ (d) Iterative values along x/D=0.035

Figure 4-1: Inversion of a sharp velocity field given a smooth source. Figures (a)-(c) plot the streamlines and $-\epsilon < d < \epsilon$ distance function level curves. Figure (a) shows the exact discontinuous solution U, figure (b) the smoothed field U_ϵ and (c) the solution $\vec{u}_\epsilon$ after two iterations. Figure (d) shows values of u_ϵ and $\delta_\epsilon \mathcal{L} u_\epsilon$ as iteration progresses.

numerical boundary layer is not physical, there is no viscosity in the flow, and may generally limit solution accuracy in the exterior. For instance, the maximum velocity value is located just outside the smoothing width and is slightly larger than U_{max}. If we let $\epsilon \rightarrow 0$ then the numerical wake defect would shrink, reducing the magnitude of this peak. Similarly, the maximum vorticity ridge in figure (b) would shift in towards the body.

While the internal flow is uniform in the no-slip case, this is not true for the no-penetration condition. As stated in section 4.1.2 the tangential momentum equations remain active across the body, allowing a confused flow to develop within the cylinder. The external flow in figure(a) appears reasonable with a reduced peak velocity shifted in towards the body but the vorticity plot shows multiple vortices have formed both within and without. Experience has shown this flow instability to be a general characteristic of the no-penetration meta-equation. This could be due to the unphysical modeling or numerical inaccuracy in the pressure inversions.

The Neumann condition offers a stable alternative slip-model. As shown, the velocity features a sharp discontinuity across the surface jumping from $v = -U_{max}$ to $v = 0.8U_{max}$ in one grid point. The width of this velocity peak is proportional to the smoothing width and the internal flow is again uniform. The vorticity plot shows the maximum ridge now lies along the surface. Figure (a) also shows that there is a negligible change in the solution when the LHS termed is modeled explicitly or dropped entirely in the pressure inversion. The term is ignored for the remainder of the thesis.

By varying the values of α in the Robin condition 4.15 further variations could be developed changing the maximum and width of the velocity peak. This modeling parameter could be put to use when data is available.

4.2 Compatibility of BDIM and VOF

The next step in appraising the Boundary Data Immersion Method is to test its compatibility with the Volume of Fluid method.

original X-axis column mirror. Fig 3-22 shows the location-dependency of the mirror non-flatness measurements does not disappear. We will proceed with the discovery process.

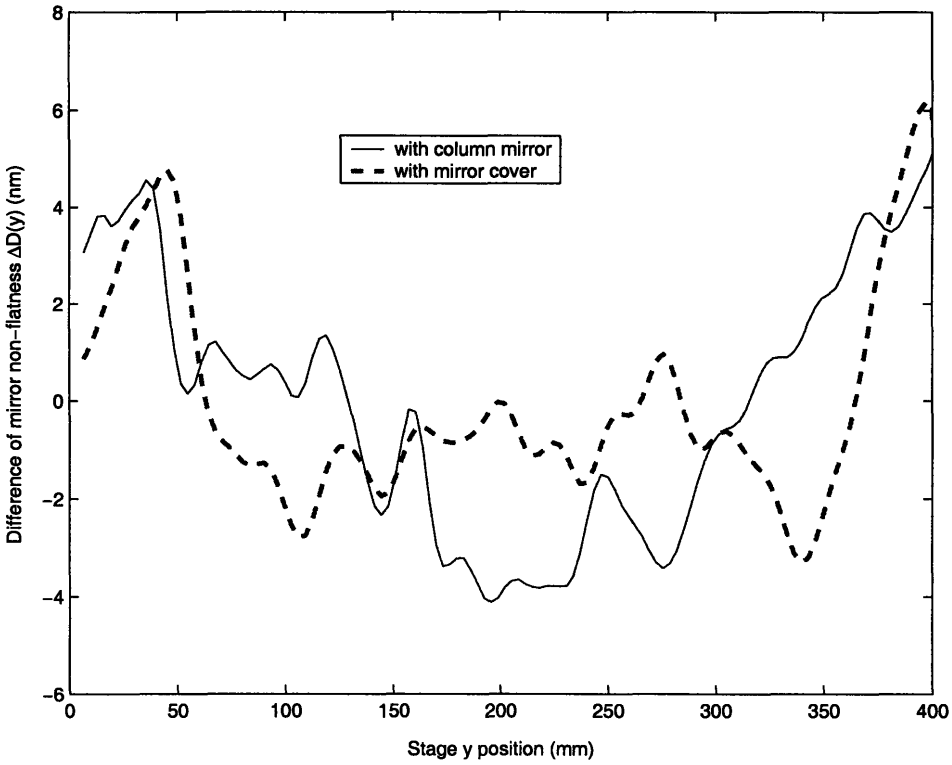


Figure 3-22: The solid line represents the difference of the X-axis stage mirror non-flatness measurement at x=150 mm with respect to the measurement at x=5 mm with the original X-axis column mirror. The dotted line represents the same kind of difference with a mirror cover fixed in the front of the X-axis stage interferometer head.

3.4.2 Thermal errors along the X-axis stage interferometer beam paths

In this subsection, we will discuss the possibility that the location-dependency of the mirror non-flatness measurements is due to the temperature gradients, which is one kind of thermal error, among the X-axis stage interferometer beam paths. In Chapter 2, we measured the temperature gradients in some critical planes inside the environmental enclosure of the Nanoruler. From the measurements we know

First off, how does the variable density field of a surface capturing method effect the meta-equation formulations of the previous section? Not at all: no properties of the density are assumed in the derivation of equation 4.12 from equations 4.1, 4.2 and 4.15. As discussed in chapter 2, if the density field is discontinuous across the free surface there are jump conditions that must be added to the momentum equation. However we are smoothing out the jump in density over a few grid points and regardless, that has nothing to do with the meta-equation approach.

The second question is how the presence of the immersed body effects the VOF surface capturing approach. Unlike in the fitted grid simulations of chapter 2 there are now regions of the background grid dominated by the solid body equations. The finite-volume cut-cell approach of [6] uses VOF and includes multiple correction terms to account for the presence of the body surface. The first correction in that work is to account for the change in the area through which volume can be advected due to the presence of the body. This is an important consideration in a sharp-interface cut-cell approach because velocity field discontinuity within a finite-volume caused by the body will otherwise render the field non-solenoidal. The second consideration is in the reconstruction step where only volume-fraction information from cells in the fluid domain are used in the reconstruction process.

Therefore we can see that the VOF method must be adjusted for a sharp solid/fluid interface to be maintained. This is intuitive because the presence of the body cuts the cell into two parts, one which may have one fluid type and one which may have the other. However, in our smoothed-body approach, the value of δ_ϵ gradually signals the transition to the solid equations and no hard line is ever crossed by the VOF method. The field f is advected by the smooth $\vec{u}_\epsilon$ velocity field and no jump conditions are applicable. The simulation do not respect sharp discontinuities but there are no errors above and beyond the second-order ϵ errors in the velocity field.

While jump adjustments are not applicable, it is possible to derive a meta-equation for the volume-fraction itself. However, this seems so far removed from the sharp-interface style of the VOF reconstruction and flux calculations that a meta-equation has not been pursued. This highlights the disparity between the sharp VOF method

and the smooth BDIM. To be philosophically consistent, one should use a cut-cell approach with VOF and a level-set approach with BDIM. In the sharp-approach all the jump conditions are relevant and interact. Typical methods of this type model the physics sharply but only to first-order near the interfaces as in reference [6]. The smooth-approach bypasses all of the small-scale 'accounting' difficulties presented in chapter 2 but replaces them with the idiosyncrasies of level-set methods. Never the less, such an approach has been used by Kelli Hendrickson at MIT to simulate the breaking wave over a submerged cylinder and is appealing enough to warrant future investigation.

Two simple tests are presented to demonstrate the ability of the Boundary Data Immersion Method to accurately simulate unsteady free-surface flows modeled by VOF. In both tests, a two-dimensional tank with aspect ratio 2 is simulated using $\Delta x = L/83$ and $\Delta t = 0.00267T$. The first case is a high amplitude standing wave test with $A = 0.2L$, $\lambda = L$ and wave period T. An image of the resulting nonlinear wave at time $t = T$ is shown in Figure 4-3(a). Note that the simulation is symmetric on either side of the $x = 0.5L$. To test the proposed method a vertical wall is placed at that location and a constant free surface height of $0.05L$ is set on the right side while the wave initial condition is maintained on the left. This simulation thus tests the method's ability to enforce the no-flux condition for f through the wall and quantify the errors in the free surface location due to smoothing the velocity field. The result using the current method is shown in Figure 4-3(b) where the smoothing width has been set to $\epsilon = 0.02L$. For comparison the result for the same simulation using the body force method is shown in Figure 4-3(c). Clearly, the body force method cannot possibly model this unsteady free-slip flow but the method fails to even respect the no-flux condition. To quantify the difference Table 4.2 lists the amount of dark fluid fluxed through the wall and the wave hieght at the wall at $t = 0.9T$ just before the impact in the unsplit tank case. Unlike the body force method, no fluid is transmitted through the wall, and errors in the wave height due to smoothing are very small. The body force method achieves a maximum height of only $0.46L$ failing to impinge on the lid at all. As mentioned above, the VOF methods are not aware of the body and

Boundary Type	Dark fluid fluxed	Wave height at t=0.9T
Symmetry Plane	0.0%	0.492L
BDIM	0.0%	0.487L
Body Force	4.6%	0.425L

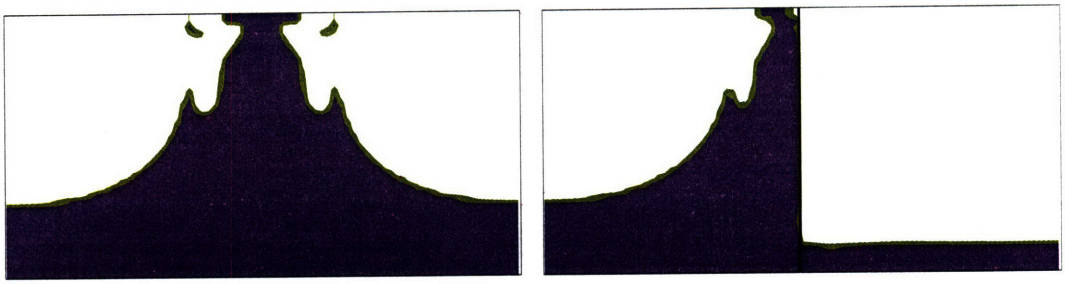
Table 4.2: Quantified results for the slip tank standing wave test.

this results in an $O(\epsilon^2)$ disturbance on the right side of the tank in figure(b). We can expect similar perturbation-level internal flows in all the free-surface simulations.

In the second case a tank half filled with water is quickly displaced to the right by $0.2L$ over T seconds and then held steady. This can be modeled using a frame of reference that moves with the tank, accounting for the acceleration by the application of a uniform body force. It can also be simulated by enforcing the no-penetration condition on the moving vertical walls of the tank using a stationary frame of reference. Therefore, the same result should be generated using a fitted-grid or moving immersed boundaries, allowing for a direct comparison. Figures 4-4(a) and 4-4(b) show the result for the fitted and immersed grid respectively at $t = 2.4T$. The images show that the comparison between the methods is excellent, even for this highly complex flow with dynamic boundaries.

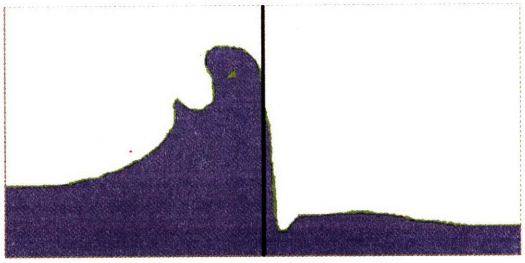
4.3 Derivative Informed Kernel Force Integration

The problem of measuring the fluid forces on the immersed body is next considered. In many engineering applications, this measurement of forces is one of the primary concerns; the drag force on a hull, the loading on a breakwater, etc. While the stated goals of this thesis (efficiently modeling nonlinear breaking bow waves) are not directly concerned with force calculation is it worthwhile to visit the topic briefly because of the complications introduced by a smoothed immersion approach. The issues of pressure smoothing and integral evaluation are addressed and resolved with a new kernel integration method.



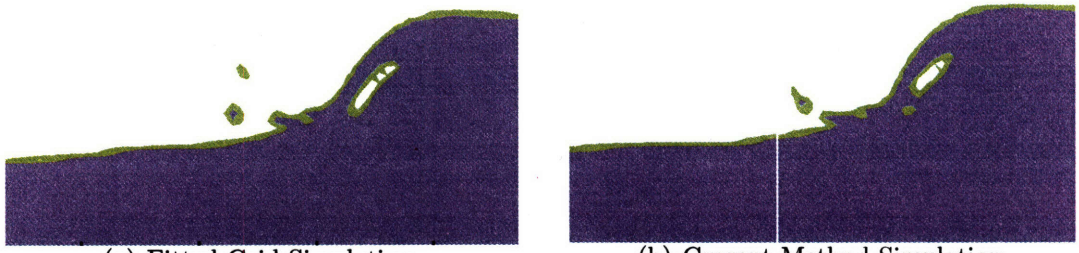
(a) Symmetric standing wave in full tank

(b) Standing wave in split tank using current method



(c) Standing wave in split tank using body force method

Figure 4-3: Images of high amplitude standing waves in a tank. Cells full of water are colored blue, cells full of air white, and partially filled cells green. Figure (a) shows the baseline case with no immersed surface. The domain has been split in two by a vertical wall in figures (b) and (c), using the current formulation and the body force formulation respectively.



(a) Fitted Grid Simulation

(b) Current Method Simulation

Figure 4-4: Image of sloshing waves in a tank generated by rapid sideways displacement using the same coloring as in Figure 4-3. Figure (a) shows the baseline using a fitted-grid formulation and Figure (b) shows the result using the current method.

The normal fluid force on the body is given by

$$\vec{f} = \oint_{\partial\Omega_b} p(\vec{x}_b)\hat{n}(\vec{x}_b)d\vec{x}_b \tag{4.18}$$

which is evaluted once the pressure p is determined by inversion of equation 4.14. In a boundary-fitted grid setting, force computation is a trivial post-processing step. The surface integral in equation 4.18 is simply numerically approximated with some quadrature

$$\vec{f} \doteq \sum_{\vec{x}_i \in \partial\Omega_b} p_i \hat{n}_i Q_i \tag{4.19}$$

where ϕ_i is the value of the field at computational point i.

When using a boundary fitted grid, the pressure is available and well defined up to the points directly off the boundary but this is not the case for the BDIM. As discussed in section 4.1.1the effect of the pressure is slowly ramped out with the $[1-\delta_\epsilon]$ coefficient in equation 4.4 resulting in a non-physical pressure in the smoothing region and on the body. However, this problem is not serious, in fact section 4.1.4 has solved it already. That section demonstrates how a repeated applications of a Neumann boundary condition operator can be used to propagate the solution in the flow through the smoothing region. Based on equation 4.5, this is only slightly more expensive than updating the boundary values on a general boundary-fitted grid.

However, there is a second difficulty, that of surface representation. One of the conceptual advantages of BDIM is that the surface is represented implicitly on the background grid by the interpolation function δ_ϵ. As discussed more completely in section 5.2 this allows for arbitrary flexibility (and accuracy) in defining the surface geometry. The price of that ability is that the background points do not line up with the surface and the pressure integral 4.18 cannot be evaluated directly. Again, solving this problem is not overly complex: once the pressure has been obtained in the smoothing region the values can be interpolated to any desired point on Ω_b. One choice is be kernel interpolation, similar to in section 3.2.2, ie

$$p(\vec{x}_b) = \int_{\Omega_f} p(\vec{x}_f) K_\epsilon^+(\vec{x}_b, \vec{x}_f)\, \mathrm{d}\vec{x}_f \tag{4.20}$$

where K_ϵ^+ is a one-sided interpolation kernel. Is it important that the kernel be one-sided because, as discussed in section 4.1.3, Neumann conditions allow discontinuities to develop. Interrogating the value of the pressure on either side of the surface $\partial\Omega_b$ results in an averaging of this discontinuity. [2]

So a step-by step approach for computing the pressure force is

1. Make an explicit surface grid and precompute the quadrature coefficients.
2. Solve for the pressure force at the current time step.
3. Update the pressure field through the smoothing region.
4. Interpolate the pressure values onto the explicit grid.
5. Perform the numerical force integration.

This process is significantly more complicated than in the fitted-grid case, but the efficiency of using background-grids in step 2 more than makes up for the effort. Still, it is interesting to note that three of these steps are concerned with getting (good) pressure values at explicit surface points, but the final desired result is an integral value, not a surface field.

A more direct and computationally efficient method is possible which uses an implicit geometry representation as in the flow solver. Begin by looking at steps 3 and 4: The pressure values in the smoothing region are determined using the pressure outside the smoothing region and the one-sided Neumann condition, then the values are interpolated with a one-sided kernel onto the surface. These steps can be combined by designing a kernel to propagate the field onto the body using the Neumann condition. In 1D we have

$$\begin{aligned} \phi(x) &= \phi(x') + (x - x') \left.\frac{d\phi}{dx}\right|_{x'} + O(2) \\ \int \phi(x) K_\epsilon^+(x, x')\, \mathrm{d}x &= \int \left[\phi(x') + (x - x') \left.\frac{d\phi}{dx}\right|_{x'}\right] K_\epsilon^+(x, x')\, \mathrm{d}x \end{aligned}$$

[2]Note that this kind of averaging is exactly what we want to do to the governing equations in section 3.2.2 but it is counter-productive here.

$$\phi(x') = \int \left[\phi(x) - (x - x') \left. \frac{d\phi}{dx} \right|_{x'} \right] K_\epsilon^+(x, x') \, \mathrm{d}x$$

where the first line is the second order Taylor expansion of a function ϕ, the second integrates that with a one-sided kernel K_ϵ^+ and the third solves for $\phi(x')$ given that the zeroth kernel moment is unity. Note that the derivative term integrates to zero for a symmetric kernel because the first moment is zero. A positive valued one-sided kernel cannot have a zero first moment but the derivative information maintains the second-order accuracy. Therefore we call this a one-sided Derivative Informed Kernel (DIK).

The equivalent DIK for the pressure is

$$p(\vec{x}') = \int_{\Omega_f} \left[p(\vec{x}_f) - (\vec{x}_f - \vec{x}') \cdot \vec{\nabla} p(\vec{x}') \right] K_\epsilon^+(\vec{x}_f, \vec{x}') \, \mathrm{d}\vec{x}_f. \tag{4.21}$$

Next this equation is substituted into equation 4.18 to give

$$\vec{f} = \int_{\Omega_f} \oint_{\partial\Omega_b} \left[p(\vec{x}_f) - (\vec{x}_f - \vec{x}_b) \cdot \vec{\nabla} p(\vec{x}_b) \right] \hat{n}(\vec{x}_b) K_\epsilon^+(\vec{x}_f, \vec{x}_b) \, \mathrm{d}\vec{x}_b \, \mathrm{d}\vec{x}_f \tag{4.22}$$

where the order of integration has been switched. With the goal of taking the arguments out of the surface integral, imagine letting the tangential support of the kernel approach zero while the normal support remains ϵ. As in section 3.2.3, this allows us to evaluate the functions of $\vec{x}_b$ only at the point on the surface nearest to the fluid point, given by $\vec{S}(\vec{x}_f)$. As the kernel is nonzero only on the edge of the smoothing region we can expand the integral domain and substitute the pressure BC to give

$$\vec{f} = \int_\Omega \left[p - d\hat{n} \cdot \left(\vec{r} - \frac{\vec{U} - \vec{u}_0}{\Delta t} \right) \right] \hat{n} \delta_\epsilon^+ \, \mathrm{d}\vec{x} \tag{4.23}$$

where $d\hat{n}$ is the vector from $\vec{S}$ to $\vec{x}$ and and $\delta_\epsilon^+(\vec{x}) \equiv \oint_{\Omega_b} K_\epsilon^+(\vec{x}, \vec{x}_b) \, \mathrm{d}\vec{x}_b$. Equation 4.23 can be directly evaluated with our background grid method and gives the normal pressure force on the body. This method can be applied without updating the pressure in the smoothing region, but the kernel must only sample from points $d > \epsilon$ from the

body. Also the $\vec{S}$ function can be adjusted if the force is only desired on one (or part of one) surface out of many in the domain.

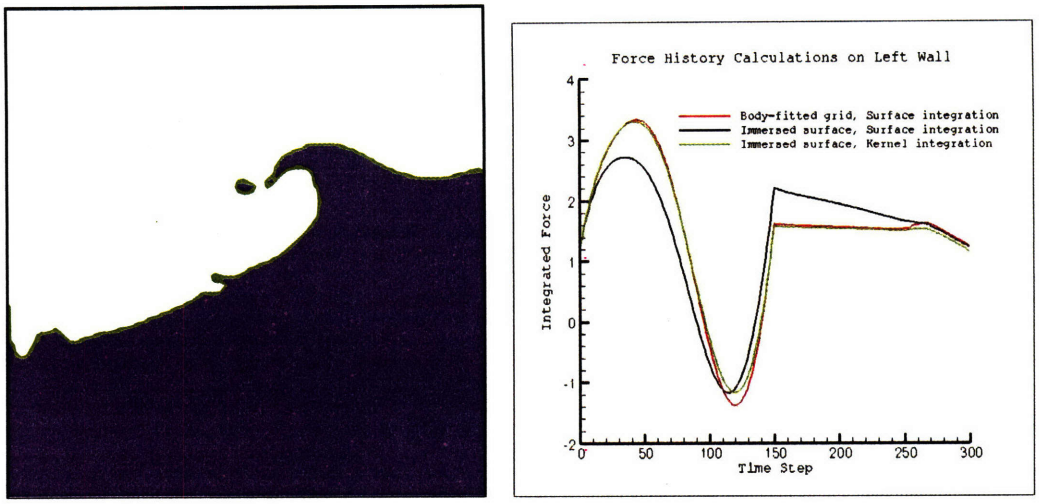
There are restrictions on this approach. The force is computed without ever determining explicit values on $\partial\Omega_b$, therefore if surface values are required (for higher moments, structural loads, etc) then equation 4.23 must be modified. As in section 3.2.3, the form of δ_ϵ^+ can be approximated without performing the body integration, but because we are not integrating over the normal direction a $1/\epsilon$ term remains, ie

$$\delta_\epsilon^+(d) \approx \frac{\delta_\epsilon^S(d - d^+)}{\epsilon[1 + d\kappa]} \tag{4.24}$$

where d^+ is the kernel offset from the surface and $\kappa = \frac{\partial n}{\partial \sigma} + \frac{\partial n}{\partial \tau}$ is the surface curvature. Like the derivative information in the pressure equation, these curvature terms come from a Taylor expansion and integrate to zero if the first kernel moment is zero. As such the error in the DIK formulation is second-order with respect to d^+ times the gradients (of the surface and the pressure field) and is not appropriate for surfaces with sharp corners.

The form of equation 4.23 might seem to imply that the linear time derivative $\frac{\partial U}{\partial t}$ is the only important factor in computing the pressure force. Clearly, this is not true; in pseudo-steady ship flow applications the nonlinear convective force is far more important. However, the convective flux through a solid surface is zero, which is why $\frac{\partial U}{\partial t}$ is the only term in the pressure boundary condition. Therefore, the linear term is the only one needed in equation 4.23 to correct the pressure in the smoothing region.

An extreme sloshing wave test has been used to verify the ability of this approach to quantify fluid forces. The displaced tank case of section 4.2 is run again but with a tank aspect ratio of 1 such that the wave does not impact the tank lid. The simulation is run twice, once using a trivial fitted-grid with an applied horizontal acceleration force and again with moving immersed boundaries. Figure 4-5(a) shows the resulting nonlinear breaking wave using the same coloring scheme as in the previous section. Figure 4-5(b) presents the resulting integrated forces on the left wall of the tank. The results for the body-fitted grid are in red and can be considered the base-line



(a) Snapshot during breaking

(b) Measured left wall forces

Figure 4-5: Repeat of the sloshing waves in a tank generated by rapid sideways displacement. The aspect ratio of the tank is now unity. Figure (a) shows a snapshot of the flow during breaking. Figure (b) shows the integrated forced using the surface-grid and DIK methods.

exact solution. Note that the pressure is dominated by the horizontal acceleration and a small bump at the end of the run indicates when the breaking wave in figure(a) impacts the left wall. The black line shows the result of integrating the pressures in the smoothing region without first updating the pressure with the boundary condition. As discussed above, these values of pressure are non-physical and the resulting errors are on the order of 10%. The green line shows the result using the DIK integration of equation 4.23. The pressure in the smoothing region is not updated and the kernel offset d^+ from equation 4.24 is set to $3/2\epsilon$. The resulting pressure calculations for the BDIM body using the DIK kernel are excellent both for positive and negative pressure gradients. The difference between then base-line and the DIK values is less than 1.5%. This is not surprising; section 4.2 has already shown that the BDIM simulation captures all the same unsteady effects as the fitted-grid simulation. It is only because the surface pressures are not defined that we have to be careful in how we integrate the forces and the DIK method allows this to be done easily.

4.4 Nonreflecting Exit Boundaries

Exit conditions are required for external incompressible flow simulations and are notoriously difficult to enforce accurately. In practice poor exit conditions require large computational domains and short simulation times to avoid corrupting the solution. This reduces the computational efficiency and reliability of the simulation. In this section a new exit boundary condition is presented within the BDIM framework which minimizes reflections for general fluid systems.

Unlike the free surface or a no-slip wall, an "exit" is a purely numerical boundary. The ideally semi-infinite domain of an external flow is truncated by the exit boundary to make the solution tractable. In establishing an exit boundary, we are inherently assuming that the flow phenomena of interest are within the numerical domain and independent of the flow beyond the exit. Exit conditions must be set despite this assumption of independence because the pressure Poisson equation is elliptic and requires information on every boundary to ensure a globally divergence-free velocity field.

One method of modeling an exit is to construct a numerical beach to absorb the energy out of exiting waves. Such methods are often ineffective and require large buffer regions, lowering computational efficiency. In [7] a body force method is developed to drive the flow at the exit to match the free stream conditions, and [12] suggests using 1D extrapolation to set the velocity profile at the exit. While these methods can be applied in limited cases they rely on a strong underlying convective flow to wash away the errors resulting from their modeling inaccuracies.

A more advanced unsteady condition is a wave exit, such as developed by [41]. The typical Orlanski-type boundary condition applied to the velocity vector is

$$\left(\frac{\partial}{\partial t} + c\frac{\partial}{\partial x}\right)\vec{u} = 0 \tag{4.25}$$

where c is the, as yet undetermined, wave speed and the x-coordinate represents the direction normal to the exit. When using this condition, the mass flux through the boundary is prescribed and a compatible Neumann condition must be set on the

pressure. Reasonable choices of the wave speed c are the free-stream velocity or some other natural velocity metric, such as $\sqrt{gL}$. In [24] exit speeds are determined by solving equation 4.25 for c one point interior to the exit. However, this method is unstable if the momentum equations are integrated explicitly.

Regardless of how the wave speed is chosen, if there are multiple waves with differing speeds exiting the domain, equation 4.25 produces reflections for waves with speeds not equal to c. The Higdon condition [23] has been developed to circumvent this difficulty. The condition is a wave product

$$H_J: \prod_{j=1}^{J} \left(\frac{\partial}{\partial n} + \frac{1}{c_j} \frac{\partial}{\partial t} \right) \vec{u} = 0 \tag{4.26}$$

where c_j are the set of exiting wave speeds. Increasing the number of wave speeds reduces the energy reflected back into the domain. Derivation and implementation of this conditions for the two-dimensional wave equation is given by [18], but applying this boundary condition to incompressible free-surface flows is nontrivial.

Additionally, a velocity boundary condition that determines the mass flux through the exit must be globally compatible with the other boundaries. If the flux through the other boundaries are prescribed (i.e., the pressure conditions on those surfaces are Neumann) then regardless of the choice of wave speed (or speeds) the calculated velocities must be adjusted to achieve conservation. In other words, global conservation of mass requires either an ad-hoc adjustment to the exiting velocity field or a Dirichlet condition to be set on the exit. Therefore neither equation 4.25 nor equation 4.26 can be applied to the exiting component of the velocity in their current form.

When the flux is given on all the other domain boundaries the standard solution is to integrate those mass fluxes and make an *ad-hoc* adjustment to the exit velocity field such that global conservation is maintained. Formally, this adjustment to the x-component of the velocity vector can be written as

$$\left(\frac{\partial}{\partial t} + c \frac{\partial}{\partial x} \right) u = -\alpha \tag{4.27}$$

where α is some, hopefully small, deviation from a true exiting wave form. This is only possible if the mass fluxes on all other boundaries can be readily determined *a priori*, which is not true for general cases.

To achieve a generally applicable governing equation for the exit we correlate the terms above to those of the x-component of the momentum equation

$$\left(\frac{\partial}{\partial t} + \vec{u} \cdot \vec{\nabla}\right) u = -\frac{1}{\rho}\frac{\partial p}{\partial x}. \tag{4.28}$$

The time derivatives match and $c\frac{\partial u}{\partial x}$ is a one-dimensional model of the convective acceleration. The remaining term is the pressure force, which leads us to formulate a boundary condition of the form

$$\mathcal{B}_x : \left(\frac{\partial}{\partial t} + c\frac{\partial}{\partial x}\right) u = -\frac{1}{\rho}\frac{\partial p}{\partial x} \tag{4.29}$$

for the exiting component of velocity. Adjusting the standard Orlanski exit in this way allows the pressure field to fulfill its normal role of projecting the velocity onto a solenoidal field. Therefore, no global flux integrations are required and the system conserves mass locally and globally to the tolerance of the pressure solver.

When using equation 4.29 a Dirichlet boundary condition must be set on the pressure equation. Again, this is because the solver must be free to determine the exit pressure gradient and resultant fluid flux at the exit boundary. If the pressure gradient is set in a Nuemann condition, equation 4.29 reduces to 4.27 (with $\alpha = \frac{1}{\rho}\frac{\partial p}{\partial x}$) and the pressure equation can not ensure global mass conservation. One option is to set the dynamic pressure to zero. This is especially attractive when using the upper domain boundary as an exit (an open lid) because the hydrostatic pressure can also be set to zero on the upper boundary. However, when dealing with a closed lid a wave condition for the exit plane pressure

$$\left(\frac{\partial}{\partial t} + c\frac{\partial}{\partial x}\right) p = 0 \tag{4.30}$$

is the most consistent option. The boundary condition is applied at the exit using

linear interpolation

$$\overline{P}_e = \frac{1}{2}\left(P_E + P_I\right) \tag{4.31}$$

where $\overline{P}_e$ is the interpolated exit value and P_E and P_I are the cell-centered values to the exterior and interior of the exit respectively. Second-order central differences and Runge-Kutta integration in time give the boundary equations as

$$\overline{P}_e^{n+1/2} = \overline{P}_e^n - \frac{c\Delta t}{\Delta x}\left(P_E^n - P_I^n\right) \tag{4.32}$$

in the predictor step and

$$\begin{aligned} \overline{P}_e^{n+1} \quad &= \tfrac{1}{2}\Big[\overline{P}_e^n + \overline{P}_e^{n+1/2} \\ &\quad -\tfrac{c\Delta t}{\Delta x}\left(P_E^{n+1/2} - P_I^{n+1/2}\right)\Big] \end{aligned} \tag{4.33}$$

in the corrector step. These equations are substituted into the pressure equation to eliminate references to the exterior pressure value. While modifying the pressure equation at the exit in this manner does entail some additional complexity it is the only method presented which can be used to model any general free-surface flow.

To quantify the performance of these exit condition a series of tests are run using two-dimensional Cauchy-Poisson waves. The domain has an aspect ratio of 2 and is discretized using $\Delta x/L = 0.04$ and $\Delta t/T = 0.02$. The exit conditions are set on the far right boundary at $x = 4L$ and free-slip conditions are set on all other boundaries. The initial free-surface elevation is set to $\eta(x)/L = 1 + \frac{1}{2}\exp(-\pi x/L)$. To measure the error induced by the exit, a reference solution is generated by doubling the length of the numerical domain and using the standard Orlanski exit with $c = 1.5L/T$ at $x = 8L$. Figure 4-6 shows the initial condition from $x = 0 \ldots 4L$ along with subsequent snap-shots in a waterfall plot for this reference solution. In this figure, time progresses by moving down the page, and the waves travel from left to right. Four nonlinear waves of decreasing amplitude, wavelength and speed can be easily identified in the figure. The figure also shows that there is little reflection into the domain during this time period justifying our use of this solution to estimate errors

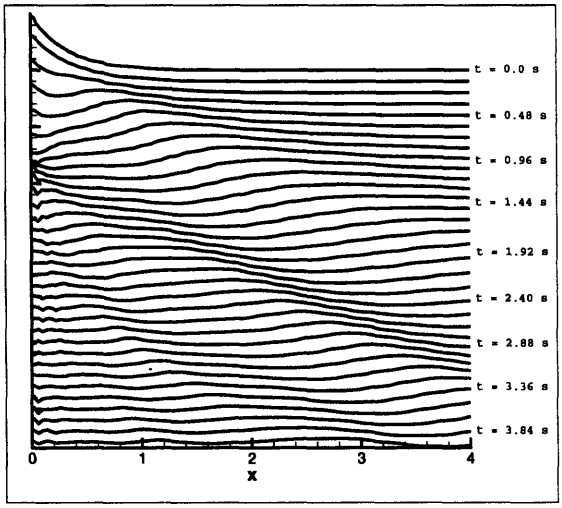
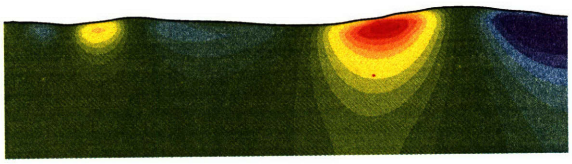


Figure 4-6: Free surface elevation waterfall plot of the reference solution. Each line is $6\Delta t = 0.12T$ later than the line above it. The view is truncated at the $x = 4L$ where the tested numerical exit conditions are enforced.

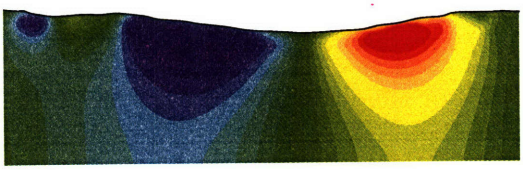
in the simulations on the truncated domain.

Four types of exiting conditions are tested, the first of which is a baseline case using a free-slip wall condition. This condition gives pure reflection. The second exit condition uses linear extrapolation to determine the velocity profile at the exit. The u component is corrected in by global flux integration to ensure mass conservation. The third condition is the standard single-speed Orlanski exit described by (4.25) using a global integration adjustment to determine α. The last condition is the pressure adjusted wave exit using (4.29) and (4.30). The wave speed is set to $c = 1.5L/T$ for all the wave cases. Figure 4-7 shows the results from all of these tests along with the reference solution as the second wave reaches $x = 4L$ at $t = 3.36T$. Contours of $\frac{\partial p}{\partial x}$ are plotted within the water to aid in comparing the methods. By this time, the large first wave has reflected back across the domain in Figure 4-7(b) and the comparison to the reference solution is already poor. The extrapolation exit condition is observed to give essentially the same solution as the reflection exit. The two wave exits compare much better to the reference solution.

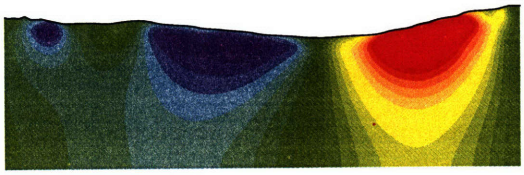
To establish a quantitative comparison, the L_2 norm of the error in the free surface elevation in the truncated domain is calculated as a function of time. The results are shown in Figure 4-8 for each exit condition and for a range of c values. The error



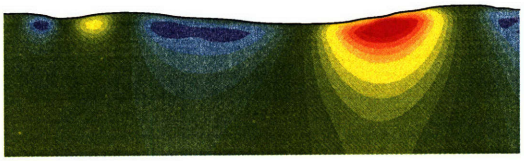
(a) Reference solution



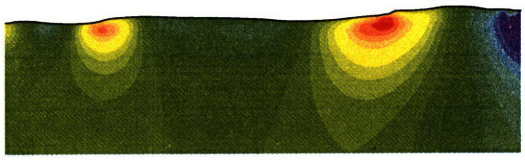
(b) Reflection condition



(c) Extrapolation exit



(d) Wave exit with global-integration adjustment



(e) Wave exit with pressure adjustment

Figure 4-7: Contours of $\frac{\partial p}{\partial x}$ in the water for the exit conditions at $t = 3.36T$. A wave speed of $c = 1.5T/L$ is used for all wave cases.

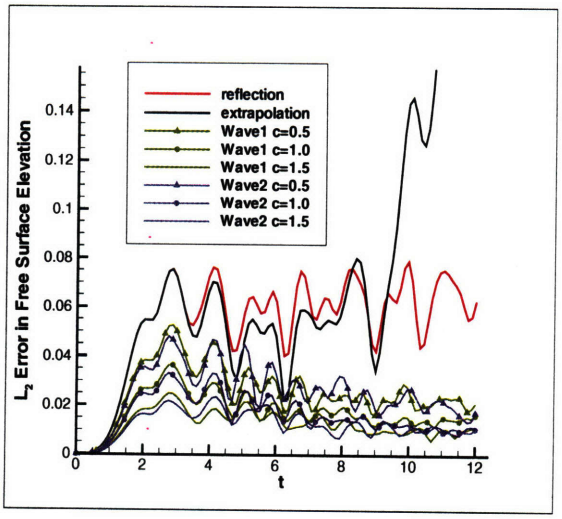


Figure 4-8: Measurements of the L_2 spacial norm of free surface error as a function of time. The wave exit with global-integration correction are labeled Wave 1, and the pressure corrected exits are labeled Wave 2.

in the pure reflection condition is seen to fluctuate in time as the waves reflect from one end of the truncated domain to the other. The Euler equations generate no physical dissipation to damp out the wave energy yet the magnitude of error remains well bounded in time. This is in contrast to the extrapolation exit which has similar characteristics to the reflection boundary initially but rapidly becomes unstable and diverges completely at $t = 9T$. This instability in the extrapolation exit demonstrates that spurious information generated at the exit has fully corrupted the solution.

The next six lines of Figure 4-8 show the wave exit with global-integration correction (labeled Wave 1) and pressure correction (labeled Wave2) using wave speeds of $c = 0.5, 1.0$ and $1.5L/T$. All six wave exits show a significant decrease in error and a fast overall trend towards zero error in time. By $t = 10T$ nearly all waves have traveled out of the truncated domain, and the remaining error is likely indicative of reflections in the reference solution. For all choices of wave speed, we see that the pressure corrected exit has comparable errors to the exit corrected by global-integration. As equation 4.29 is more physically relevant than equation 4.27 and allows comprehensive treatment of free-surface flows, we conclude it to be a significant advancement.

However, the measurements shown in figure 4-8 still show reflections which could corrupt solutions. The Higdon condition may be used to extend the method to

multiple wave speeds but solution algorithms for such methods are complicated, particularly for variable density flows. Simplifying these numerical methods is a topic of current research.

4.5 BDIM Validation Test Cases

This section presents unsteady 2D and 3D free-surface flows with immersed bodies to validate the inclusion of the Boundary Data Immersion Method into the fluid solver.

4.5.1 Unsteady 2D wave-maker

The first test case is that of a vertical wall 2D wave-maker. Lin studied this case in his 1984 paper [32] and PhD thesis [33] with experimental tests and a boundary-element potential-flow method. These results provide the data against which the BDIM solutions are validated.

A sample test from that thesis is a wave-maker which follows a simple harmonic motion, such that the horizontal velocity is of the form

$$U_s(t) = \frac{\omega A}{2} \sin(\omega t) \tag{4.34}$$

$$U_c(t) = \frac{\omega A}{2} \cos(\omega t) \tag{4.35}$$

where U_s is referred to as 'sine-motion', U_c is 'cosine-motion', ω is the frequency, and A is the double-amplitude of the wave-maker. Note that the cosine motion is impulsively started.

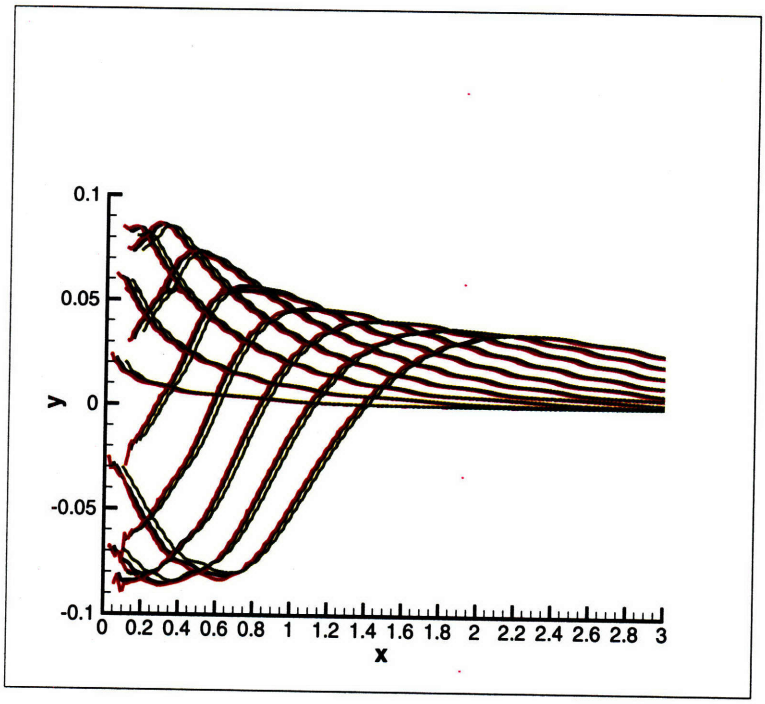
The results are shown in figure 4-9 and in all plots the free surface location is shown along the length of the tank for a number of time steps. This is the manner in which the data is presented in [33] and the time scales always match up with that reference. Figure (a) shows the results for smoothing width $\epsilon/h = 0.1, 0.05, 0.025$ for the $A/h = 0.1$ sine-motion case. The variation is small and limited to the near wall region of the flow. A grid convergence study was also preformed and the results indicate second order convergence in the L_2 norm of the free surface elevation for the

same non-breaking $A/h = 0.1$ test case. Figure (b) shows the comparison for a challenging high-amplitude impulsively started run; cosine-motion and $A/h = 0.25$. The bulk flow result compares well, but the potential flow simulation displays premature breaking. This could be due to details of the impulsive start-up or the instability issues mentioned in that section of the 84 thesis. Figure 4-10 (a) and (b) show the results for a high-amplitude case $A/h = 0.25$ with sine-motion. The comparison is excellent, with the height and phase of the bulk wave correct as well as the inception time and type of breaking. Again, this is likely due to the slightly easier task of matching test conditions without an impulsive start.

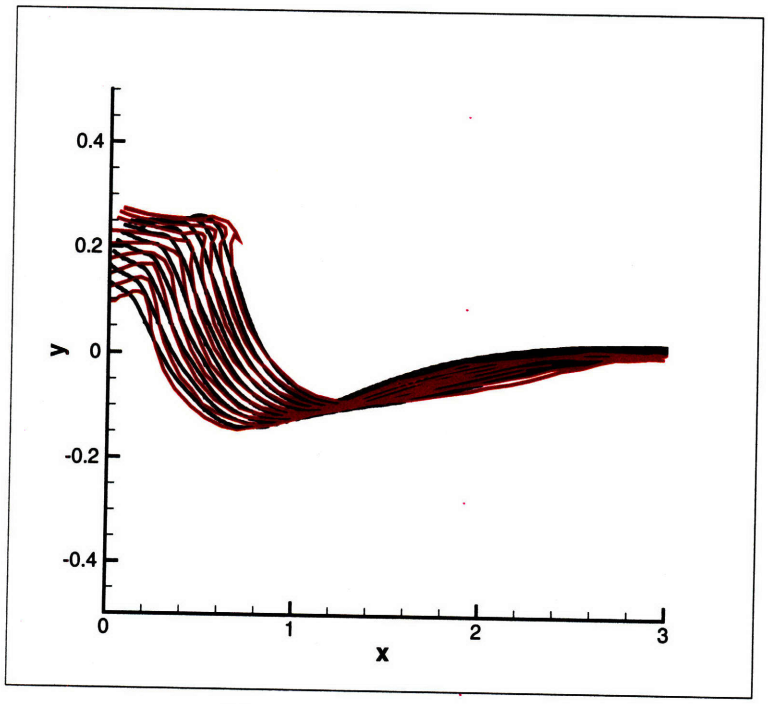
4.5.2 3D Semi-Submerged Sphere

Next the results are presented from a 3D semi-submerged sphere flow. The physical test consists of a sphere, initially at rest half submerged in the water. The sphere is rapidly accelerated up to a full speed Froude number of 2. The results for the slip-model BDIM formulation are compared to the cut-cell numerical results of Dommermuth. The cut-cell has an advantage in resolution, using 3 million grid points and featuring grid clustering in the domain. The BDIM solution uses 0.8 million grid points and no clustering but the comparison is excellent even for this reduced resolution. In the BDIM method, the flux-corrected Orlanski exit is used with a wave speed of the free stream velocity, ie $c = U$. The domain size is 3x1.5x3 diameters with a symmetry condition on the $y = 0$ plane. The smoothing width is set to $\epsilon/\Delta x \approx 5$ to ensure maximum accuracy.

The snapshot results just as the sphere reaches full speed are shown in figure 4-11. Figures (a) and (b) show the 3D wave elevations and sphere geometry for the cut-cell and BDIM solutions respectively. Here the sphere's are traveling towards the viewer and to the left. Clearly, despite the highly unsteady nature of this impulsive, bluff-body flow, the qualitative comparison is surprisingly good. Figure (c) allows a more quantitative assessment, showing the unsteady water-line along the sphere. Here, the sphere's are moving to the right, the blue line is the BDIM water-line and the red line is the cut-cell water-line. The comparison is still excellent, with matching run-up



(a) ϵ-study, sine-motion A=0.1



(b) cosine-motion A=0.25

Figure 4-9: Free surface elevation results for the vertical harmonic wave-maker. Figure (a) shows the dependance on ϵ for a sine-motion case. Figure (b) compares the current results to the published results for A=0.25 using cosine-motion.

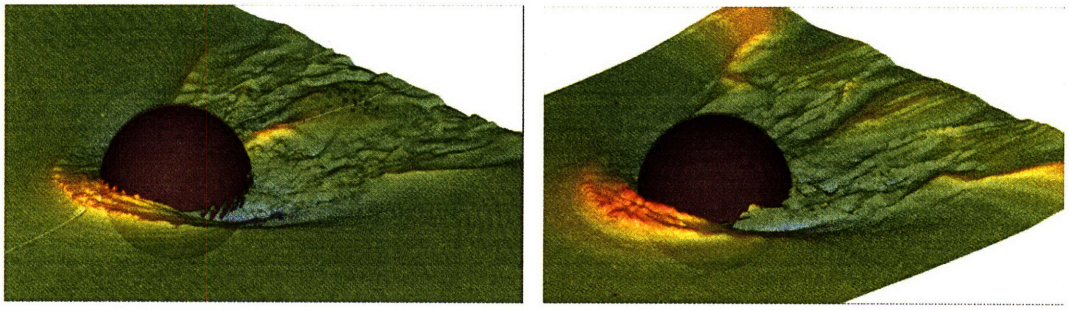
alignment grating lines (θ), which is also the angle difference between the interference fringes and the alignment grating lines, is calculated by,

$$\theta = \frac{\phi_B - \phi_A}{2\pi K} p \tag{4.6}$$

where $\phi_B - \Phi_A$ is the phase variation between points A and B on the grating (the measurement procedure has been discussed at the beginning of Section 4.3), p is the period of the alignment grating, and K is the distance between points A and B. Since the scan direction (θ_f) is known, the angle of the alignment grating with respect to the Y axis (α) can be obtained by

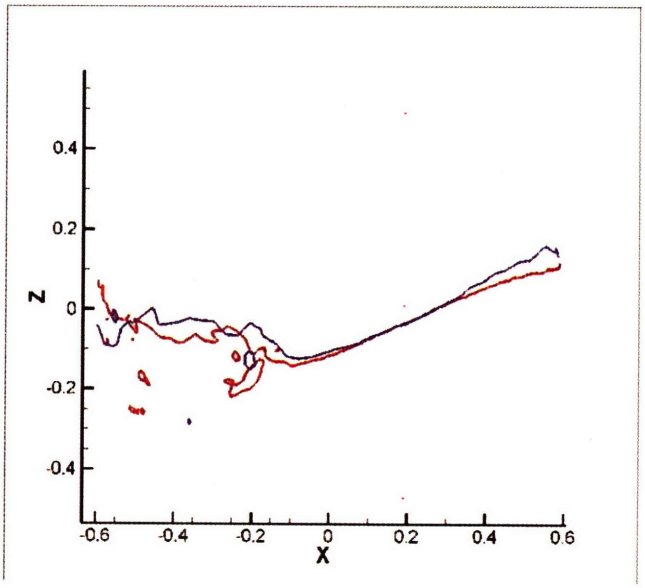
$$\alpha = \theta_f - \theta \tag{4.7}$$

Next we will discuss the magnitude of the angle measurement error and how large an overlay position error it will cause. In Equation 4.6 we approximate the actual value of p as the period by which we exposed the alignment grating. The period measurement variation of the grating is around 2 pm from day to day, which corresponds to less than 4 ppm for 574 nm-pitch grating. Since the angle difference between the alignment grating lines and the interference fringes, which is θ, is around $10 \sim 30$ μrad, thus the angle measurement error due to the period approximation is less than 4 ppm$\times 30$ $\mu rad = 0.00012$ μrad. Another source of the angle measurement error is the phase measurement errors at points A and B. As we will discuss in Section 4.3.3, the phase measurement error at a single point is around 0.007 rad for 3σ when a 30-sec averaging filter is utilized. Thus the angle measurement error due to the phase measurement errors is $\sqrt{2} \times 0.007\ rad \times p/2\pi K = 0.028$ μrad when the scanning distance $K = 30$ mm and $p = 574$ nm. Thus the angle measurement error of the alignment grating is about 0.03 μrad. The corresponding overlay position error due to the alignment grating angle measurement error is 0.03 $\mu rad \times L = 0.9$ nm, where $L = 30$ mm is the size of the overlay area.



(a) cut-cell, 3 million points

(b) BDIM, 0.8 million points



(c) water-line profiles

Figure 4-11: Free surface elevation results for the semi-submerged sphere external flow test case. Figure (a) and (b) are the cut-cell and BDIM results. Figure (c) compares the (blue) BDIM and (red) cut-cell water-line profiles.

heights on the right and similar bubbly wake characteristics on the left.

In conclusion, this chapter has demonstrated the applicability of the Boundary Data Immersion Method to general two and three-dimensional fluid flows. The method has been found compatible with free surface capturing methods and moving immersed solids. The Derivative Informed Kernel method was developed and allows fluid forces to be computed easily and accurately on immersed bodies. BDIM is verified and validated through numerical convergence tests and direct comparison to previous computational results for free-surface flows.

Chapter 5

Investigation into the Slender-Ship Model of Divergent Ship Waves: 2D+T

2D+T models simplify the analysis of ship bow waves by assuming that changes in the longitudinal direction are small compared to changes in the transverse directions. This chapter investigates the use of Cartesian-grid numerical methods to simulate an experimental wavemaker approximation to this flow as well as the ideal 2D+T flow and the fully 3D system. Comparisons are made between the experimental and numerical results for the wavemaker, 2D+T and 3D tests.

5.1 Introduction

Many important physical effects are generated based on breaking waves near the bow of a surface ship. Slamming and impact, spray generation, air entrainment and bubbly flows. However, the three dimensional nonlinear free-surface flow associated with high-speed ships is difficult to analyze numerically because the characteristic length scales for the problem typically run over seven orders of magnitude [65]; from small-scale turbulence and contact line dynamics up to transverse waves in the wake of the vessel. This disparity makes direct simulation of full-scale flows impossible, even

using modern computational resources. In light of this issue, simplified modeling of the physics must be done in order for tractable predictions to be made, and '2D+T' is one such model.

In the 2D+T model for divergent waves the three-dimensional bow flow is approximated by a two-dimensional, time evolving flow with fully nonlinear boundary conditions satisfied on the free surface and the hull surface. The roots of this method in slender body theory are first presented and the basic mathematical analysis is derived. Next the previous 2D+T studies and the current work is outlined.

5.1.1 Slender-body Theory Background

Slender body approximations have a long history of application to ship flows, dating back to Michell's integral for prediction of ship's resistance in 1898 [36]. In that work, the beam is assumed much smaller than the draft or the length, and the resulting flow predictions scaled linearly with the beam. This theory was later applied to airfoils with great success (lift-surface theory), but is not as well suited to ships because they are not sufficiently thin.

A successful ship application came with 'strip' theory, first developed by Korvin-Kroukovsky and Jacobs in 1957 [28] and further developed in the sixties. In this theory, the forces due to the cross-flow induced by incident waves are computed independently on cross-sections of a long slender ship. The interaction of the cross-sections and other nonlinear effects (such as steep waves and large motions) are completely ignored, but the theory has been used successfully for over 50 years.

However, both thin-ship and strip theory make assumptions which restrict their predictions to simple linear flows. Less severe assumptions can be made for slender vessels which allows nonlinear waves to be studied. One such approach is the '2D+T' model of ship flows. In 2D+T theory the flow is assumed to be parabolic in the axial direction, i.e. that changes in the longitudinal direction are small compared to changes in the transverse directions [60]. This enables modeling of the bow waves around slender high-speed vessels to be generated as an unsteady two-dimensional system instead of a steady three-dimensional system. The 2D+T approximation was

first introduced in aerodynamics by M. Munk [38] for the prediction of loads on slender bodies with low-aspect-ratio wings at small angles of attack and reference [14] provides a thorough history of the method.

5.1.2 Mathematical Framework of 2D+T Modeling

The ship geometry is characterized by the length L, beam B and draft h. Thus the non-dimensional geometry parameters are defined as the entrance angle $\tan\alpha = B/L$ and the draft aspect ratio $\gamma = h/L$. The 2D+T method is based off of slender body assumptions, ie

$$\tan\alpha \simeq \gamma \ll 1. \tag{5.1}$$

Historically, this assumption was used in the framework of potential flow, allowing the introduction a scaled laplace equation for the velocity potential ϕ close to the body

$$\left(\gamma^2\frac{\partial^2}{\partial\hat{x}^2} + \frac{\partial^2}{\partial\hat{y}^2} + \frac{\partial^2}{\partial\hat{z}^2}\right)\phi = 0 \tag{5.2}$$

where $\hat{x} = x/L$, $\hat{y} = y/h$, $\hat{z} = z/h$. Therefore, if the vessel is slender the equation has no x-dependance to leading order. This is the fundamental simplification, reducing the 3D spacial dependance to a 2D spacial dependance.

The dynamic free-surface boundary conditions is also simplified, and the first order term takes the form

$$\left[\frac{\partial^2\phi}{\partial\hat{x}^2} + \frac{1}{2}\frac{\tan\alpha}{\gamma}\left(\left(\frac{\partial^2\phi}{\partial\hat{y}^2}\right)^2 + \left(\frac{\partial^2\phi}{\partial\hat{z}^2}\right)^2\right)\right] = -\frac{\hat{\eta}}{\hat{F}_L^2} \tag{5.3}$$

This equation imposes a constant downstream x-dependance on the solutions, but no upstream influence [15]. Therefore, the x-dependance is more directly expressed as a time dependance, using the simple mapping

$$x = Ut. \tag{5.4}$$

This is the parabolic flow assumption, allowing the steady 3D flow to be modeled as

a time-varying 2D flow.

The length based Froude number is the standard parameter for characterizing the gravity waves generated by a 3D ship

$$F_L = U/\sqrt{gL} \tag{5.5}$$

where, U is the forward speed and g is the acceleration of gravity. Note that the length based Froude number cannot be directly applied to this unsteady 2D flow because L and U are both x-direction scales. Instead scales from y, z, t are required. The beam and draft are the new length scales and the time scale is simply

$$T = L/U \tag{5.6}$$

the period required for the flow to travel the length of the ship. Thus an appropriate non-dimensional parameter for the unsteady 2D flow is

$$F_{2D} = \frac{1}{T}\sqrt{\frac{b}{g}}. \tag{5.7}$$

When the 2D+T assumptions are made, this becomes the only relevant non-dimensional scale describing the gravity waves. This 2D scale is related to the standard Froude number by

$$F_L = \frac{F_{2D}}{\sqrt{\tan\alpha}}. \tag{5.8}$$

The slender body assumption (equation 5.1) thus requires that $F_L >> F_{2D}$.

5.1.3 Applications and Limitations of 2D+T Models

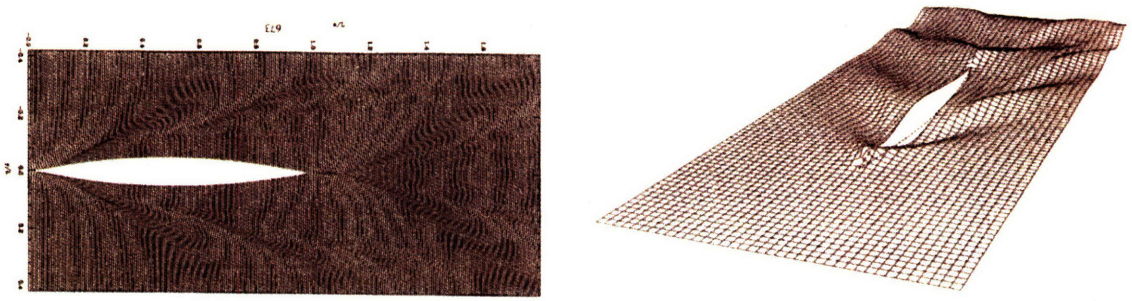
As shown in the previous section a 2D+T model of ship waves may be developed but application of this model is very restricted in scope. The vessel must satisfy the slenderness assumption of equation 5.1. In addition, application of 2D+T models to vessels which have local geometric features such as a blunt bow or appendages is questionable.

A second limitation is imposed by assuming the longitudinal (x-component) velocity is equal everywhere to U. This means that 2D+T simulations cannot model the transverse waves generated by a ship. Additionally the longitudinal momentum deficit created by the viscous near-wall effects are ignored. Thus 2D+T models can give little to no indication of the forward drag of a displacement ship.

Also note that the 2D+T flow is insensitive to proportional scalings of U and L. This is made clear by examining the non-dimensional parameters developed above. The 2D Froude number F_{2D} is only dependent on the time scale $T = L/U$. Therefore, taking a geometry and doubling its length (halving $\tan\alpha$) as well as the forward speed does not effect T, F_{2D}, or the 2D+T solution. On the other hand this doubling of the length and speed does change the length based Froude number, scaling it up by $\sqrt{2}$. Therefore 2D+T modeling of the ship flow is only valid in the limit of very large Froude numbers and forward speeds.

These restrictions have shaped the previous research in 2D+T modeling. The first investigations are of high-speed planing flows and high-speed slender displacement hull flows [59]. Tulin and Wu [60] used the 2D+T technique for simulating the high-speed flow around a Wigley hull using a nonlinear 2D potential-flow code. They compared the calculation with the 3D fully nonlinear potential-flow RAPID, created by Hoyte Daven. Figure 5-1 shows a comparison of the wave pattern computed by the 2D+T code and the three-dimensional code. The same individual divergent waves at the bow and stern can be seen in both calculations, and the origin and extent of these waves match well. A prominent rooster tail behind the stern and the resulting diverging waves are also found in both calculations.

The Froude numbers in this case were limited to $F_L < 0.46$ which may be sufficient for the thin and smooth Wigley hull but not 'much greater than one' as required by theory. However, the divergent bow waves created by a vessel with a fine entrance angle are not as sensitive to the length based parameters, as shown in [15]. Therefore 2D+T modeling of slender vessel bow waves does not require the speeds to be as extreme, but there are problems in this region as well. Because it only allows down stream influence the 2D+T approach cannot model the waves created in front of the



(a) (b)

Figure 5-1: A comparison between (a) 2D+T: $L/B = L/D = 10$; $F_L = 0.30$, and (b) exact nonlinear (RAPID) wave prediction for a wigley hull. The RAPID calculation is by Hoyte Raven of MARIN. From Tulin and Wu (1996).

ship bow. This further limits the simulations to ships with very fine bow entrance angles.

While 2D+T theory has existed in the literature for a number of years and had its fair share of proponents and opponents, the fundamental question of its usefulness in the prediction of practical ship flows has yet to be established. This is partially because the assumptions of potential flow theory limit its usefulness in the analysis of steep and overturning ship waves. Breaking waves are not irrotational and must be modeled using the momentum equations instead of a Laplace equation for the velocity potential. Additionally most potential flow methods require laborious human assistance to compute solutions after breaking takes place, required for any 2D+T simulations of a ship with an overturning bow wave.

Yet the slender body assumptions used on the potential flow equations above can also be applied to the Euler equations (allowing for rotational flow) or the complete Navier-Stokes equations or even to an experimental system. The parabolic nature of the flow is maintained along with the resulting simplification of the 3D wave field problem into a 2D one. One such study which has recently come our attention is Andrillon 2004 [1]. This short paper presents simulations of the 2D+T Wigley flow for three Froude numbers using a VOF scheme for the free surface and a boundary fitted grid to model the ship hull. The VOF scheme is not conservative, the resolution

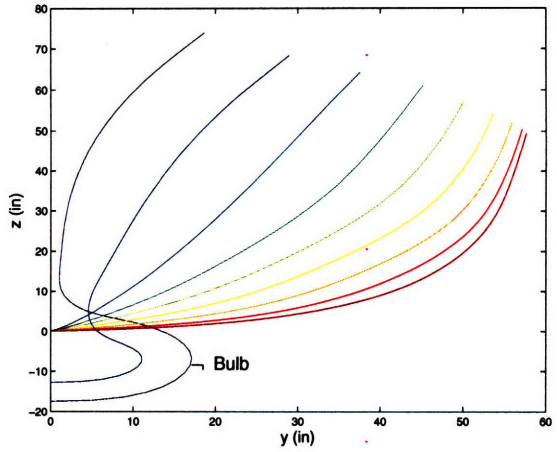


Figure 5-2: Vertical cross sections of the model 5415 hull form at 9 equally spaced intervals from stem to midship.

is poor and the analysis is limited to 2D, but the attempt is still encouraging.

An experimental application of the 2D+T concept has recently been completed by Mostafa and Duncan [53]. In this study a large flexible wavemaker, driven by hydraulic pistons is used to push out the 2D+T representation of the 5415 ship hull. A series of profiles of one side of the 5415 hull at different streamwise stations from stem to mid-ship are plotted in figure 5-2. In the experimental 2D+T technique, these profiles represent the shape of the wavemaker geometry at various times and are shown in figure 5-3. The advantage of performing 2D experiments instead of simply towing a 3D model of the ship is that of scale. In a typical experiment, the wave maker generates a wave nearly a half meter tall. Note that for this study the bulb (sonar dome) of the of the model (Figure 5-2) has been removed from the representation of the hull due to limitations in the wave maker experimental apparatus and the fact that it is not well represented by slender ship approximation. This shows that while these experimental results bypass the limitations of potential flow they introduce limitations of their own.

These new studies offer a fresh look at 2D+T modeling but do not answer the question of the model's applicability to ship bow flows. Mostly this is due to a lack of direct correspondence between the 2D+T and 3D flows, either from modeling or geometric restrictions. Even when available, calculations from fully three-dimensional

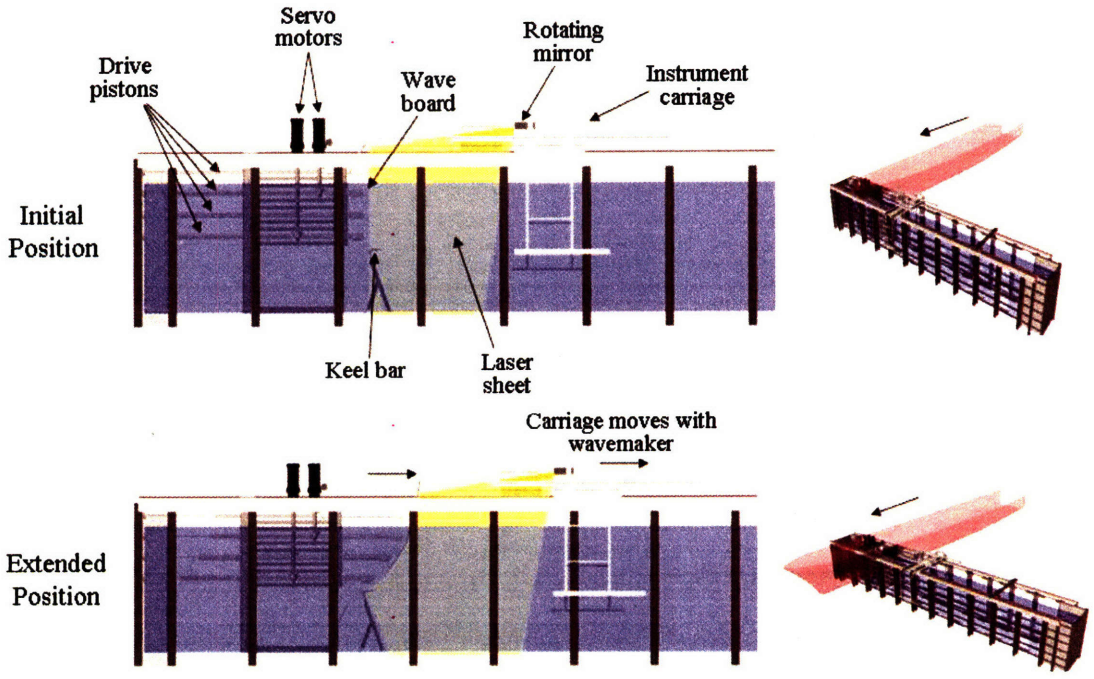


Figure 5-3: Schematic drawings showing the tank, 2D+T wave maker and instrument carriage. The tank is shown in the two drawings on the right in perspective view with an imaginary ship hull in two positions as it moves through the tank. Corresponding side views of the tank are shown on the left with the wave board at the instantaneous shape corresponding the to position of the imaginary ship hull in the drawings on the right. The tank is 14.8 m long, 1.15 m wide and 2.2 m deep. The water depth is 1.83 m. Taken from Shakeri [53]

methods and 2D+T methods are difficult to compare since the methods typically use different levels of approximation in computing the free surface boundary conditions, the turbulent flow and air entrainment. However, the new active-boundary and free-boundary immersion methods described in the preceding chapters offer the unique capability to quantitatively determine the bounds on the validity of 2D+T models for bow flows with breaking waves. Before that study can be undertaken a general method of representing non-analytic 2D and 3D geometries is required. This topic is covered in the next section, after which the 2D+T results are presented.

5.2 Extension of BDIM to Nontrivial and 2D+T Body Geometries

For the test geometries in chapters 3 and 4 (circles and lines) determining the geometric properties and boundary value fields δ_ϵ, $\vec{U}$, and $\hat{n}$ are obvious and trivial. However, the complex geometries involved in 3D and 2D+T ship flow simulations require a general method of defining the body and determining these functions. This section details how these fields are computed based on a NURBS-defined surface, allowing a rich set of geometries to be naturally immersed in the fluid simulation.

5.2.1 Mathematical Surface Representation

All flows with bodies must describe the geometry in some way and the most useful description is problem dependent. When solving biological fluid-dynamics problems with the Immersed-Boundary method, a simple algebraic mesh representation is logical because the elastic body equations need to be solved on a mesh. However, when solving flows with the active-boundary immersion methods of the previous chapters a more powerful set of functions can be chosen to describe the geometry.

The surface $\partial\Omega_b$ is an $(\mathcal{N}-1)$ dimensional submanifold of the $\mathcal{N}$ dimensional Euclidean space. To explicitly define this surface requires a set of functions of the

form

$$\vec{X} = \vec{X}(\vec{s}) \tag{5.9}$$

transforming a value of the surface coordinate vector $\vec{s} = \{\sigma, (\tau)\}$ into a Euclidean space vector $\vec{X} = \{x, y, (z)\}$. It is required that the functions implied by equation 5.9 be single valued but they may be many-to-one, ie different values of $\vec{s}$ may result in the same $\vec{X}$ location. If the functions are differentiable, the infinitesimal distances in space are given by the chain rule as

$$d\vec{X} = \frac{\partial \vec{X}}{\partial \vec{s}} \cdot d\vec{s} \tag{5.10}$$

These functions may take on virtually any form, and in fluid mechanics the standard representation for surfaces is a structured or unstructured mesh. This representation has the advantage of simplicity, being an array of control point locations connected by straight lines, but it also has many disadvantages. Most obvious is that the generally continually varying geometry is represented by piecewise planar surfaces, but other problems are noted below.

NURBS (Non-Uniform Rational B-Splines) are one of the most popular tools used to represent lines and surfaces in computer aided design and graphics, and are the backbone of such programs as Rhino and FastShip. This is because of their efficient implementation, their intuitive control point interface, and the smoothness properties of the resulting forms. In this work, the NURBS description used to design a solid body is maintained in the computational analysis of the flow around that body. This eliminates all need for gridding and assures the designer that WYSIWYG [1]. Another consideration is that NURBS surfaces have far fewer parameters than algebraic grids making them better suited for shape optimization problems. Details on the analytic form and computational evaluation of NURBS can be found in many texts such as [44], but are not relevant here.

[1] What You See Is What You Get

5.2.2 Correlation Function Determination

Regardless of the method used to describe the surface, certain features of the topology must be known in order to perform a fluid dynamic simulation. When using a structured fitted-grid the ends of the grid array always lie on the boundary, making the correlation trivial. For unstructured or over-set methods, there is a list which tells each fluid cell who its spacial neighbors are and if it is in contact with a boundary face.

In the current background-grid method a correlation between the point $\vec{x}$ and the nearest point on the surface $\partial\Omega_b$ is established with the function $\vec{S}$, first discussed in section 3.2.3. Defining the vector from any Euclidean point $\vec{x}$ to any surface point $\vec{X}$ as

$$\vec{v}(\vec{x},\vec{s}) \equiv \vec{X}(\vec{s}) - \vec{x} \tag{5.11}$$

allows the squared minimum distance function to be defined as

$$\mathrm{d}^2(\vec{x}) = \min_{\vec{s}} \left(|v(\vec{x},\vec{s})|^2\right) = \left|v\left(\vec{x},\vec{S}(\vec{x})\right)\right|^2 \tag{5.12}$$

and therefore

$$\vec{S}(\vec{x}) = \arg\min_{\vec{s}} \left|\vec{X}(\vec{s}) - \vec{x}\right|^2. \tag{5.13}$$

Equation 5.13 is a nonlinear minimization problem for determining the correlation function $\vec{S}$. In this work we use a Gauss-Newton method for non-linear least squares which has a second-order convergence rate and is described in any classic optimization text such as [13].

There is an advantage in defining the function $\vec{S}$; as well determining d, $\vec{S}$ establishes a general correlation between the fluid and the surface. *Any* quantity known on the surface is easily referenced at the Eulerian points using $\vec{S}$. One notable example beyond the grasp of the distance function is the tangential velocity of the surface; but heat flux, bending energy and other parameters could also be referenced off the surface for appropriate multi-physics simulations.

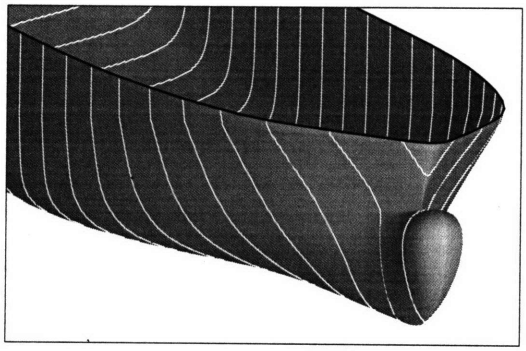


Figure 5-4: Bow of a general containership hull-form with bulb. The lines are cross sections of the surface with x-planes, as are required for 2D+T simulations.

5.2.3 Cross-Section Geometries

An additional advantage of a parametric representation is that the solution method above can be easily adjusted to compute distance function to geometric subsets of the surface. Cross sections are often useful when modeling a three-dimensional flow with a two-dimensional simulation. As described in the previous section, 2D+T simulations reduce the 3D fluid problem to a 2D cross-section plane which moves along the length of the ship in time. At time t the plane is located at $x_1 = Ut$ where U is the in-flow velocity. Figure 5-4 shows a containership bow with the cross sectional lines which are used in 2D+T simulations. Notice from figure 5-4 that the curve is discontinuous at the bulbous bow. The BDIM can handle this discontinuity with no special treatment and a similarly robust way of determining $\vec{S}$ is possible.

In fact, the only change in the formulation above is that we are not interested in the closest point on the surface, but the closest point which lies on the cross-sectional plane. In other words we solve equation 5.13 with the additional constraint that

$$X_1(\vec{s}) - Ut = v_1(\vec{s}, Ut) = 0. \tag{5.14}$$

where v_1 is the first component of the position vector from equation 5.11. Optimization with a nonlinear constraint is known as nonlinear programming, but the method of Lagrange multipliers recovers an unconstrained problem which can again be solved using the Gauss-Newton method. This is known as Sequential Quadratic

Programming ([13]) and maintains the same second-order convergence rate as the unconstrained Gauss-Newton.

5.2.4 Geometric Properties and Velocity Conditions

With the correlation function established it remains to determine the values of $\hat{n}$, κ and $\vec{U}$ for use in the meta-equations.

The Euclidian tangential vectors are defined from equation 5.10 as

$$\begin{aligned} \frac{\partial X_i}{\partial \sigma} &= X_{i,1} \\ \frac{\partial X_i}{\partial \tau} &= X_{i,2} \end{aligned}$$

and from them the normal is easily found

$$\vec{n} = \vec{X}_{,1} \times \vec{X}_{,2} \tag{5.15}$$

and $\hat{n} = \vec{n}/|n|$. In equation 4.24 we also require the mean surface curvature, and this is given by

$$\kappa \hat{n} = \nabla_s^2 \vec{X} = \vec{X}_{,jj} \tag{5.16}$$

which is also a second way of computing the normal for surfaces with nonzero curvature.

Obtaining the velocity of the surface defined by equation 5.9 is simply a matter of taking the time derivative of the position, $\vec{U} = \frac{\partial}{\partial t}\vec{X}$. However, determining the proper velocity boundary condition for a time-varying 2D+T cross section is more subtle. Taking the time derivative of the 2D+T constraint and using equation 5.10 gives

$$U = \dot{X}_1 = X_{1,i}\, \dot{s}_i \tag{5.17}$$

which relates the time rate of change in the surface coordinates to U, the speed of the cross section plane in the X_1 direction. If $X_{1,i} = 0$ then the surface is locally tangential to the x_1 plane and the equation has no solution. If either $X_{1,i} \neq 0$ equation 5.17

has infinitely many solutions. This is because the cross section is not itself defined by a parametric equation such as equation 5.9, but by the intersection of two surfaces. Therefore, there is no *mathematical* constraint on the tangential velocity of the curve, only the normal velocity.

In general therefore, the cross-section velocity can be given any orientation vector $\vec{m}$ lying in the x_1 plane, ie

$$(\dot{X}_2, \dot{X}_3) = \alpha(m_2, m_3). \tag{5.18}$$

Plugging the second and third component of equation 5.10 results in the system

$$\begin{bmatrix} X_{1,1} & X_{1,2} & 0 \\ X_{2,1} & X_{2,2} & -m_2 \\ X_{3,1} & X_{3,2} & -m_3 \end{bmatrix} \begin{pmatrix} \dot{s}_1 \\ \dot{s}_2 \\ \alpha \end{pmatrix} = \begin{pmatrix} U \\ 0 \\ 0 \end{pmatrix} \tag{5.19}$$

which can be inverted for α and plugged into equation 5.18 for the velocity boundary conditions. Note that the system defined by equation 5.19 is not diagonally dominant so complete inversion with pivoting is required.

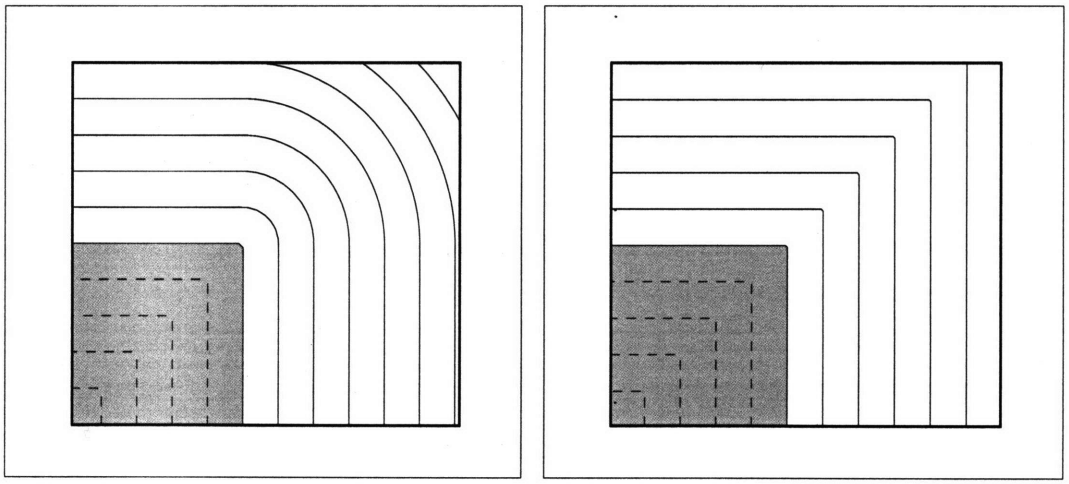
5.2.5 Distance Function and Composite Surfaces

The only remaining variable is the distance function d, for use in defining δ_ϵ. The most obvious way of specifying the distance d is taking the square root of equation 5.12, however this does not give sign information. The equation for the projected distance is

$$\tilde{d} = \hat{n} \cdot \vec{v} \tag{5.20}$$

which does results in the correct sign. However the value of $\tilde{d}$ at a given point is less than the true minimum distance to the surface if $\vec{v}$ does not lie on $\hat{n}$, occurring where κ is not defined. Figure 5-5 illustrates the difference between d and $\tilde{d}$ for a simple box geometry. The level curves of the true distance function are round at the corner while the projected distance level curves are sharp.

The correct representation is determined by the context; and in this case the



(a) d (b) $\tilde{d}$

Figure 5-5: The level curves of the distance function for a box geometry. Figure (a) shows the true distance to the geometry while figure (b) plots the projected distance.

distance function is used to define the interpolation function. As discussed in sections 3.2.3 and 4.3, the true definitions of δ_ϵ and δ_ϵ^+ are integrals of kernels over the body geometry. The distance-based relations of equations 3.14 and 4.24 are idealization which are not valid where the curvatures are extreme. Therefore, technically neither d or $\tilde{d}$ are correct at the corner.

To avoid this conflict we can think of geometries such as in figure 5-5 as composite surfaces. NURBS geometries are often designed in a composite manner and dealing with them is required in any event. Figure 5-6 depicts the distance function for a vertical (d_v) and horizontal (d_h) line, as well as a point (d_p). The two lines have zero curvature, therefore the interpolation function idealizations apply to those geometries, and their intersection defines the box geometry of 5-5. In other words

$$d_{box} = d_v \bigcap d_h = max(d_v, d_h) \, \forall \, \vec{x} \tag{5.21}$$

where the larger of d_h and d_v is taken at every point $\vec{x}$ to construct the intersection set. The resulting distance function in figure 5-6 (d) matches the projected distance function in figure 5-5(b). It may seem that by including the point distance function d_p, we recover figure 5-5(a), but in fact the result is $d_{box} = d_p$ and the $d = 0$ contour no longer corresponds to the box geometry. Furthermore, the concept of $\hat{n}$ and κ are

$R/\Delta x$	NURBS time	Mesh time	Mesh Error
32	0.301e-1	0.201e-1	0.119e-2
64	0.121e+0	0.660e+0	0.178e-3
128	0.181e+1	0.202e+2	0.453e-4
256	0.127e+2	—	—

Table 5.1: Distance calculation statistics for immersed sphere using mesh and NURBS based surface representations.

not defined for a point geometry. Thus the projected distance of equation 5.20 is the consistent choice for use in the interpolation functions.

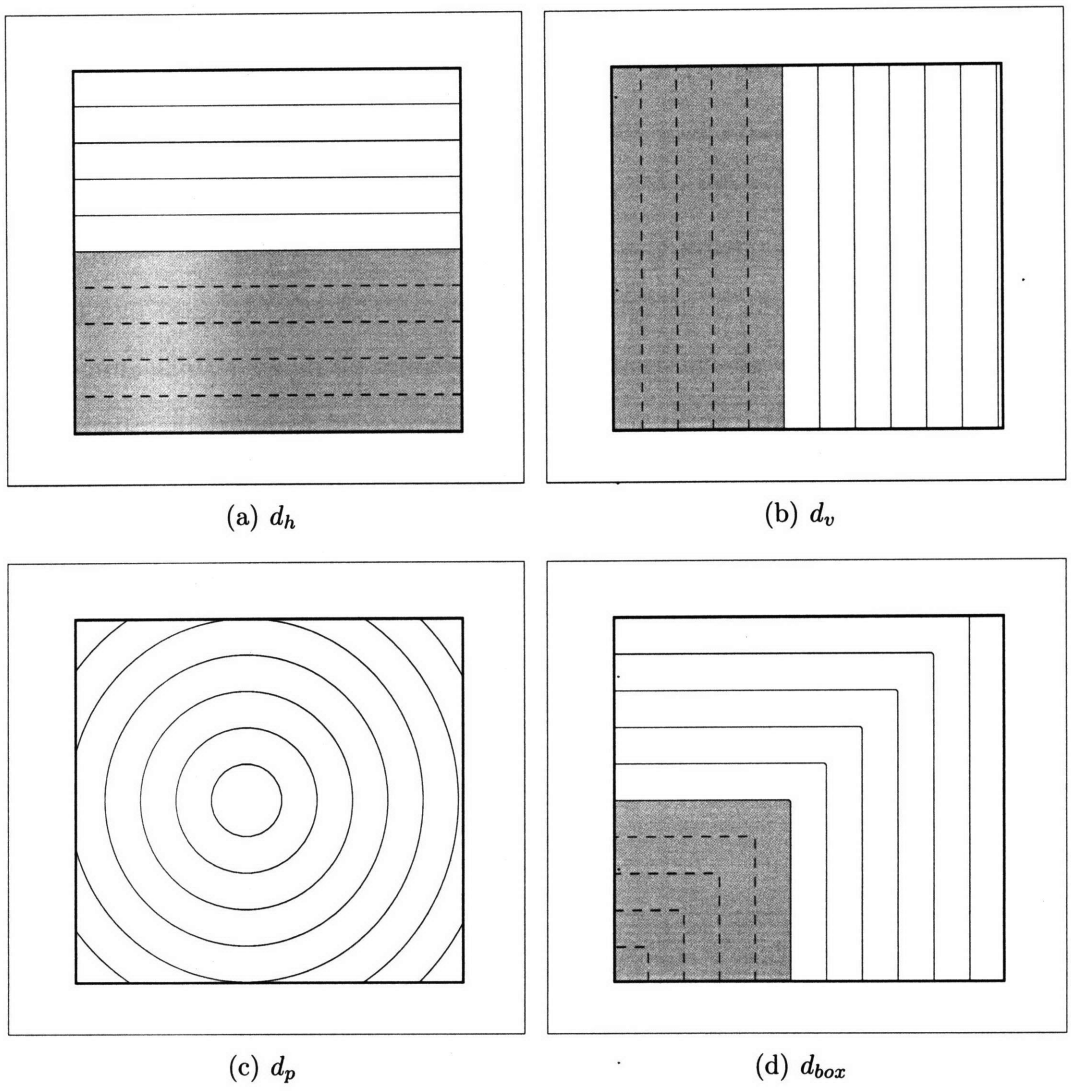
The union of two geometries is simply given by,

$$d_{union} = d_a \bigcup d_b = min(d_a, d_b) \forall \vec{x} \tag{5.22}$$

and is used for constructing the global distance function when multiple bodies are immersed in the simulation.

5.2.6 Sphere Validation Test

To quantify the speed-up obtained by solving equation 5.13 on a NURBS surface, the signed distance function and normal vector for a spherical test geometry are computed on a series of background grids. This is compared to the average accuracy and speed of calculation of the distance function and normal using a structured surface mesh. Table 5.1 shows these results. The distance function and normal vector are found using a single processor computer, and the times (given in seconds) are only meant for relative comparison. The value $R/\Delta x$ is the sphere radius over the grid spacing and therefore proportional to the number of grid points in one direction. The error tolerance in the parameters using the NURBS solver is set to be less than 1e-7. The mesh-based method has been stopped after 45 minutes on the finest grid and the results are not shown for this case. The table shows that the cost of determining the parameters scales linearly with the number of points when using the Gauss-Newton solver, but at least quadratically when using an exhaustive search on the mesh. This is especially important when the geometry is continually changing as the simulation



(a) d_h (b) d_v

(c) d_p (d) d_{box}

Figure 5-6: The level curves of the distance function for pieces of a composite geometry (a horizontal line(a) a vertical line (b) and a point (c)) and the composite box distance function(d).

progresses, such as in dynamic boundary problems or hull shape optimization.

5.2.7 2D+T Surface Generation

The final preparation step is to create a mathematical surface to reproduce the experimental tests performed at the University of Maryland. The geometry data is produced by applying an edge finding algorithm to high speed video of the unsteady wave-maker runs. This data set presents interesting challenges. First, there are multiple obstructions obscuring the view of the wave-maker plate location, resulting in large data-free zones in time and space. Second the edge finding method does not track material points on the plate, it merely finds as many points along edges as it can. After cleaning up spurious lines, the data ranged from 20-100 points per image over 70 images. And of course, the times the position data are available do not generally line up with the time steps of the simulation.

Given the special nature of the data set a NURBS surface is fit to the experimental measurements instead of creating a time-space mesh. The NURBS representation easily fills in the missing spacial-temporal features while ensuring smoothly-varying derivatives. It also allows the corresponding 2D+T 'hull' to be visualized and compared directly to the ideal 5415 hull surface.

A nonlinear least squares method is used to fit the NURBS control-points to the $\hat{X}(Ut, y, z)$ data. Defining the squared error as

$$\varepsilon_i^2 = \sum_j \left(X_i^j - \hat{X}_i^j\right)^2 \tag{5.23}$$

where i is the coordinate index, j is the data sample index. However, this metric begs the question: which point on the NURBS surface to compare to data point $\hat{X}^j$. Stating the problem another way, the NURBS surface value is

$$X_i^j = X_i(\vec{s}^j | C_i) \tag{5.24}$$

where C_i are the control-point values and $\vec{s}^j$ is the value of the surface coordinate

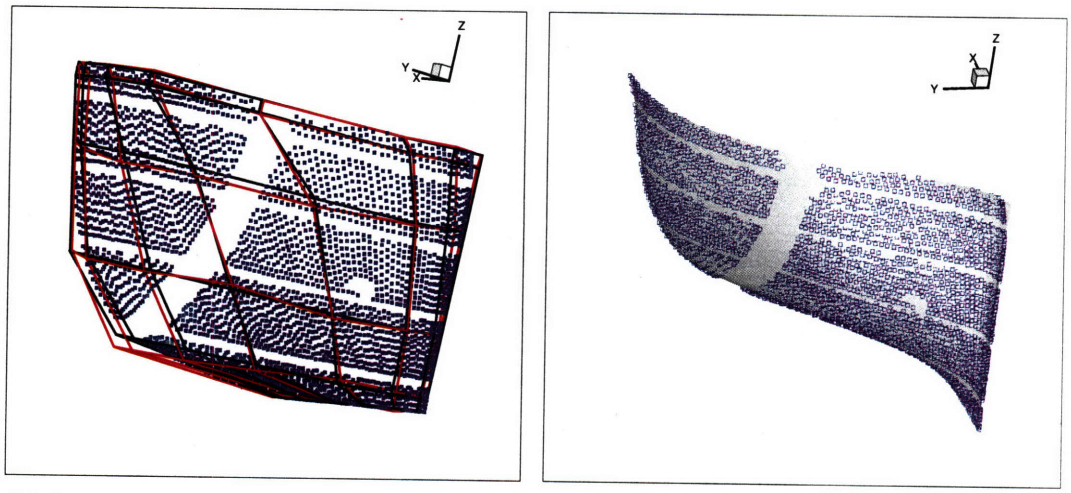
corresponding to data-point j. The goal is to optimize C_i to minimize the error in equation 5.23 but first we must set the value of $\vec{s}^j$ for every data point. This problem is not dependent on our particular choice of equation 5.23 as the error metric; it is an outcome of comparing two sets with independent parameterizations. The data is only parameterized by the label j, while the surface is parameterized by $\vec{s}$ and C.

In the machine learning literature, this is known as an unsupervised learning problem and we use an Expectation-Maximization (EM) style algorithm to solve it[4]. In the E-step the values of $\vec{s}^j$ are determined by solving equation 5.13 with $x = \hat{X}^j$, ie $\vec{s}^j = \vec{S}(\hat{X}^j)$. In the M-step the control-point values C are optimized as

$$C_i = \arg\min_{c_i} \varepsilon(\hat{X}_i, c_i)^2. \tag{5.25}$$

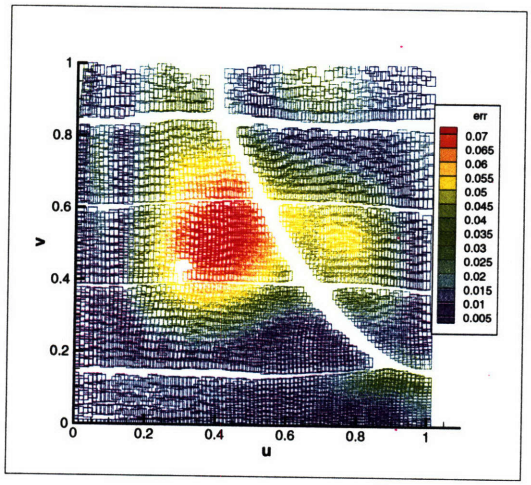
Because the change in C changes the values of $\vec{s}^j$ the process must iterate to reach the optimum. The success of this class of optimization problems is heavily dependent on the initial conditions and in this case the initial NURBS surface is fit to the data by hand.

Figure 5-7 shows the results of this process for the wave-maker run corresponding to 3D flow speed of 27.5 knots. Figure (a) shows the control-point mesh for the hand-fit geometry in black. In the same figure the blue squares are the data points from the video-capture and the red mesh is the optimized control-point set. Figure (b) shows a 3D rendering of the optimized hull along with the data points. The optimized surface fits the data well and the is quantified by the point-by-point error plots of figures (c)and (d). In these plots the error for every data point is plotted in the normalized surface coordinate space $u = \sigma/\sigma_{max}$, $v = \tau/\tau_{max}$. The RMS error dropped from $1e-2$ to $3e-5$ after optimizing the surface as detailed above. These figures also help illustrate the inhomogeneous distribution of data in the set, and that the NURBS surface is smooth and reasonable even in those regions. This low level of error should enable accurate representation of the experimental geometry.

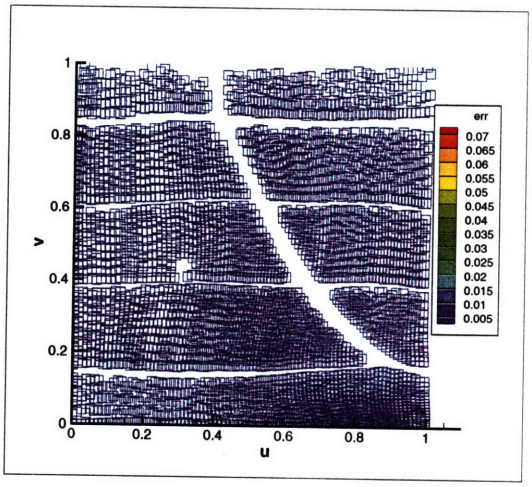


(a) Hand-fit and Optimized Control Points

(b) Optimized Surface



(c) ε^j: Hand-fit



(d) ε^j: Optimized

Figure 5-7: Optimization of the NURBS surface for the '27.5 knot' unsteady wavemaker data. Figure (a) shows the control-point meshes for the hand-fit (black) and optimized (red) hulls and the experimental video-capture data. Figure (b) shows the rendered optimized surface with the data points. Figure (c) and (d) show the point-by-point error for the hand-fit and optimized geometries, respectively.

5.3 Nonlinear Ship Bow Flow Results

The Cartesian-grid methods developed in the previous section and chapters are now applied to the nonlinear flow around a ship bow. First the data from the experimental wavemaker tests is used to further validate the numerical code with a set of 3D and 2D tests. Next an effective hull geometry is constructed based on that data. This hull is used in a series of 2D+T and 3D simulations, establishing the detailed characteristics of the 2D+T model for breaking ship bow waves.

5.3.1 Flexible Wavemaker Validation

The BDIM method has been 'validated' in 2D and 3D in section 4.5 but the flexible wavemaker data of [53] offers further opportunity to estimate the empirical risk of the cartesian-grid method for this class of flow.

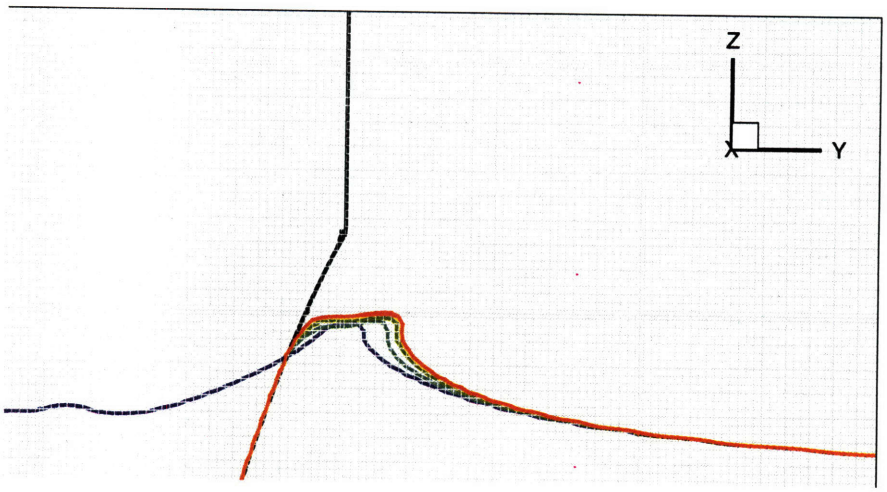
As shown in figure 5-3 the 'keel' of the experimental wavemaker is 0.914m above the tank floor and .914m below the undisturbed free surface. The full extension width of the wavemaker is 1.42m. Using an aspect ratio of $L/B = 8.29$ for the 5415 hull form the equivalent model dimensions are $D = 0.914m$, $B = 2.82$ and $L = 23.38m$. Based on these 3D scales the 2D Froude number for a given test may be converted to the length based Froude number through equation 5.8 and to the model-scale speed through equation 5.5. These are the 'speeds' commonly referenced in [53] and the speeds run experimentally are 12.5, 15.0, 16.5, 17.5, 20.0, 22.5, 25.0 and 27.5 knots. However, the true non-dimensional scale governing the wavemaker flow is the 2D Froude number, which are $0.120, 0.144, 0.158, 0.168, 0.191, 0.215, 0.239$ and 0.263.

The BDIM method is first applied to a direct simulation of the experimental 3D unsteady wavemaker flow. This is done to determine the effects of the cross plane dimension on the flow. The position of the wavemaker is prescribed to match the observed experiential motion for a 2D Froude number of $F_w = 0.191$. The experimental apparatus has a very small gap between the moving wave-plate and the sidewall of the tank. The 3-dimensionality of the simulation is enhanced by increasing the size of that gap to $h/D = 0.005$ and the results are shown in figures 5-8 and 5-9. Those

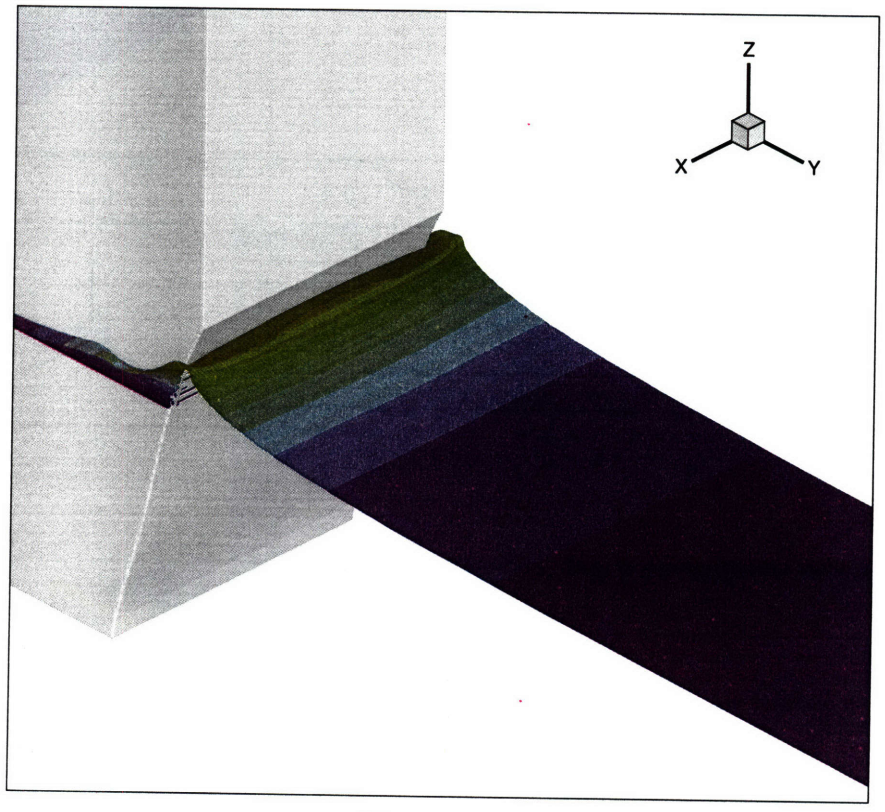
figures show snap shots of the free surface and wavemaker at times $t/T = 0.2, 0.3, 0.4$ colored by the wave elevation. In this simulation the free slip boundary conditions are used, the grid spacing is set to $\Delta x/D = 0.001$, $\epsilon/\Delta x = 5$ and $\Delta t/T = 0.0001$ to ensure the CFL conditions are met. The figures show that even for this large gap size the flow is essentially 2D at the centerline of the tank. The figures also show a peaked wave and spilling breaker at formed at this speed, which matched the observed experimental results.

The quantified experimental data available for comparison are the key wave features such as the contact line, wave crest and jet tip position as a function of time. These points are labeled in figure 5-10. The contact line is defined as the intersection point of the free surface with the wavemaker surface. The wave crest is defined as the highest point on the wave generated by wavemaker. The jet tip is only well defined for strongly plunging breakers, which are observed for the 22.5,25.0 and 27.5 knot cases.

The second set of simulations are run in 2D and use the measured wavemaker positions for the 20.0 and 27.5 knot cases. The simulations use the same grid size, time step and smoothing width as in the 3D case. The simulated free surfaces are shown in a waterfall-type plot in figure 5-11. As seen in these figures the wave generated in the 20 knot case is a strong spilling breaker and the 27.5 knot case is a strong plunging breaker with a well defined jet. Figure 5-12 shows a zoomed in view of that plunging breaker. While it is not specifically measured experimentally, the wave generated by the impact of the plunger appears to be more energetic than expected physically. Observations of similar large splash-up features have been observed by many 2D VOF studies of these types of breaking waves. This is likely due to a basic instability in the velocity field due the the extremely strong interface gradients and the discontinuous velocities they generate in the VOF method. In order to resolve this instability a simple Smagorinski subgrid-scale (SGS) model is added to the right hand side on the interface. Figure 5-12(b) shows the SGS model reduces the splash up but also artificially smoothes out the fine scale features of the flow. With the availability of more detailed measurements such models could be tuned to extract the

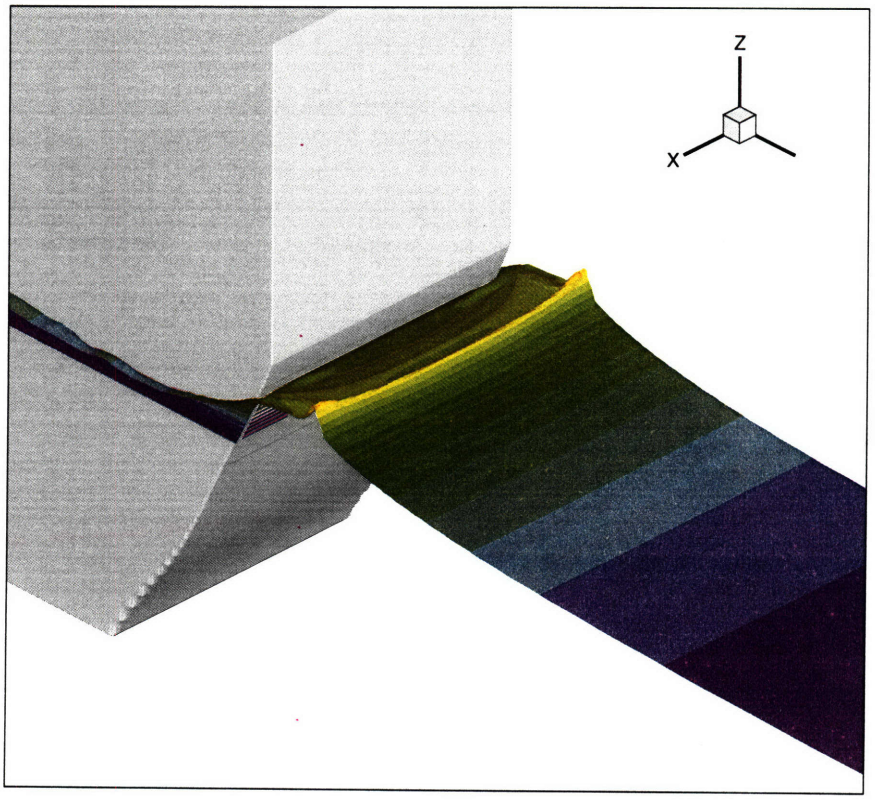


(a) cross plane slices

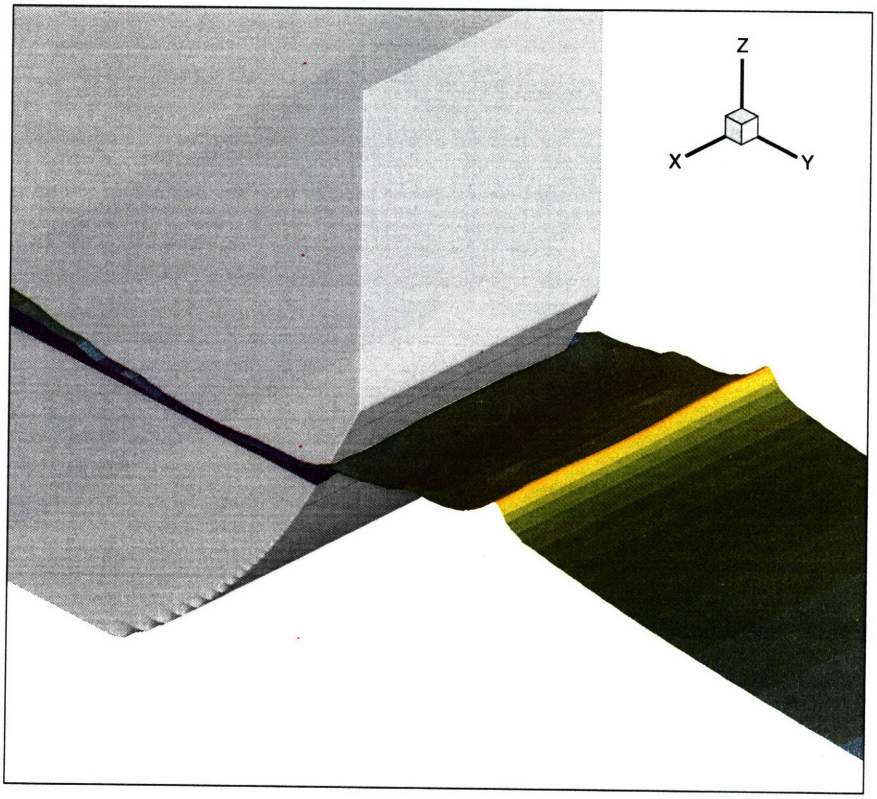


(b) isometric

Figure 5-8: 3D unsteady wave-board simulation at the equivalent 20.0 knot case. The first figure shows slices of the free surface at different cross sectional planes. The second shows an isometric view of the complete free surface. Both pictures are at t/T =0.2. The flow at the center plane appears to be quite 2D and minimally effected by the side-gap.



(a) $t/T = 0.4$



(b) $t/T = 0.6$

Figure 5-9: 3D unsteady wave-board simulation at the equivalent 20.0 knot case at $t/T = 0.4$ and 0.6.

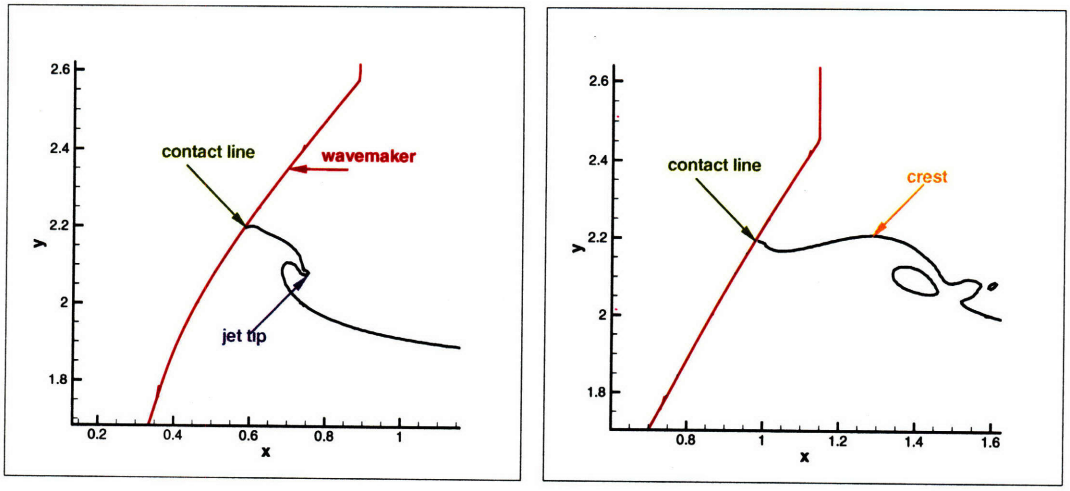


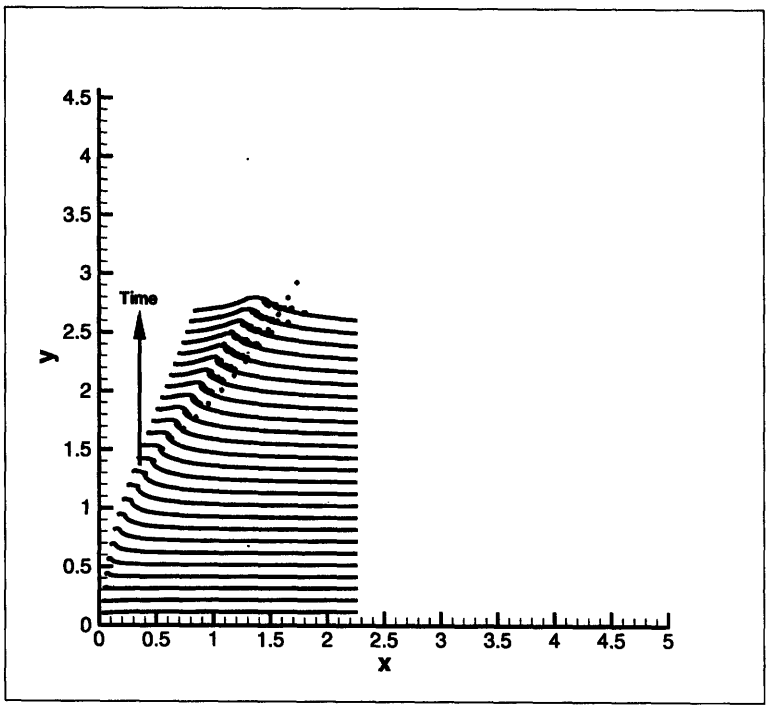
Figure 5-10: A sketch of the 2D+T breaking wave with the contact line, wave crest and jet tip labeled.

appropriate energy from the simulation.

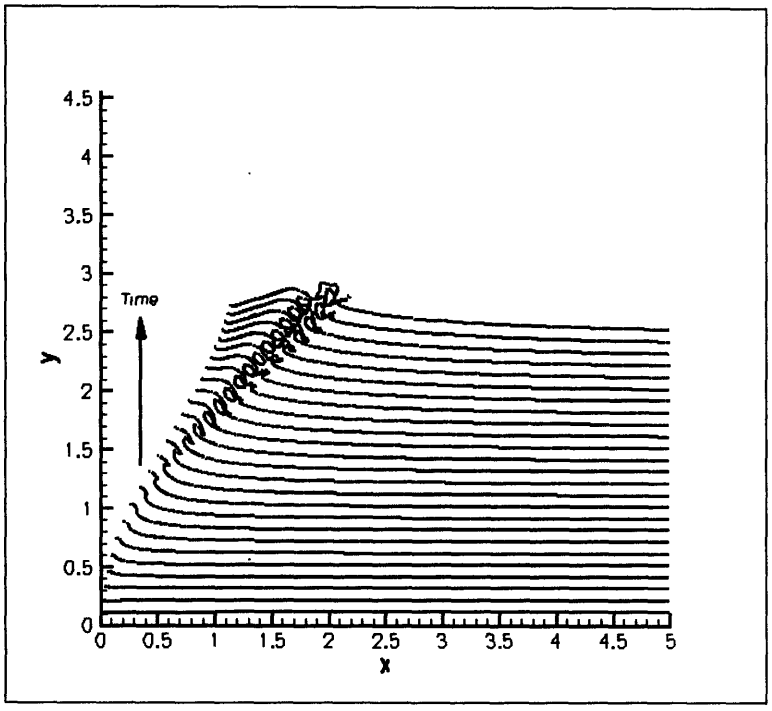
Quantitative measurements for the contact line are shown in figure 5-13 for the experiments, no-slip meta-equations and the no-penetration and momentum-slip meta-equations. Note that this physical problem is highly complex; the no-slip boundary condition and wetted and drying of the wavemaker wall means that the dynamic contact line is governed by many length scales. The qualitative comparisons are good for all the cases but it is clear that no model exactly captures the quantitative measurements. The no-slip method is adequate, but does not allow the contact line to fall as quickly as the experiments. However, at lower resolutions (no pictured) the no-slip results are very poor. The momentum-slip results are shown for 3 resolution levels with Δx =0.007 and 0.01 in additional to the Δx =0.005 case which is shown for both BCs. The momentum-slip allows for free motion of the contact line but over-shoots the measured maximum value more and more with increased resolution. The issue of boundary condition is further explored below.

5.3.2 2D+T Ship Geometry and Boundary Conditions

The third set of simulations are a set of true 2D+T runs. As such they must maintain the same effective geometry over the full range of speeds tested. The experimental

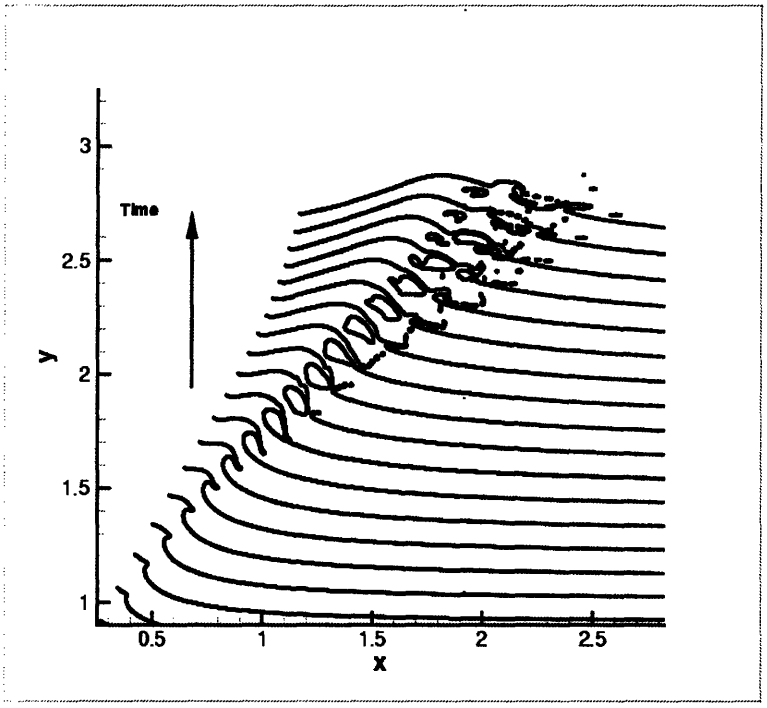


(a) 20.0 knot case

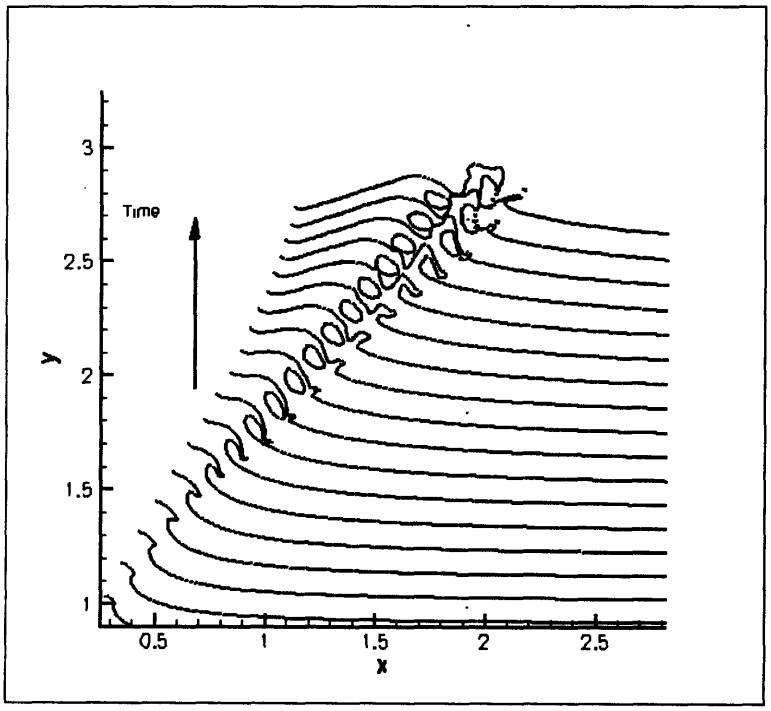


(b) 27.5 knot case

Figure 5-11: Waterfall plots for F_{2D} =0.191,0.263 corresponding to the 20.0 and 27.5 knots cases. The free surface profile is plotted every 40 times steps and each is shifted up by 0.1m from the time before to allow for ease of viewing.



(a) SGS Interface Model



(b) No SGS Model

Figure 5-12: Zoomed in view of the plunging breaker in the 27.5 knots case. The first figure shows the results when an SGS Interface model is added to the right hand side of the Euler equation and the second does not.

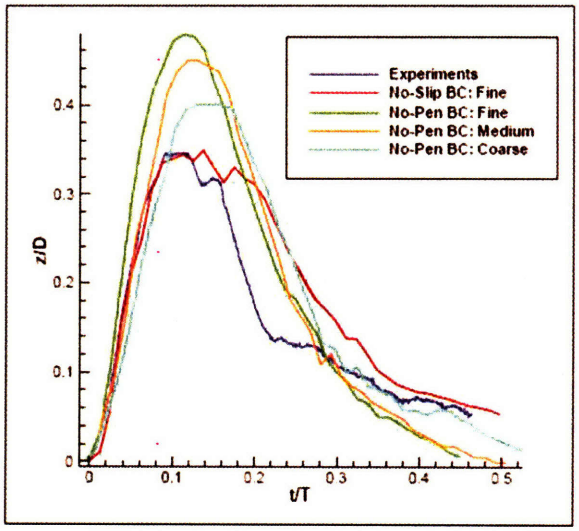


Figure 5-13: Comparison of experimentally measured contact line results with no-slip and momentum-slip (no-pen) simulated results for the 27.5 knot case.

apparatus is a physical system and there are slight variations in the shape of the wavemaker for different speeds. This is due to small controller errors as well as bending of the plate between the pinned points which is dependent on the pressure forces.

The wavemaker position data is available for two speeds 20.0 and 27.5 knots and the effective ship geometries for those hull are shown in figure 5-14. Clearly, the shapes do not match exactly but the NURBS optimizer described in section 5.2.6 can generate the best 'average' hull by including the position data for both speeds. Figure 5-15 shows the distance from every measured point to the nearest point on the average hull, and the maximum difference is less than 1% when scaled by the beam. Figure 5-16 shows the new average hull with the 20 and 27.5 knot hulls.

The boundary condition must also be fixed for these tests. The no-slip boundary condition is not applicable because of the axial flow assumptions of 2D+T and the numerical boundary layer which is generated for insufficient resolutions. The momentum-slip condition has been shown to be unstable; therefore a free-slip condition is used.

To test this boundary condition and the new hull form, figure 5-17 shows the results for the 27.5 knot hull, the average hull at the same speed and the experimental

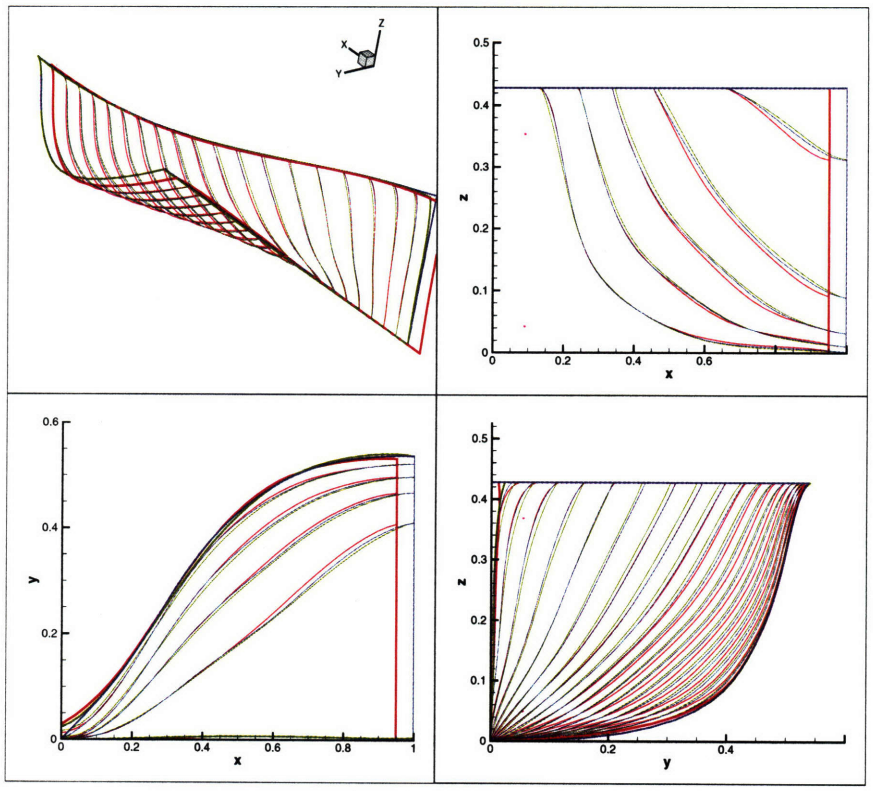


Figure 5-14: Hulls from the different speeds. Note the shift in the 27.5 hull and the mismatched width.

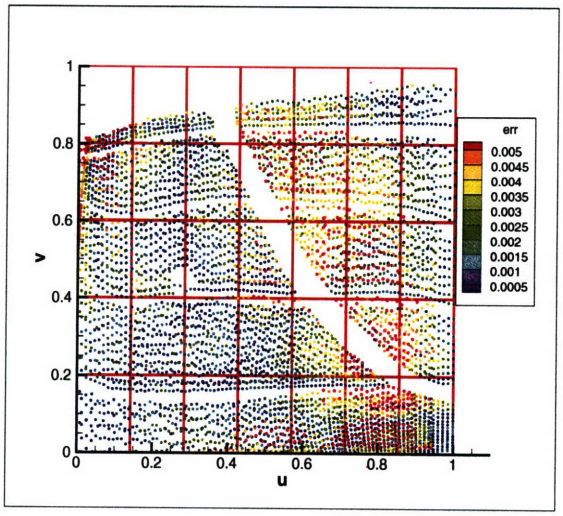


Figure 5-15: Distance from wave-board position data points to the optimized geometry. Data points are taken from three speeds and shifted to give maximize the overlap. The error is less than 1/2% and relatively random other than in the upper bow.

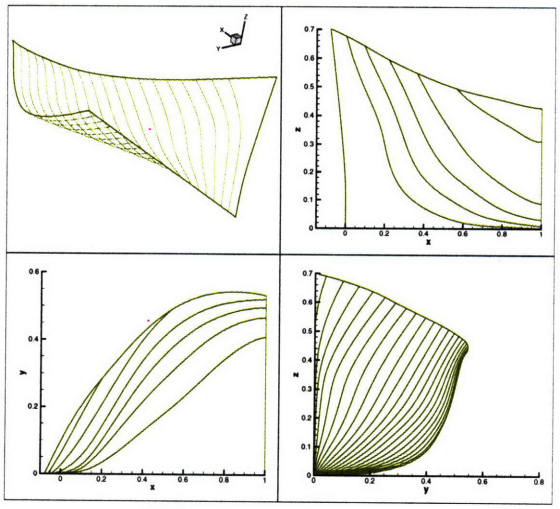
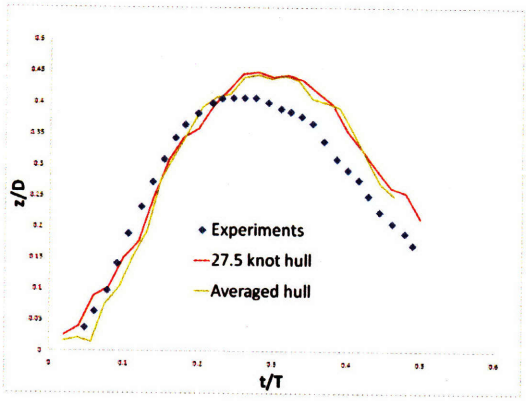


Figure 5-16: Engineering views of the optimized geometry. The views show the isometric, buttock-lines, water-lines and station-lines. The top of the vessel is taken after the 5415, the lower half is optimized to match the wave-board position data.

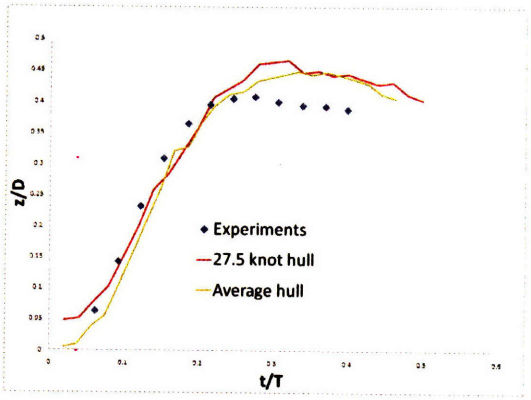
results for the contact line, wave crest and jet tip elevation as a function of time. Clearly, the free-slip boundary condition still over shoots the maximum contact line and wave crest elevation measured by the experimental results which feature a no-slip plate. However, the variance is small and the results are consistent for both hull forms for the contact line and wave crest. The jet tip shows very different behavior. The jet tip location compares well with experiments but is very sensitive to the hull geometry. The small 1% change in hull form has resulted in a 10% change in the tip elevation. This hints that the specifics of the jet tip of a plunging breaker are highly nonlinear. At this level of sensitivity, it may be difficult to replicate the complex experimental set-up with sufficient accuracy at the other 6 speeds. However, the contact line and wave crest results are likely to be representative.

5.3.3 2D+T Simulation Results

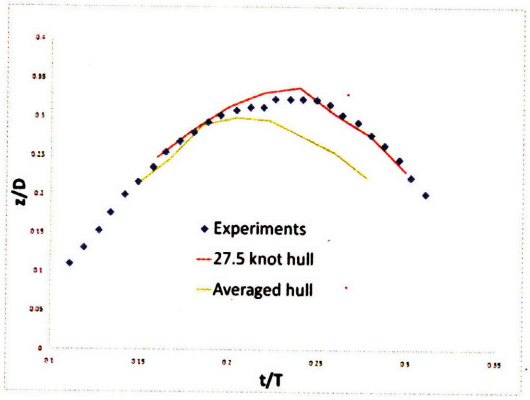
With the effect of geometry and boundary condition assessed, the 2D+T simulations are run over the full range of speeds. Figure 5-18 shows the results for the 12.5-17.5 knot cases, figure 5-19 shows the results for 20.0-27.5 knot cases and figure 5-20 shows a zoomed in view for the three highest speeds which have plunging breakers. The SGS model is used for these high speed cases to control the after breaking splash-up



(a) Contact Line

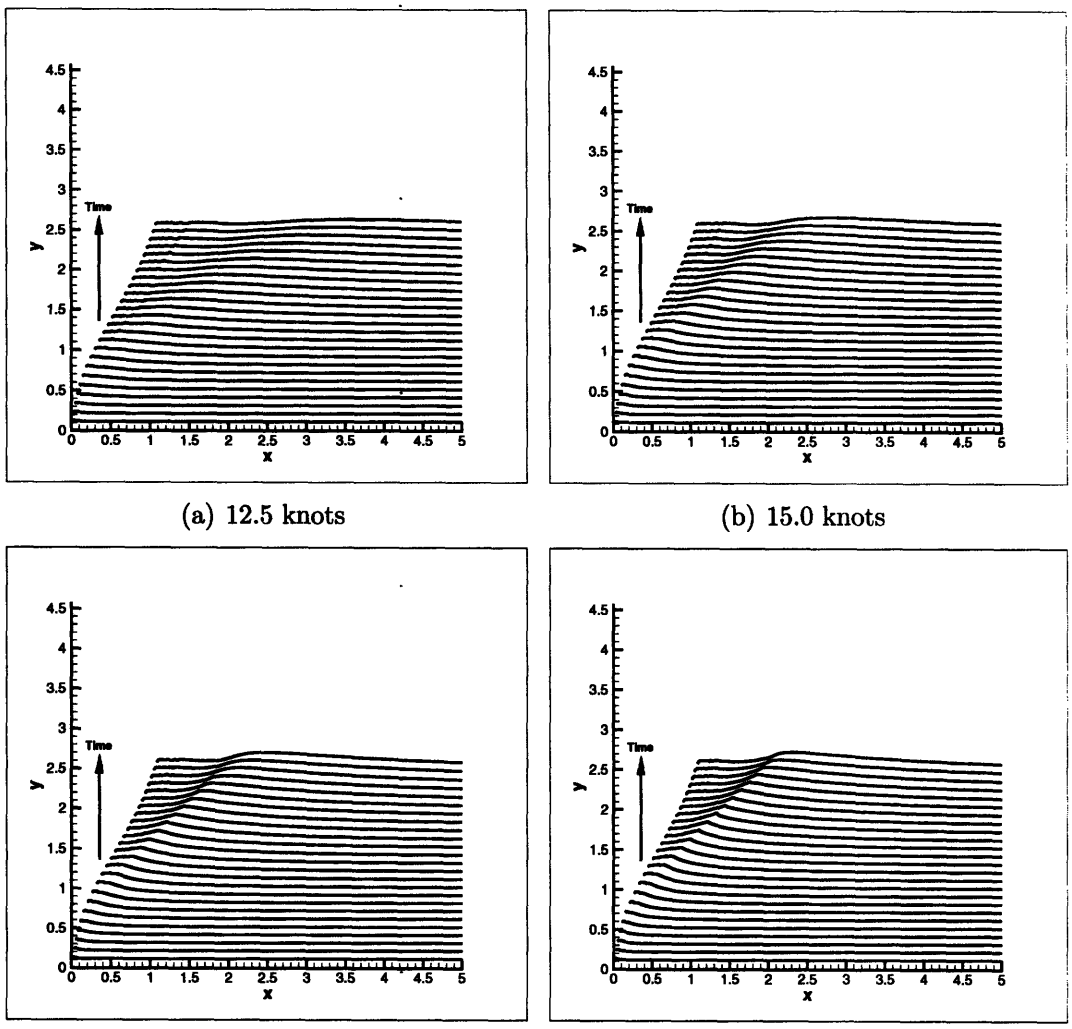


(b) Wave Crest



(c) Jet Tip

Figure 5-17: Quantitative results for the 27.5 knot case. Experiments are compared to the simulation results for the 27.5 knot hull and the average hull. Only the jet tip appears to be highly sensitive to the geometry.



(a) 12.5 knots

(b) 15.0 knots

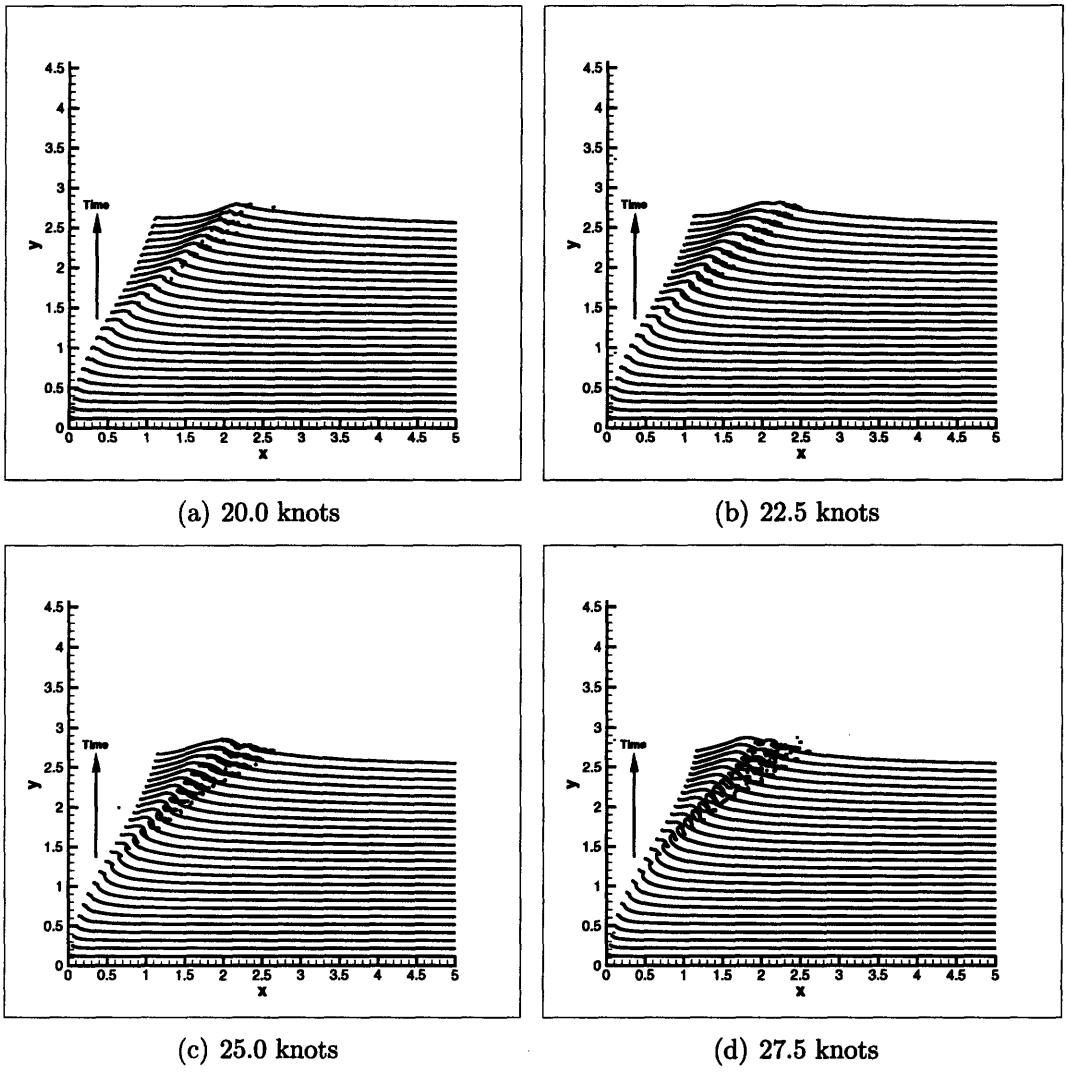
(c) 16.5 knots

(d) 17.5 knots

Figure 5-18: Waterfall plots for the 2D+T results for equivalent ship speeds of 12.5-17.5 knots. The wave progressive begins nearer to the hull and becomes more peaked. Small spilling breaking occurs for 16.5 and 17.5.

instability.

The breaking is shown to match the expected characteristics. As the speed increases the waves become large and steeper, with the onset of gentle spilling breakers at 16.5 knots. By 20 knots the spilling has become much stronger and above 20 knots the wave breaks with a plunging jet. This matches the experimental observations.



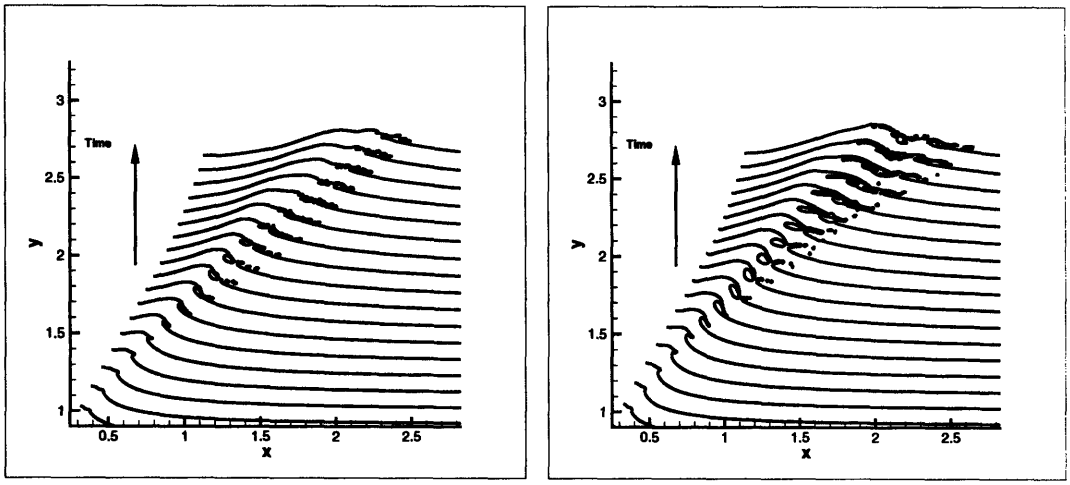
(a) 20.0 knots

(b) 22.5 knots

(c) 25.0 knots

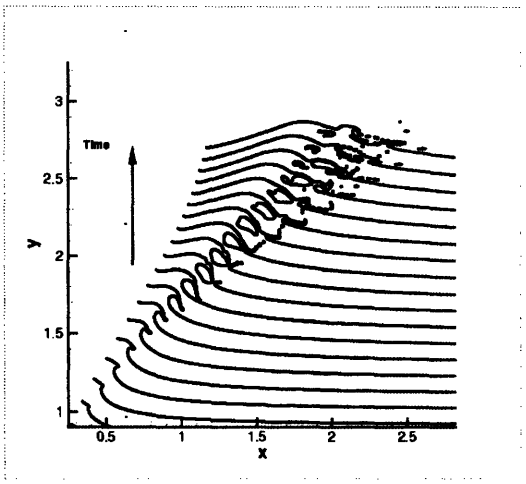
(d) 27.5 knots

Figure 5-19: Waterfall plots for the 2D+T results for equivalent ship speeds of 20-27.5 knots. The wave breaking continues to become more energetic with larger spilling breakers becoming plunging breakers with a well formed tip.



(a) 22.5 knots

(b) 25.0 knots



(c) 27.5 knots

Figure 5-20: Waterfall plots for the 2D+T results for equivalent ship speeds of 22.5-27.5 knots. This zoomed in view shows the plunging breakers more clearly. Because the flow is 2D the air entrained by the plunging breaker cannot escape until after breaking has become less extreme.

5.3.4 3D Simulation Results

The final set of tests are simulations of the 3D flow around the ship bow. It is discussed above that the 2D+T simulations are invariant to linear scalings of the speed and length while such scalings change the length-based Froude number and ship geometry. To study the effect, the slenderness ratios $\tan\alpha = B/L$ are varied while keeping the 2D Froude number constant for F_{2D} =0.144, 0.191 and 0.263. The actual slenderness value of $L/B = 8.3$ and double that $L/B = 16.6$ is run for all speeds. One more doubling $L/B = 33.2$ is used for the lower two speeds. The grid in the cross plane has half the resolution of the 2D+T runs ($\Delta z/D = 0.005$) and the axial spacing is set to $\Delta x/L = 0.005$. Therefore the axial spacing is doubled as the length is doubled. The ship speed is also doubled when the length is doubled to maintain the same 2D Froude number. As the bow flow is the only concern the exit plane is placed at midships. The new BDIM exit can handle this complex task and reduces the computational cost by more than a factor of 4.

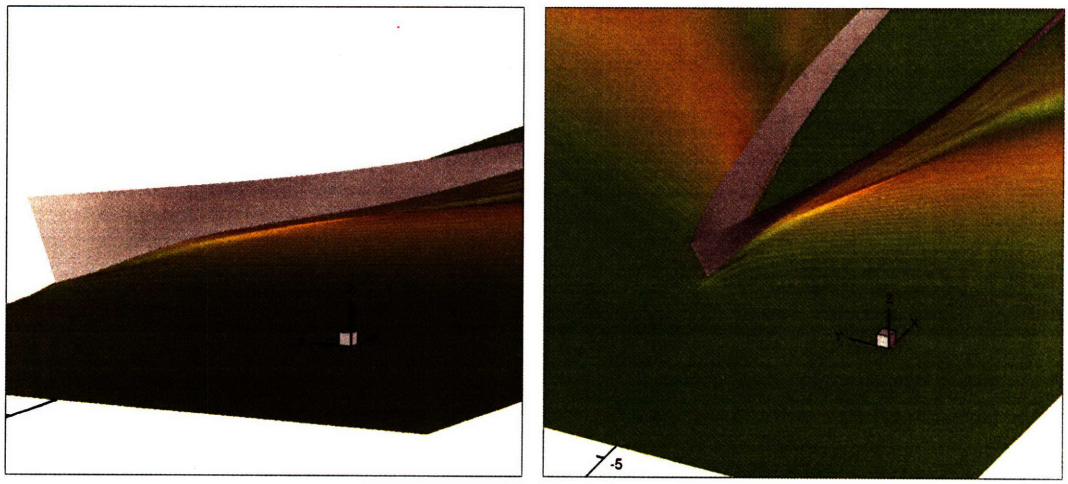
Figure 5-21 shows the 3D free surface for the lowest speed case. The figure has two views to allow for a more complete picture of the flow dynamics. It is clear from the side view that while the bow wave is steep, it is not breaking. Figures 5-22 and 5-23 show the same views for the intermediate and highest speeds which have progressively larger and more energetic breaking waves.

The waterfall plots for the three slenderness ratios are shown in figures 5-24 (the lowest speed), 5-25 (the intermediate speed) and 5-26 (the highest speed). The results show a clear trend towards steeper waves and earlier breaking with increased L/B.

5.3.5 2D+T Modeling Assessment

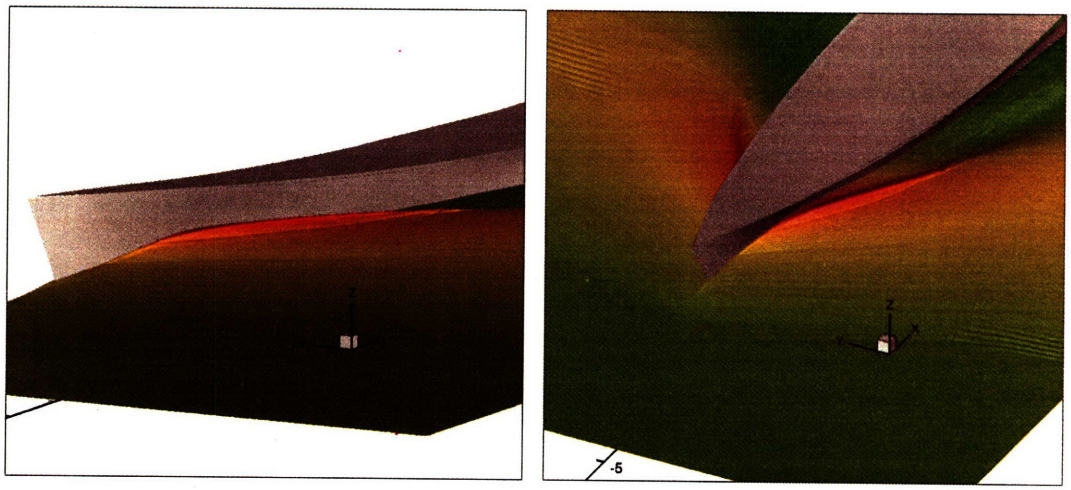
Next the quality of 2D+T modeling of 3D breaking bow waves is assessed. As stated above the geometry and 2D Froude numbers are identical in the 2D and 3D cases. Additionally, the same code and boundary conditions are used. This enables direct quantitative comparison of 2D+T and 3D simulations.

The plots for the contact line and wave crest are shown in figures 5-27 5-28 and



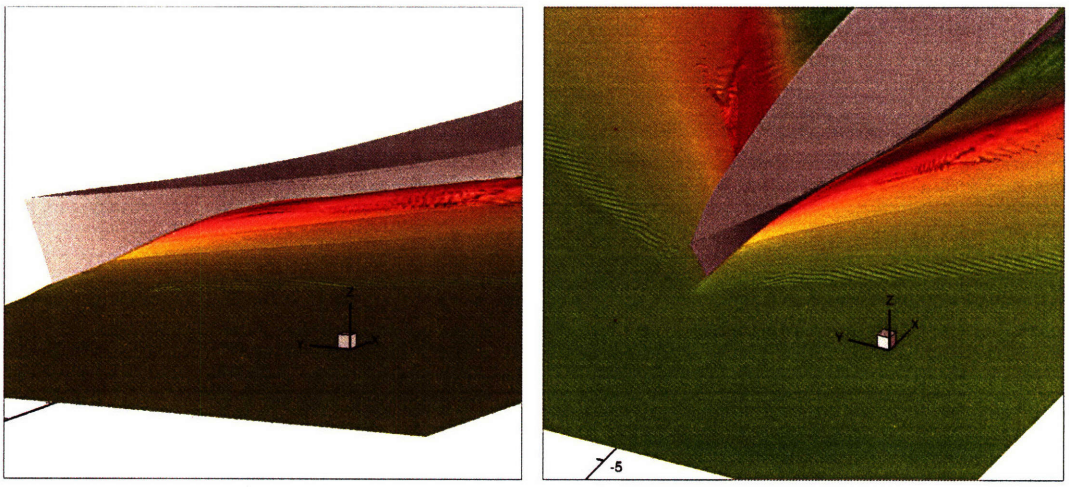
(a) Side view (b) Top view

Figure 5-21: 3D solutions for the nominal ship slenderness value of $L/B = 8.3$ for a side and top view at the lowest speed (15 knots model speed). The plots shows the ship hull in grey and the free surface colored by the elevation. The figure shows a steep but not breaking wave.



(a) Side view (b) Top view

Figure 5-22: 3D solutions for the intermediate speed (20 knots model speed). The figure shows a spilling breaking wave is formed.



(a) Side view (b) Top view

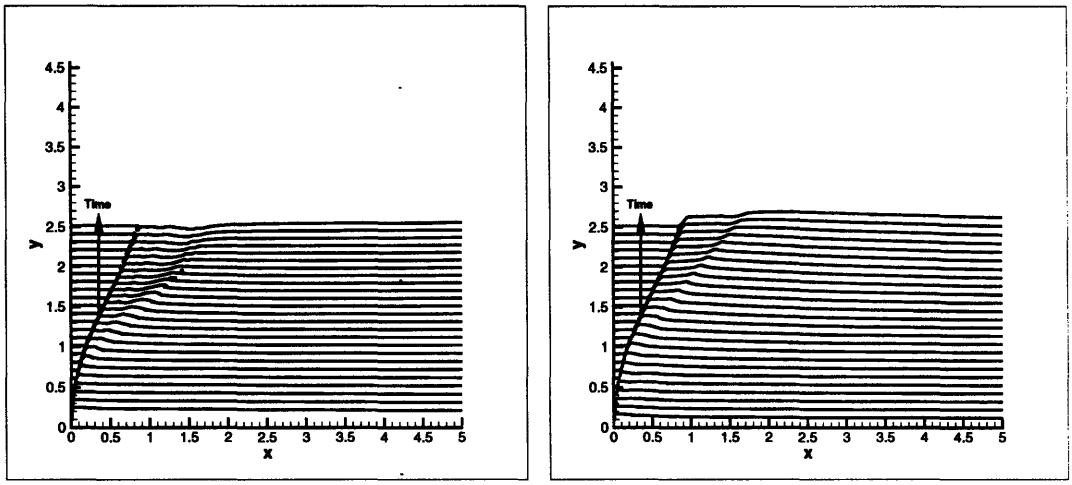
Figure 5-23: 3D solutions for the highest speed (27.5 knots model speed). The figure shows a large plunging breaker is formed.

5-29 for the three 2D Froude numbers. The 2D+T results for the lowest speed show poor agreement with the 3D results unless the vessel is extremely slender. At the level of $L/B = 33.2$ the 2D+T results do qualitatively predict the divergent wave, with quantified L_1 errors around 10%.

As predicted by 2D+T theory, the 2D+T model fares better at higher speeds. The intermediate speed predictions are qualitatively accurate for all but the least slender vessel and the L_1 error on the most slender vessel is around 4%. Note that while this is a spilling breaking wave the contact line and crest do not measure breaking wave characteristics.

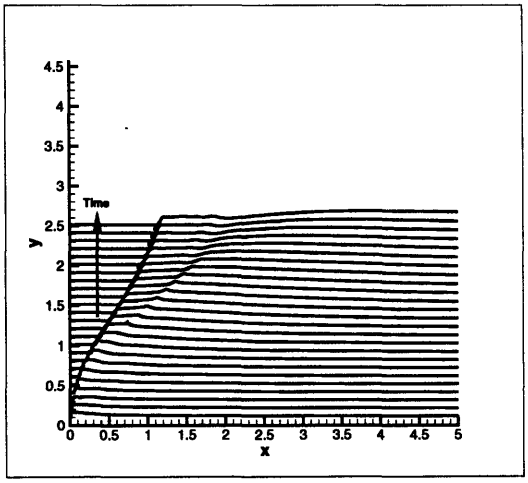
The highest speed shows 2D+T with good qualitative agreement in the contact line and wave crest height for even the $L/B = 8.3$ hull. Again, this is in agreement with 2D+T theory which assumes a slender vessel and very high Froude numbers. However, the jet tip predictions are far off the mark. Breaking is predicted much too early by the 2D+T model compared to either slenderness value hull. In previous sections it is shown that the tip position is very sensitive to changes in the geometry. While there is exact geometric correspondence in these tests, the jet tip again seems to be the most sensitive factor.

On reason for the poor modeling of the breaking jet tip by 2D+T is the assumption



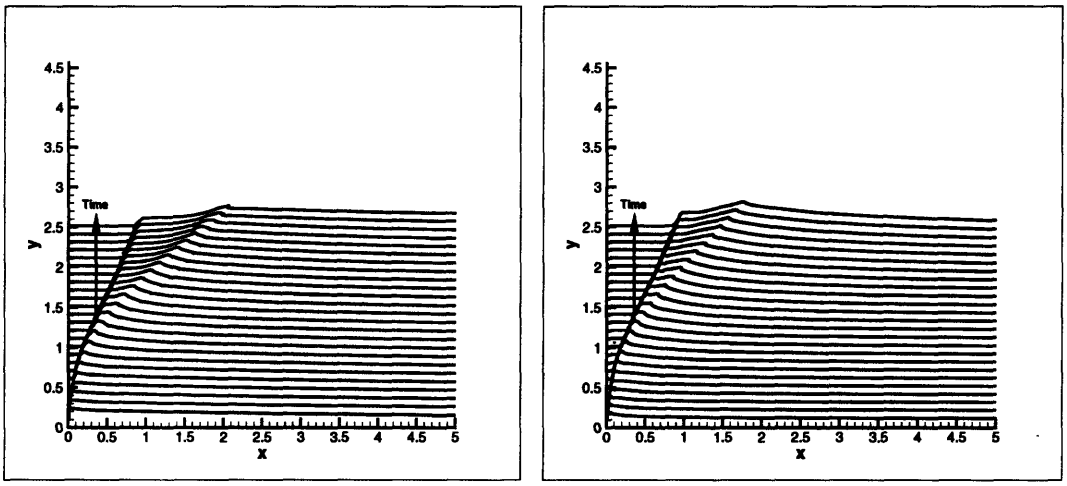
(a) L/B=8.3

(b) L/B=16.6



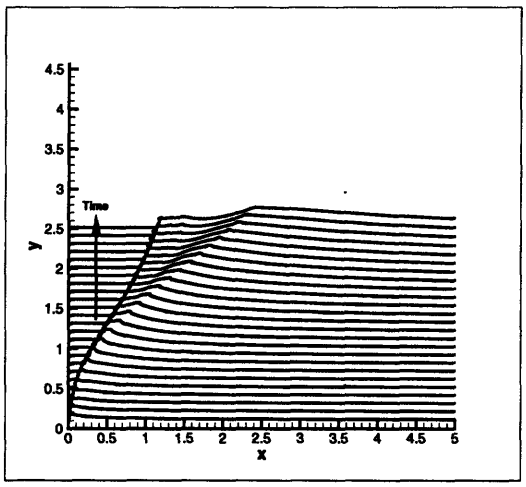
(c) L/B=33.2

Figure 5-24: Waterfall plots for the 3D results for $F_{2D} = 0.144$ and ship slenderness values from $L/B = 8.3 - 33.2$.



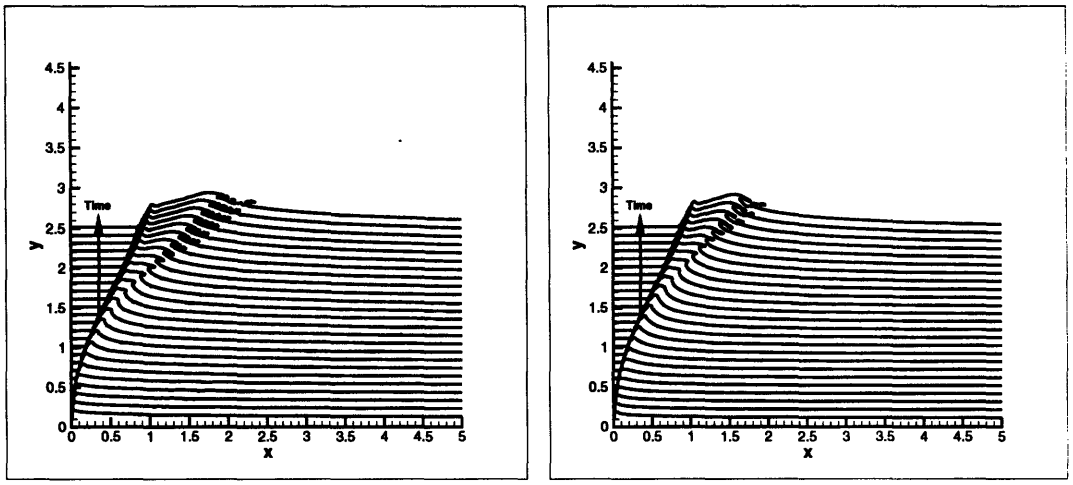
(a) L/B=8.3

(b) L/B=16.6



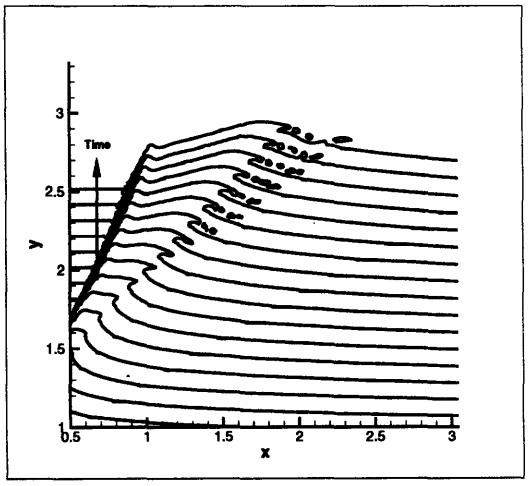
(c) L/B=33.2

Figure 5-25: Waterfall plots for the 3D results for $F_{2D} = 0.191$ and ship slenderness values from $L/B = 8.3 - 33.2$.



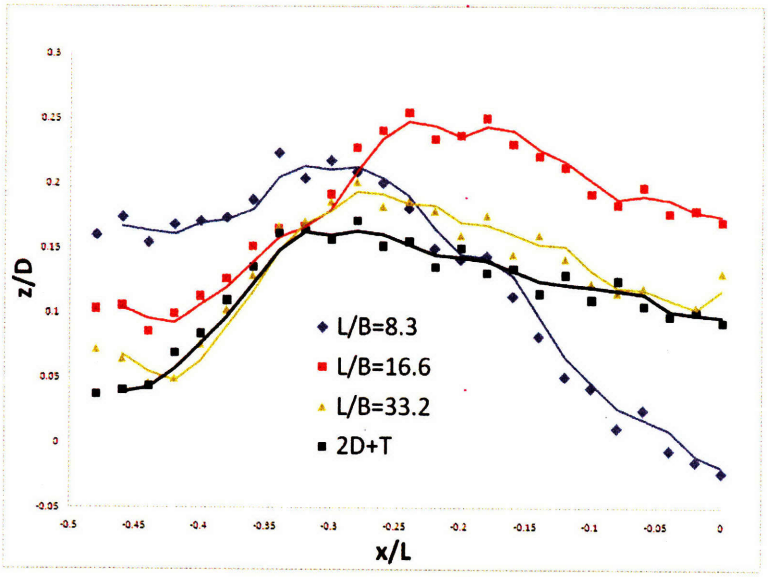
(a) L/B=8.3

(b) L/B=16.6

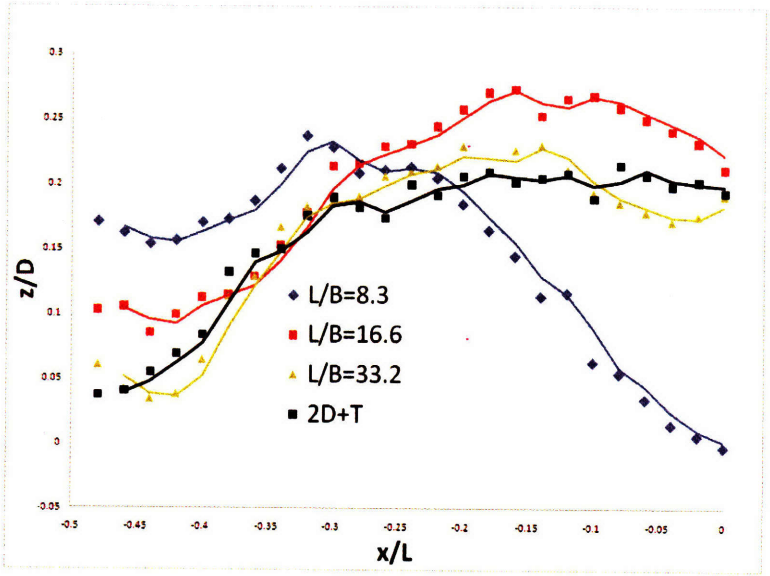


(c) L/B=8.3, zoomed

Figure 5-26: Waterfall plots for the 3D results for $F_{2D} = 0.263$ and ship slenderness values from $L/B = 8.3 - 16.6$. A zoomed view of the breaker is also shown.

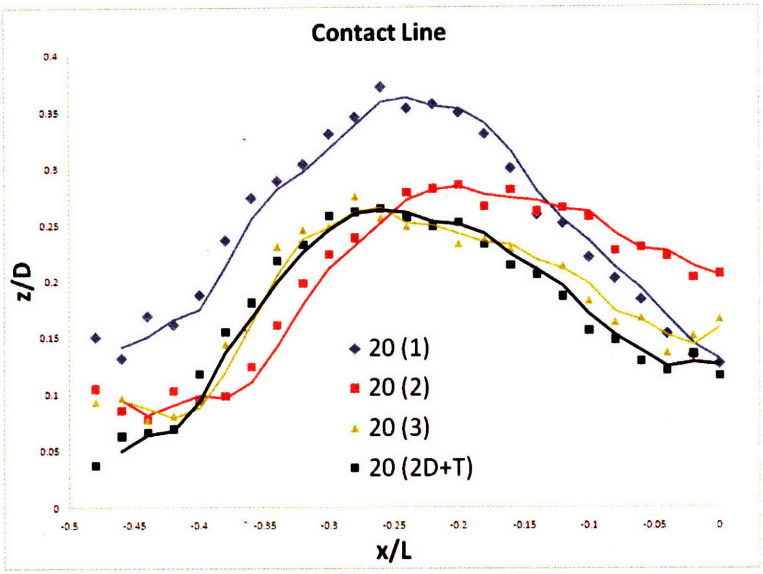


(a) Contact Line

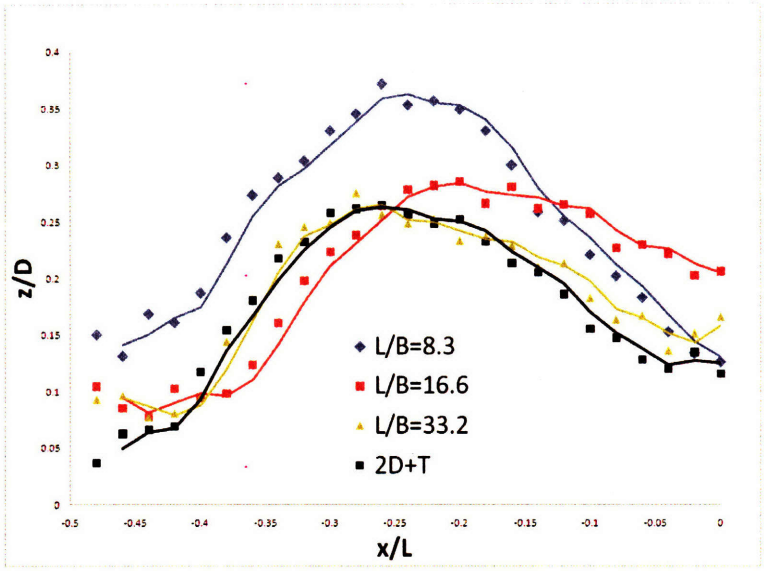


(b) Wave Crest

Figure 5-27: Effect of slenderness on the contact line and wave crest elevation for $F_{2D} = 0.144$. The 2D+T prediction in black is compared to the 3D prediction for three slenderness values. 2D+T is only valid for the most slender vessel.

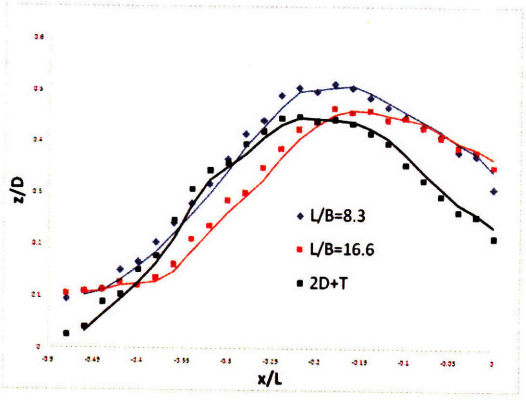


(a) Contact Line

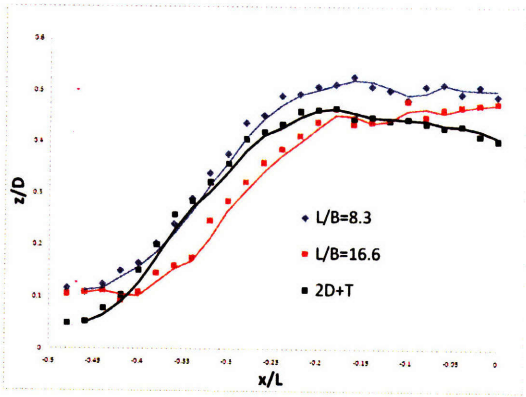


(b) Wave Crest

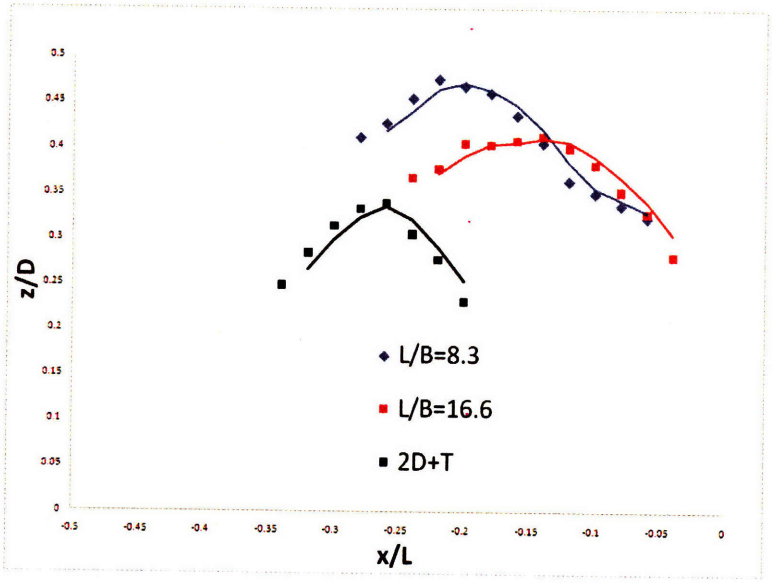
Figure 5-28: Effect of slenderness on the contact line and wave crest elevation for $F_{2D} = 0.191$. The 2D+T prediction in black is compared to the 3D prediction for three slenderness values. 2D+T is qualitatively reasonable for the all the vessels but quantitatively matches only the most slender.



(a) Contact Line

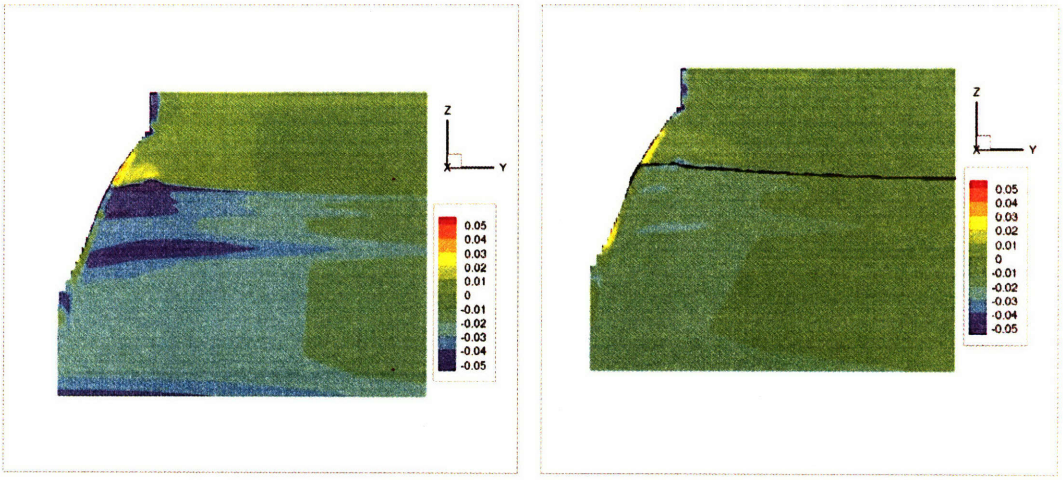


(b) Wave Crest



(c) Jet Tip

Figure 5-29: Effect of slenderness on the contact line wave crest and jet tip elevation for $F_{2D} = 0.263$. The 2D+T prediction in black is compared to the 3D prediction for two slenderness values. The qualitative and quantitative comparisons are good at this speed for the crest and contact line, but 2D+T does not predict the jet tip accurately.



(a) L/B=16.6 (b) L/B=33.2

Figure 5-30: Effect of slenderness on the axial velocity deviation from the free stream velocity for the lowest speed. The contours are taken at the same scaled axial position $x/L = -0.25$ and plotted on the same scale. The variation appears to be second order convergent to zero with increased slenderness.

of uniform axial velocity. In the bulk of the fluid, the variation of axial velocity from the uniform flow converges to zero with increasing vessel slenderness. This is demonstrated in figure 5-30 and is clear from conservation of mass arguments. The vessel pushes out a volume of fluid as it cuts through the ocean and this produces a transverse (y, z) velocity with an axial (x) gradient. There must be an axial velocity to balance this gradient but it is second order in the slenderness, as shown by equation 5.2. In the figure the scaled variation of axial velocity, $(u - U)/U$ where U is the free stream, shows this second order convergence to zero.

However, this argument does not apply to the breaking wave. The curvatures in a breaking wave, especially a plunging breaker, are much more severe than the gradients along the ship hull. Even in the limit of a very long ship the closure of a plunging breaker is a local small scale feature, and not likely to be well modeled by 2D+T.

In conclusion, this chapter details the application of the Boundary Data Immersion method to the nonlinear slender body model of breaking ship waves. Previous limitation in the stability and generality of 2D+T simulation are completely bypassed through the use of this new Cartesian grid method. The simulation are

validated through comparison to experimental measurements and the Euler equation was shown to model the flow well with appropriate slip conditions. The predictive usefulness of 2D+T was tested through direct comparison to 3D simulations and it was found that 2D+T predictions are only valid for very slender ($L/B > 16$) high speed vessels without overturning breakers.

Chapter 6

Modeling Theory for Physical Systems

Good modeling is critical for the successful design and assessment of engineering systems. The essential trade-off when modeling a complex system is between the cost of the model and the accuracy of its predictions. In modern CFD, the computational cost of highly-resolved physics-based predictions are enormous [64]. Cost can be reduced by using empirical models or simplified physics-based models but the accuracy of these models is limited and obtaining example data has its own costs.

This chapter contributes to the study and development of physics-based and learning-based models for physical systems. An introduction to a theory of modeling is presented to further the goal of generalized classification and characterization of physical system models. This new framework allows information theory results [54] to be applied to physical system models and help to establish universal modeling metrics. Those results are used to further motivate Cartesian-grid methods and set up the mixed learning and physics-based models of chapter 7.

6.1 Introduction

We begin with a typical example; predicting the roll motion of a ship at sea. The magnitude of this motion must meet strict requirements for the safe operation of the

vessel and there are many roll models of varying levels of accuracy and complexity. A typical model assumes the small scale roll motion of the vessel to be governed by a damped harmonic oscillator. The restoring force is determined by linear hydrostatic analysis of the hull geometry. The damping force and added mass are often determined by semi-empirical hydrodynamic relationships based on the vessel geometry and previous experience. The complete model is a tractable and useful prediction tool of a very complex physical system.

This kind of example is ubiquitous in engineering models and incorporates a mixture of many levels of idealizations. First, the system is idealized as having particular input and output variables. In this case the inputs are characteristic geometric variables and operation conditions such as sea state and forward speed. The output is the roll motion of the vessel. Next, those variables are given a specific relationship: a deterministic mapping from input to output. Analytic and/or numerical assumptions are used to simplify these mappings to the point of tractable evaluation. Above, the conservation of momentum is simplified to a harmonic equation for the roll. The highly complex hydrodynamics of bluff body flows are simplified into equations for the required harmonic coefficients; added mass, damping and restoring forces.

Often, modeling parameters are introduced at one or more of these stages which must be determined by observation. The hydrodynamic coefficients are themselves modeling parameters, for which additional equations are required. The semi-empirical mapping for the damping features many additional free parameters which are fit by a regression technique to available data. In more complex models numerical assumptions are made which result in numerical modeling parameters. For example, the numerical grid in CFD simulations is a set of millions of non-physical variables.

6.1.1 Fundamental Terminology for Model Theory

The first step in the analysis of a general system is to define the set of physical input variables X and the set of physical output variables Y. The outputs are the quantities of interest; hull resistance, slamming loads, failure rate, etc. The inputs are carefully chosen to capture the relevant behavior of the output without excessive

model complexity.

Note that, in general, there is no 'true' model for the physical system. By defining the input and output sets and imposing a causal relationship we have already begun the idealization process. The best that be done is to perform experiments for l instances of the input set $X^i = \{X^1, X^2, \ldots X^l\}$ and measure the corresponding output values $Y^i = \{Y^1, Y^2, \ldots Y^l\}$.

The list of data points $\{Y|X\}^i$ represents the 'true' system behavior (within the error margin of the measurements) but a mapping must be defined to make predictions for general values of the input set X. As such, we idealize the physical system as some deterministic mapping f between the set of input and output variables, ie

$$f : X \rightarrow Y$$

or

$$Y = f(X) \tag{6.1}$$

While we restrict ourselves to deterministic mappings, statistical properties can be modeled by including random variable metrics such as the mean and variance in the input and output sets.

As mentioned above, mappings for engineering systems generally incorporate a set of modeling parameters in addition to the input set, ie

$$Y = f(X|\theta) \tag{6.2}$$

where θ is the set of additional modeling parameters. The members of θ range from purely numerical (integration order-of-accuracy) to semi-physical (friction coefficients) and they must be determined through inductive or deductive [1] methods before a model may be designed and used.

[1]This terminology is explored in section 6.1.3.

6.1.2 Physics-based Models and Internal Variables

An additional level of sophistication is often employed when modeling physical systems, that of internal variables. The input and output sets X and Y are most often composed of variables of engineering interest such as drag coefficients, aspect ratios and so on. However, the conservation laws governing a general physical system cannot be posed directly in terms of those variables. Instead these laws are based on fundamental values such as time, position, velocity, local stresses and strains, etc.

This set of internal variables, denoted χ, are the degrees of freedom of the physical system. In the variational framework, for example, χ is the complete set of generalized coordinates [30]. For a general system based on a continuum, such as the fluid-body mechanical system for breaking bow waves, the size of this set is infinite. To make these problems tractable, the set is parameterized, typically by a tessellation of the continuum. This results in the finite numerical grids and nonlinear systems of equations at the core of computational fluid dynamics (CFD) and other physics-based methods. We denote this system of the governing equation and its internal variables generally as

$$\mathcal{M}(\chi) = 0 \tag{6.3}$$

and χ_o as the solution of that problem. The formulation and solution of this internal model is not trivial, and occupies four chapters of this thesis. However, the important issue in this chapter is that the set χ is not equal to the input or output sets of the engineering problem. Thus, a general 'physics-based' model includes an additional set of mappings from X to the set χ and from the solution χ_o to Y,

$$\chi = g(X|\theta_i) \tag{6.4}$$

$$Y = h(\chi_o|\theta_o) \tag{6.5}$$

where the set θ has been split into input θ_i and output θ_o subsets. Stated another way, the mapping g sets up the conservation problem based on the input variables and modeling parameters. The mapping h 'measures' the solution of the conservation

equations to establish the values of Y. It is often the case that human engineers carry out these mappings themselves and it is important to include their influence on the overall properties of the model. This is because the combination of these two processes with the solution of equation 6.3 defines the complete mapping 6.2 for the physical system.

6.1.3 Analogy Between Model Theory and Logical Reasoning

Formalizing the construction and assessment of models is akin to the process of casting propositions in their analytic logical form. The logical form of a proposition is a way of representing it to display its similarity with all other propositions of the same type [35]. While claiming no such rigor, this work shares an analogous goal of broad categorization of model types and functionality.

The analogy to logic may be extended by observing that typical engineering models, such as those for ship roll prediction, combine a mixture of processes akin to the basic deductive and inductive logical methods. Recall that in deduction, a set of axioms are assumed, and further statements are created by applying formal operations. For example

> *Male birds with brightly colored feathers attract more mates.*
>
> This bird has no feathers.
>
> He is less likely to attract mates.

The first two statements are the axioms and the last is the conclusion. The truth of the deduced result follows directly from the truth of the axioms. An example of induction along the same lines is,

> Male jays, robins, and especially peacocks have flamboyant plumage.
>
> *Brightly colored feathers help male birds attract more mates.*

Unlike in deduction, inductive statements have a likelihood of generalized truth dependent on the size and relevance of the sample set.

The essence of induction is generalization from instances while deduction is application of generalities to an individual instance. By analogy, the evaluation of the

mapping f given the inputs X and modeling parameters θ is a deductive modeling process. In contrast, determining the model parameter values θ which best fit the mapping to example data $\{Y|X\}^i$ is an inductive modeling process. A weaker form of the analogy also applies to the model equations themselves. A physics-based model is deductive in nature; incorporating a specific instance of a general conservation law. A generic functional mapping is inductive in nature; relying purely on examples to build a model of the system.

Deduction and induction both play a role in the scientific method. Induction is required to generate the axioms which are tested and made fruitful by deduction as in the italicized sections of the example above [2]. Similarly, both inductive and deductive processes are used in engineering models; ship roll models, Froude's resistance decomposition, etc. The 'best' of these models tend to incorporate physics-based mappings where possible and use data to inductively establish mappings for the other terms. In this work such a model is called a physics-based learning model but to define what is meant by the 'best' model requires the development of universal metrics on which models may be compared and judged.

6.2 Universal Modeling Metrics

Up to the last few decades there has been minimal use of computational fluid dynamic (CFD) in design. Historically the solution techniques used to solve these problems were of limited accuracy and applicable only to a very small subset of the complete system thus lacking generality. With the continuing increase of availability of computational resources, modern computationally intensive methods became feasible. However, producing quality solutions with modern CFD techniques is limited by the method's complexity. The codes are sensitive to literally millions of input parameters meaning that highly experienced users are required to achieve meaningful results. Such complexity also limits the speed and reliability of model alteration and testing,

[2]The most notorious counter example in science is Einstein's theory of general relativity and he had little appreciation for induction later in his life [29].

limiting their use in practical science and engineering applications.

While complexity and accuracy are indicators of a model's effectiveness a concrete sets of relevant metrics must be derived in order to determine the optimality of a given prediction model. Such metrics must establish the two fundamental quantities; the model's cost and its general predictive performance.

6.2.1 Cost Metrics

The economic realities of meeting design schedules or research objectives demand that cost is an important factor for any model of an engineering system. There are many units for the model cost C but the most universal is time; man-hours, compute-cycles, etc. The time a model requires to make predictions can be broken down into a few basic components: the development cost (including data acquisition), the interface cost, and the evaluation cost.

The model evaluation cost is the time required to map a given input X with parameter values θ to the output Y. The evaluation time can range from milliseconds for a spline evaluation to on the order of hundreds of years for a fully resolved direct numerical simulation (DNS) of a full-scale ship flow. The additional cost of the DNS evaluation is due to a complex internal model, discusses in section 6.3. The evaluation cost of a physics-based simulation can be parameterized by the size N of the internal variable set χ. The exact performance depends on the details of the computational scheme, but increasing N at least proportionally increases C. DNS simulations use $N \sim O(10^{10})$, accounting for their enormous evaluation cost.

The evaluation cost is the most common cost associated with computational models because it is simple to quantify, but it is only a part of the picture. The development cost is the time required to build and validate a model and it is more difficult to measure than the evaluation cost. Generally, the development cost is an overhead and can range from hours to many years. One element of the development time is the cost of generating a data set for the system. This cost is proportional to the number of examples l in the data set and the difficulty in obtaining each value. A deductive model uses data for validation, whereas an inductive model uses data for optimization

of the model parameters θ. As discussed in the next section this implies that l will need to be larger when using an inductive model, increasing the development cost.

Inductive models also have the cost of parameter optimization included in their development time. This process is also known as inversion, training and learning. This cost is dependent on the mapping type and evaluation cost as well as the optimization technique and the number of model parameters. For example, if the mapping function is linear in the parameters then it may be inverted using a least-squared method in $O(m^2)$ time for the m parameters. On the other hand if the mapping is complex and highly nonlinear in the parameters (such as a CFD code) then the optimization may take many days.

Finally, the interface cost is the time required for a user to become capable of setting up the mapping and digesting the results. In the case of the physics based models discussed in section 6.1.2 the input set must be converted to a set of optimal internal variables and the resulting physical values must be measured to obtain the output set, as in equation 6.4. For a complex CFD code this may require years of training and advanced engineering degrees on top of day or weeks of time for each model evaluation. Models which are intuitive, robust to non-optimal modeling parameters and automate the mappings of equation 6.4 greatly reduce this cost.

6.2.2 Predictive Performance Metrics

The fastest model is not optimal if it does not deliver predictions of sufficiently high quality, necessitating predictive performance metrics. While different problems may have different specialized concerns (conservation of certain metrics, etc) the most general metric is the statistical expectation of the prediction error. We write the expectation as

$$R(\theta) = \int L\left(f(X,\theta), Y\right)\ \mathrm{d}P(X,Y) \tag{6.6}$$

where L is an error norm (called a loss function in the statistical learning literature) and $P(X,Y)$ is the unknown probability distribution from which the instances are

drawn[3]. The quantity R is known as the expected 'risk' of the model in information theory and is a function of the choice of model parameters θ. By choosing the model (and model coefficients) which have the lowest expected 'risk' we ensure the best predictive capabilities over the full range of possible inputs.

As the distribution $P(X,Y)$ is unknown, the available alternative is the empirical risk, defined by

$$R_{emp}(\theta) = \frac{1}{l}\sum_{i=1}^{l} L\left(f(X^i,\theta), Y^i\right) \tag{6.7}$$

where, the l points in the $\{Y|X\}^i$ data set are used to approximate the distribution. Therefore, the empirical risk R_{emp} is a function not only of the parameters, but also of the data set used to compute it. A model which has very low empirical risk may perform poorly on the unseen data, similar to the limitations of inductive logic. Only in the limit of infinite data does the empirical risk converge to the actual risk of the model. *Therefore, the empirical risk must not be used alone to select a model or model parameters.* A more complete estimate of the true risk of a model is needed and it is supplied by information theory.

6.2.3 Vapnik-Chervonenkis Bound

There is a remarkable result of statistical learning theory due to Vapnik [63] which bounds the difference between the empirical risk of equation 6.7 and the unknown actual risk of equation 6.6 without knowledge of the probability distribution $P(X,Y)$. With the help of this bound, we can choose a model which minimizes the true risk instead of only the empirical risk, increasing the chance of accurate predictions on unseen data.

To introduce the bound, first consider the case of a binary classification model. This model is trained to respond $y = \{1,-1\}$ based on the value of the input X and model parameters θ. A pertinent classification question from this thesis is whether or not a certain hull generates a breaking bow wave. Let us define event κ as the

[3]Reference [3] is an excellent and highly recommended review of statistical learning applied to pattern recognition. Much of the following sections follow the general approach presented in the early parts of that paper.

occurrence of breaking bow wave when a ship travels at 20 knots. Define the input set X as some set of hull shape characteristics such as the entrance angle $\alpha = B/L$ and draft aspect ratio $\gamma = D/L$. A binary classification model for this event predicts

$$f(X|\theta) = \begin{cases} 1 & \text{if } \kappa \\ -1 & \text{otherwise} \end{cases} \tag{6.8}$$

where θ are model parameters optimized to reproduce the observed examples. Such a model would be very useful to engineers if the predictions were accurate.

The loss function for this model is $L = \frac{1}{2}|f(X,\theta) - Y|$ where Y is the observed binary result. When the model predicts correctly $L = 0$ and when it does not $L = 1$. Vapnik [63] showed that if the empirical risk is computed from independently drawn and identically distributed (i.i.d.) sample data $\{Y|X\}^i$ then a bound on the true risk of the form

$$R(\theta) \leq R_{emp}(\theta) + \sqrt{\left(\frac{h}{l}(\log(2l/h) + 1) + \frac{1}{l}log(\eta/4)\right)} \tag{6.9}$$

holds with probability $1 - \eta$, where l is the number of samples drawn and h a characteristic of the model called the Vapnik-Chervonenkis (VC) dimension. The second right hand side term is called the called the VC gap and it is not dependent on the system, the distribution, or values of the model parameters. It is only dependent on l and h.

For example, assume model 6.8 is trained using $l = 100$ available examples for ships with different values of the input set X. Using a model with a VC dimension of $h = 3$ and setting $\eta = 0.01$ (such that our confidence level is $1 - \eta = 99\%$) then equation 6.9 states that the true risk is bounded by $R(\theta) \leq R_{emp}(\theta) + 0.3$. Assume that when the parameters θ are optimized the empirical risk of the model is found to be $R_{emp} = 0.1$ (the model predicts incorrectly for 10% of the sample problems). In this case we can say with 99% confidence that the true risk is below 0.4 knowing nothing more about the problem or model. To better explain the implications of this bound requires a discussion of the VC dimension h.

6.2.4 Description of Model Learning Capacity

The VC dimension h is a measurement of a model's learning capacity; the model's ability to replicate examples through adjustment of the parameters θ. The VC dimension is a critical metric in comparing the optimality of different system models because it bounds the generalization error, as shown in equation 6.9.

In other elements of engineering the word 'capacity' has straightforward and positive connotations, but in information theory the capacity embodies a trade-off between empirical and generalization errors. Equation 6.9 can be extended to any class of models (not just binary classifiers) by writing it as

$$R(\theta) \leq R_{emp}(\theta) + W_\eta(l, h), \tag{6.10}$$

where W_η is the general VC gap, quantifying the generalization error. Equation 6.10 expresses the balance that must be achieved between model flexility and model generality, and the capacity h is the key parameter in that balance.

On the one hand, increasing the capacity of the model will generally allow it to fit the example data better, reducing the empirical error R_{emp} after optimizing the parameters θ. Returning to the breaking wave classifier of equation 6.8, by increasing the capacity to $h = 10$ it is possible that the trained model could correctly classify all $l = 100$ examples, reducing the empirical error to zero.

On the other hand, the VC confidence is monotonically dependent on the VC dimension; for a given amount of data increasing h leads to a larger bound on the true risk. Figure 6-1 demonstrates this point by plotting equation 6.9 versus h/l. In the case of the hypothetical wave breaking classifier, increasing h to 10 increases the VC gap to $W_{0.01} = 0.5$. Even if the empirical risk has been driven to zero the total risk bound increases to $R \leq 0.5$; worse than the $h = 3$ case. *Thus, a model with excessive capacity is predicted to perform poorly on unseen data even if it has made no errors on the training set.*

The definition of the VC dimension for a binary classifier (such as the breaking wave predictor discussed above) is the maximum number h of input vectors

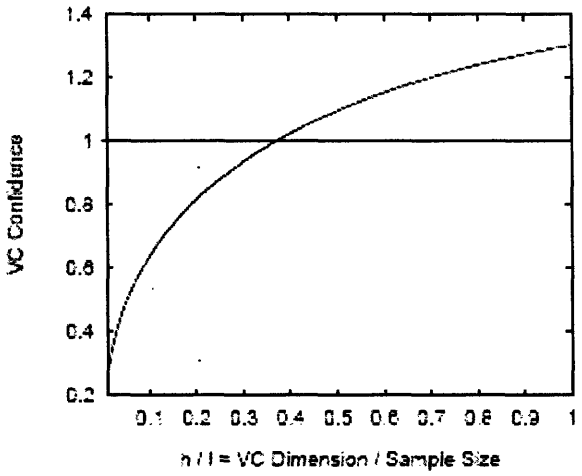


Figure 6-1: A plot of equation 6.9 for $\eta = 0.05$ and $l = 10,000$ as a function of h/l. Reproduced from [3].

$X^1, \ldots, X^h$ that can be separated into two classes in all 2^h possible ways using functions of the set. This ability to classify all possible labeling for a set of points is called 'shattering' in the machine learning literature. Thus, a model which can shatter h points has VC dimension of h.

For example, if the hull geometry set in equation 6.8 contains only the entrance angle and draft aspect ratio (ie $X = \{\alpha, \gamma\}$) then a 2D linear classifier could be used,

$$f(\alpha, \gamma|\theta) = H(\theta_1\alpha + \theta_2\gamma + \theta_3), \tag{6.11}$$

where H is the heavy-side function returning 1 for positive arguments and -1 for negative arguments. This classifier can label three points in all $2^3 = 8$ possible ways ($Y^i = \{0,0,0\}, Y^i = \{1,0,0\}, \ldots, Y^i = \{1,1,1\}$), as pictured in figure 6-2. Stated another way, the 2D classifier can exactly duplicate the labels ('breaking' or 'non-breaking') for three example data points, but not for four points. Therefore the VC dimension is $h = 3$ for this model [4].

Equation 6.8 has a binary output but it is often the case in engineering that a real valued output is required. For example, a model may predict the drag coefficient C_D

[4]Note that if the points are aligned linearly the classifier can only duplicate two points. The VC dimension is defined for the 'best' distributed set of points, not every set.

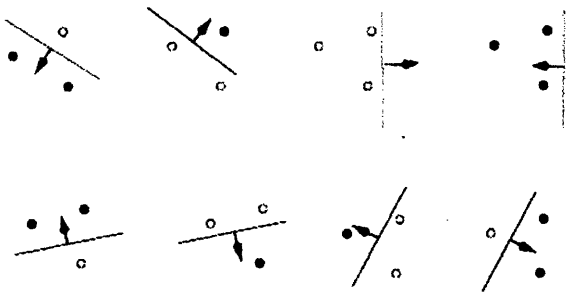


Figure 6-2: The eight possible binary labeling of 3 points, shattered by the linear classifier of equation 6.11.Reproduced from [3].

of a flat plate for a given Reynolds number Re, ie

$$f(Re|\theta) = C_D. \tag{6.12}$$

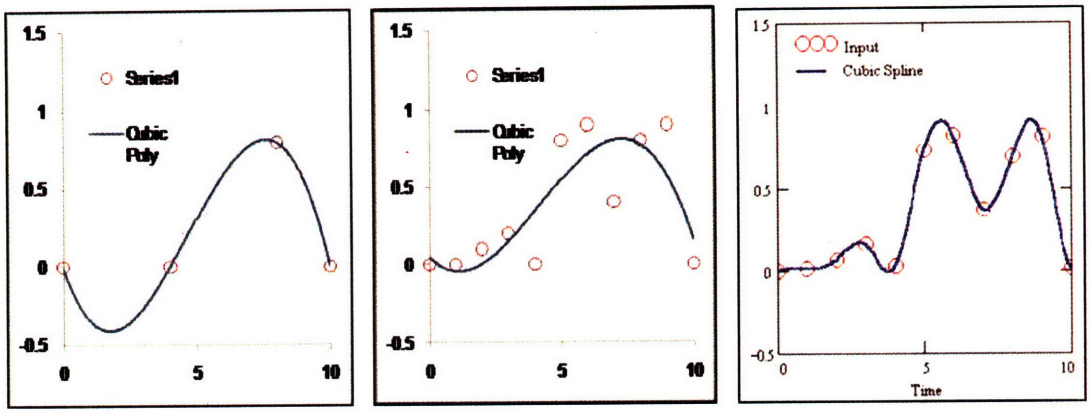
The capacity and VC dimension can also be defined for this regression-type model. Reference [9] discusses different capacity metrics for regression models, considering convergence characteristics and noisy data. But for the purposes of this section, it suffices to state that a regression model can shatter a set of points if it can exactly replicate any combination of their real-valued outputs. And as in classification, a model which can shatter h points has VC dimension of h.

For example, a cubic-polynomial model for equation 6.12

$$C_D = \sum_{j=0}^{3} \theta_j Re^j \tag{6.13}$$

can exactly reproduce the output of 4 points but not more, as depicted in figure 6-3. Therefore $h = 4$ for a cubic polynomial and by trivial extension for any polynomial of order n we have $h = n$. The figure also shows a cubic spline shattering 11 points and as a spline can shatter any number of points, its capacity is infinite.

Note that there is a strong correlation between the capacity and the number of free parameters in these simple examples. For instance, a natural spline defined by l points has l coefficients. For nonlinear models the correlation is not exact (some



(a) Poly, l=4 (b) Poly, l=11 (c) Spline, l=11

Figure 6-3: Illustration of shattering for regression. A cubic polynomial can shatter 4 points (a) but not more (b). On the other hand a cubic spline can shatter any number of points (c).

parameters may be less sensitive than others or redundant) but often the number of parameters does indicate the capacity. Therefore, model's with fewer parameters will typically have lower VC dimensions and therefore smaller VC gaps. Thus we achieve an Occam's Razor [57] for models: when dealing with limited examples the model with fewer parameters is preferred.

6.2.5 Influence of the Internal Model Size: N

Next we discuss a common alternate *á priori* predictive performance metric in the physics-based model literature; the size N of the internal variable set χ. When using physics-based models, N is often related to the performance of the model, but it is important to note that this parameter is only positively correlated with one element of the model risk.

The risk can be broken up into three components. First is the VC gap W_η measuring the generalization error. Next, the empirical error in a computational model may be broken down into two elements: numerical error and modeling error [48]. Of these three elements, increasing N will decrease only the numerical error.

The numerical error is due to the finite parameterizations of the infinite set χ in equation 6.3. Increasing the size of the internal set increases the resolution which reduces the numerical error for a convergent and stable computational method. The

level of numerical error is estimated by systematically varying N and measuring the effect on the solution. Based on these studies an N-independent solution can be determined using Roache extrapolation [48].

However, the other elements of the true risk do not decrease with increased N. The modeling error is dependent on the choice of model equation $\mathcal{M}$ and modeling parameters θ while the generalization error is only dependent on the capacity h and sample data size l. Additionally, if the capacity and number of model parameters are linked to the size of the internal set then these errors will *increase* with increased N. More modeling parameters are more difficult to optimize and higher capacities lead to worse generalization. Additionally, the numerical error should always be minimized *á posteriori* by constructing an N-independent solution. Therefore we conclude that the capacity h is a more fundamental *á priori* metric for estimating the true risk of a given model.

6.3 Preliminary Conceptual Applications to Physical System Models

The optimality of a particular model is related to its cost-effectiveness, e.g.

$$\Psi \propto \frac{1-R}{C} \tag{6.14}$$

where R is the true risk and C is the cost. The best model is the one with the largest value of Ψ due to minimal cost and risk. As the true risk is unavailable we utilize equation 6.10 to develop the alternate form

$$\Psi \propto \frac{1/2 - R_{emp} - W_\eta(l,h)}{C} \tag{6.15}$$

where R_{emp} is the empirical risk and W_η is the VC confidence which monotonically increases with h/l as in figure 6-1. Equation 6.15 is meant only as a sketch and the exact form of an optimality metric varies from application to application, but the

goal of utilizing a cost-effective model is universal.

Using the optimality metric, we can perform a preliminary estimate of the properties of a number of iconic examples from fluid modeling. First, consider a Direct Numerical Simulation (DNS) code. The goal of DNS is to use the absolute minimum number of modeling parameters. In the limit of this ideal being achieved, the capacity of a DNS code is zero and any small number of tests are sufficient to validate the method and establish a general estimate of the true risk. Note that this does not mean the risk is zero, only that it can be easily determined. If the set X is deficient (i.e., a key input has been neglected) or if an inappropriate internal mapping $\mathcal{M}$ is used then the measured and true risk may be quite high. Additionally, as mentioned above, the cost of DNS simulations for realistic engineering flows are extremely high. Even in the limit of zero risk the optimality of DNS as a design model is extremely limited.

Next, consider a least-squares curve-fit of resistance data for a particular class of vessel with O(10) modeling parameters. The optimized empirical risk of such a model is still likely to be larger than the (hypothetical) DNS simulation. However, if the data saturation is large ($l >> h$) then the performance is likely to generalize well *to members of the same class as the training set.* This is a critical limitation. Recall that the expressions for the VC confidence is derived based on the assumption of *i.i.d.* training data. Any predictions outside the training set (ie, using mono-hull data to predict trimaran resistance) have unbounded generalization risk, a fact that is well to remember among engineers and venture capitalists alike. Despite this limitation, the cost of the model is extremely small and so the optimality for making predictions within the scope of existing data may be sufficiently high.

However, greater optimality is certainly possible. What is required is a significant reduction to the cost of first-principle physics-based models with reasonably small increases in the generalized risk. This requires that the empirical error of fast physics-based models be mitigated through learning from available examples. This requires adding learning capacity to the physics-based model, but controlling that capacity level to minimized the dependance on training data. This leads to a second, more

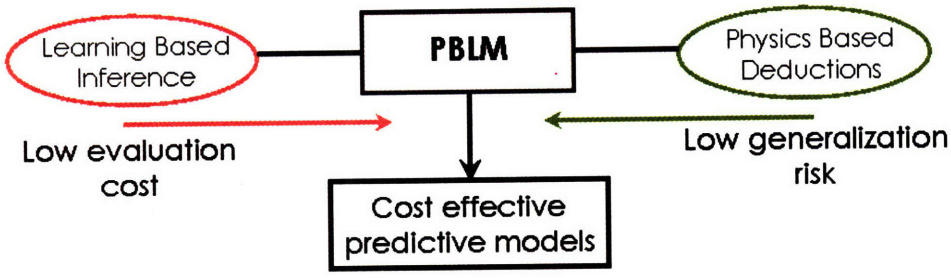


Figure 6-4: Illustration of physics-based learning model which combines physics-based deductions with data-based inferences.

informed request for a mixed physics-based learning-based model, as pictured in figure 6-4. Such a model minimizes evaluation time, estimated risk, and capacity thus providing optimal system predictions. General architectures and specific examples of such physic-based learning models are discussed in chapter 7.

To conclude this chapter, consider a fluid model with an extremely large capacity and $O(10^3) - O(10^8)$ modeling parameters which are changed from evaluation to evaluation based on a secondary model. As risk is a function of the parameters θ (equations 6.6 and 6.7) any evaluation of the empirical risk of the first model alone is invalid. However, the secondary model has an unbounded capacity and an unknown number of training examples guiding its behavior. The risk of the predictions must therefore be labeled as complectly unbounded regardless of the loss on the training data.

In fact, we have just described any CFD code which relies on the user to generate the numerical grid for each new test (or any other parameter for that matter). While many programs and rules of thumb exist for the generation of numerical grids, most codes require a large amount of user guidance and even after accumulating years of experience simple mistakes are likely. This not only adds to the risk but also to the model development and interface costs. Additionally, the capacity of the user is unbounded and large changes in the model output are achievable based on the user's predetermined notions of the physical system and grid aesthetics. In effect, no assurances can be made for the "CFD+user" model.

However, significant progress has been made in limiting this unintended (and often unnoticed) source of model risk. High-order numerical methods, especially prevalent

in the finite element approach ensure the best possible solution for a given space tessellation [42]. Note however, that this does not necessarily translate to a grid insensitive solution. The use of Cartesian-grid approaches, particularly the new and well validated approaches described in earlier chapters, directly limits the number of adjustable grid parameters. These Cartesian-grid approaches are thus ideal for the direct modeling of physical systems or as part of a mixed physics-based and learning based approach.

Chapter 7

Physics-Based Learning Models

Learning-based models offer an alternative to physics-based simulations which are extremely fast; individual design evaluations taking from seconds to minutes. However, such models require example data to learn from and have limited accuracy when interpolating between this data. Regions of the design space for which there are few or no examples have a high risk of prediction errors [63]. Additionally, more complex physical systems require more elaborate learning models which must be trained with correspondingly larger sets of examples [26]. These models are custom-tailored for each learning problem and are not easily extended to new problems and designs.

This chapter contributes to the field of physics-based learning models (PBLMs) through the development of a new general approach to the mixture of physics-based and example-based information. First, the chapter continues the model classification of chapter 6 with a more detailed treatment of model units and architectures. Next, this advanced framework is used to analyze existing PBLMs in the ocean engineering literature, ranging from Froude scaling [16] to modern computational ship maneuvering predictions [22],[10]. Finally, the chapter introduces a new PBLM architecture based on 'physics-based basis functions' which enable fast and accurate predictions of new systems. The new method is tested for two ship flows; the waterline of the Wigley hull [55] and the breaking wave predictions of chapter 5.

7.1 Model Units and Architectures

Chapter 6 introduced the goal of broad categorization of model types and functionality. The metrics defined in that chapter provide the means to analyze broad categories of models. Examined from this perspective, Direct Numerical Simulation (DNS), which is completely physics-based, is found to be excessively computationally expensive. Direct curve-fitting, which is completely learning-based, is found to be excessively dependant on training data. [1] Neither extreme is ideal for developing models of physical systems.

A mixed physics-based and learning-based model has the potential to overcome these limitations. In order to develop such models a refined analysis of modeling processes is required. This section presents a conceptual framework for this analysis, defining abstractions of fundamental modeling units and architectures built from those units. In this way arbitrarily complex models may be diagramed, analyzed, and eventually optimized.

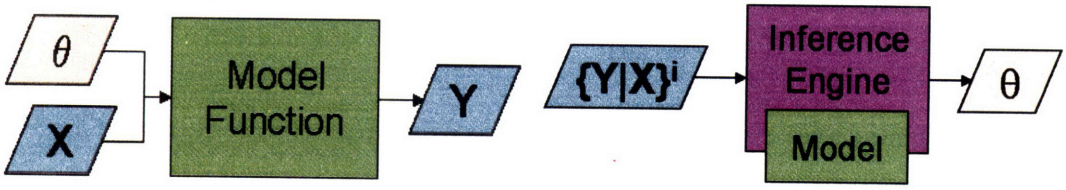
7.1.1 Mapping and Inference Units

The first basic unit in the model is the mapping unit, a sketch of which is shown in figure 7-1(a). As introduced in section 6.1.1, the model function f generates the output Y given the input X and modeling parameters θ.

These model parameters are, by definition, initially unfixed and required some additional process to determine them; the inference unit of figure 7-1(b). In the inference unit the inference engine is used to generate optimized model parameters based on a given model function and a set of examples $\{Y|X\}^i$. The goal of the unit is to choose the parameters that allow the model function to best replicate the examples.

The details of the inductive process are not fundamental to this thesis and learning is essentially treated as a 'black box' as in figure 7-1(b). The key point is that the

[1] Additionally, fitted grid methods, which use 10^8 numerical gridding parameters, are found to have excessive generalization error.



(a) Mapping Unit (b) Inference Unit

Figure 7-1: Flow chart conceptualizations of the generic mapping unit (a) and the inference unit (b). Note that the inference unit incorporates the model function into the inference process.

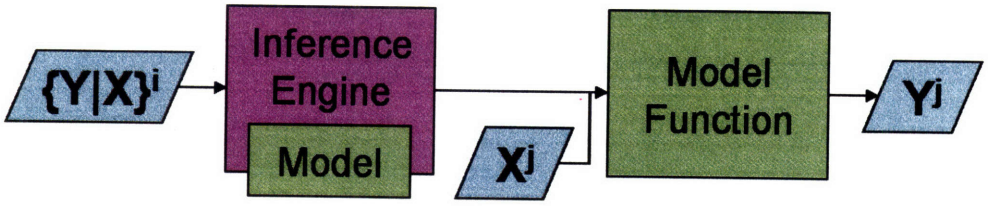
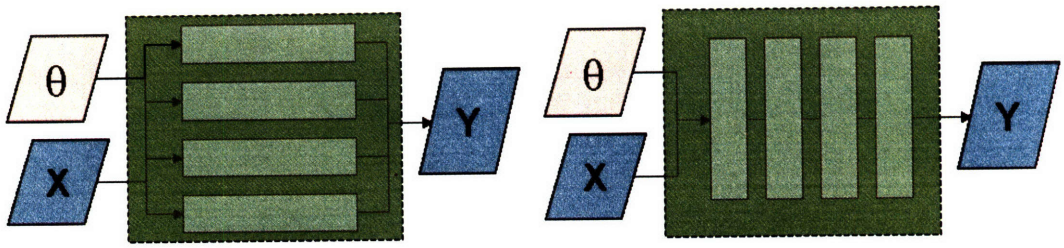


Figure 7-2: Information diagram for the basic learning model architecture. The example data $\{Y|X\}^i$ is used to fix the model parameters after which new output values Y^j can be determined for any set of new input values X^j.

mapping unit may not evaluate before the inference engine determines the model parameters θ and that this requires a set of examples $\{Y|X\}^i$. Therefore, any model which makes use of learning must combine the units above into the basic learning model architecture, shown in 7-2. Putting the two units together in this fashion forms a complete information diagram, where information is propagated from left to right without any gaps. The two models have been joined directly through the model parameters θ, which no longer appear explicitly. This graphical elimination of the free parameters represents the completion of the learning model, which is now capable of evaluating a new set of output values Y^j for new input values X^j.

Recall from section 6.2 that each unit is characterized by a set of metrics. Model functions have an associated interface and evaluation cost C and capacity h. Inference engines have an associated learning cost and an acquisition cost proportional to the number of examples l. In order to analyze architectures such as the basic learning model in figure 7-2 we must determine how the metrics of the model units influence the metrics of generic model architectures.



(a) Parallel Architecture (b) Serial Architecture

Figure 7-3: The generic parallel and serial model architectures. Combining simple modeling units in parallel or serial results in a compound unit with adjusted properties.

7.1.2 Generic Architecture Configurations

Analysis of current flow in circuits, heat transfer in mechanics, and many other branches of science define two essential methods of combining individual units to form a larger system of units. In those fields and in information diagrams the units may be combined in series or in parallel, pictured in figure 7-3. In the parallel architecture, different modeling units are each fed a subset of the same input and each generates their own subset of the output. In the serial architecture, the output from one model is piped directly into the input of the next. The dashed boxes outlining each group of modeling units is the effective compound mapping unit. Note that further recursion is possible; any small box in the figure could also contain a set of component units.

The properties of the compound unit depend on the architecture and properties of the component units. The arithmetic of these properties is straightforward; for the cost we have

$$C_p \geq \max_i C_i \tag{7.1}$$

$$C_s \geq \sum_i C_i \tag{7.2}$$

where C_i is the cost of the i unit, C_p is the cost of a parallel architecture and C_s is the cost of a serial architecture. These bounds are self-evident, and are met exactly if there is no additional cost for the architecture (such as splitting the set and piping the variables). Clearly, the parallel architecture offers the advantage of speed if the

processes can actually be run simultaneously.

Recall from section 6.2.4 the definition of capacity for a regression model is the number of points the model can shatter. Using the same indexing notation the capacity is bounded by

$$h_p \leq \sum_i h_i \tag{7.3}$$

$$h_s \leq \prod_i h_i \tag{7.4}$$

where h_f is the capacity of the final unit in the series. The bounds may be motivated conceptually through simple polynomial models for a scalar problem (f(x)=y). If the component models are

$$f_1 = \theta_1 x \tag{7.5}$$

$$f_2 = \theta_2 x^2 \tag{7.6}$$

$$f_3 = \theta_3 x^3 \tag{7.7}$$

such that $h_1 = h_2 = h_3 = 1$ then parallel and serial linking gives

$$f_p = \theta_1 x + \theta_2 x^2 + \theta_3 x^3 \tag{7.8}$$

$$f_s = \theta_3 \left(\theta_2 \left[\theta_1 x\right]^2\right)^3 = \theta_s x^6 \tag{7.9}$$

where the new parameter $\theta_s = \theta_1^6 \theta_2^3 \theta_3$. Because the components are orthogonal the capacity of the parallel architecture is the sum of the components, $h_p = h_1 + h_2 + h_3 = 3$. Clearly, if the components are linearly dependant the parallel architecture will not achieve this upper bound, instead there will be some level of redundancy. The capacity of the serial architecture is $h_s = 1$ because there is only one effective parameter to optimize $theta_s$. If the component models are

$$f_1 = \theta_{11} x + \theta_{12} x^3 + \theta_{13} x^4 \tag{7.10}$$

$$f_2 = \theta_{21} x + \theta_{22} x^2 \tag{7.11}$$

such that $h_1 = 3, h_2 = 2$ then linking the models in series gives

$$f_s = \theta_{21} f_1 + \theta_{22} f_1^2 \tag{7.12}$$

$$= \theta_{s1} x + \theta_{s2} x^2 + \theta_{s3} x^3 + \theta_{s4} x^4 + \theta_{s5} x^6 + \theta_{s6} x^8 \tag{7.13}$$

where θ_{si} are the products of the appropriate coefficients (ie $\theta_{s2} = \theta_{11}\theta_{22}$). In this examples the terms generated when the second unit operates on the first unit are all orthogonal, therefore the capacity is $h_s = h_1 h_2 = 6$. Thus this polynomial example shows that the capacity upper bound is a summation for parallel architectures (achieved when the units are orthogonal) and a product for serial architectures (achieved when the units operating on each other generate orthogonal terms).

Note that in this example, component models each have their own set of parameters $\theta_{1i}, \theta_{2i}, \ldots$. When component models are identical and given a shared set of learning parameters the serial architecture becomes a recursive model.[2] In this case, the capacity of the compound model is bounded by the capacity of the repeated component.

Using bounds 7.1-7.4 we can propagate the metrics characterizing a set of well understood component models up to the level of the complete architecture. If a potential flow code f_{pf} with capacity $h_{pf} = 5$ and cost $C_{pf} = 60s$ is linked to a linear regression model for post-processing f_r with capacity $h_r = 25$ and cost $C_r = 1s$ in serial then the total capacity is not more than 30 and the cost is at least 61 seconds. If the potential flow code is replaced by a RANS code with capacity 10^5 and cost $10^4 s$ then those values will dominate the total system. This approach is applied to the analysis of existing model architectures in the following section.

7.2 Example PBLM Architectures

As discussed in chapter 6, the optimality of a model is related to its cost effectiveness. As such, decreasing cost and risk is always advantageous. Capacity, on the other

[2]In parallel the result would be a redundant model.

hand, is not simply good or bad; it represents a design decision. Additional learning capacity results in higher generalization risk but may lead to lower empirical risk as well. Therefore, to design the best compound model for a system requires adding low capacity physics-based models to capacity-controlled learning models. Using appropriate model units and architecture results in a compound model with greater optimality than any of its components.

This section explores different pre-existing physics-based learning models (PBLMs) from the literature. Examples range from classical (Froude [16]) to modern computational vortex induced vibrations (Mukundan [38]) and ship maneuvering (Faller [10]) predictions. Examining these models identifies key features of successful PBLM architectures and motivates the need for a general approach for mixing physics-based and learning-based models.

7.2.1 Semi-empirical Models

The most common type of PBLM found in engineering is the so-called semi-empirical model. Such models are built upon a system-specific analytic mapping with a number of free parameters. Examples from the same class of systems are used to determine any free parameters, and the diagram for this process is shown in figure 7-4. Note that this is simply a basic learning model (of figure 7-2) which incorporates a system-specific functional mapping. As in the basic case, available training data $\{Y|X\}^j$ is input to the inference engine which optimizes the model parameters θ. Once those parameters are determined the model function can be evaluated for new input cases X^j and generate the model predictions Y^j. Note that the first component is only run once for a given data set, after which any number m of new model evaluations may be performed.

The total cost of the model is the cost of gathering the data set, plus the cost of learning in the inference engine, plus the cost of the m evaluations. The total capacity of the model is bounded by equation 7.4. Often a least-squares method is used for the inference engine in these models in which case the compound capacity equals the model function capacity. The empirical risk depends on the complexity of

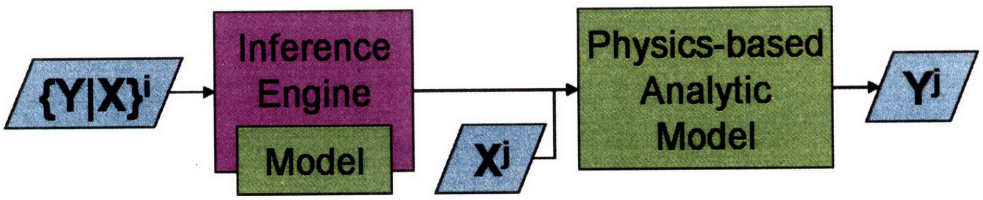


Figure 7-4: The semi-empirical model. The inference engine runs once to set the parameters, after which any number of model function evaluations are run.

the problem class and simplification level. When a large amount of sample data is available within the class the VC gap is very small and the empirical risk is expected generalize well as long as the model is faithfully applied only to problems within the training set.

Semi-empirical models abound in the fluids literature; boundary layer theory, wake theory, and others. One classic example is the model for the friction drag of a flat plate at turbulent Reynolds numbers,

$$C_f = f(Re|\theta) = \frac{\theta_1}{(\log Re - \theta_2)^2} \tag{7.14}$$

where C_f is the friction drag coefficient. Granville [19] showed that this functional, first adopted by the International Towing Tank Conference (ITTC) in 1957, can be derived for a turbulent flat-plate boundary layer from physics-based arguments. The free parameters of this equation are determined using resistance data taken from a fully submerged flat plate. Setting the parameters at $\theta_1 \approx 2$ and $\theta_2 \approx 0.075$ results in model output (C_f) which agree with the experimental data to within 9% over a wide range in the input ($Re = 10^5 - 10^{10}$). The generalization risk is extremely low within this range because of the abundance of data relative to the model capacity ($h = 2$, $l/h > 200$). Therefore the total risk is around 9%, and since the cost is nearly zero this model for skin friction is highly optimal.

The semi-empirical approach is a hallmark of engineering and should be used when possible. However, this class of model is not suitable for the accurate prediction of a general system's behavior. These models are developed by simplifying the governing equations with system-specific assumptions to the point where they become analytic.

This broad set of assumptions limit the model's applicability to a small class of problems, such as the specific case of skin friction prediction given above. Many physical systems are too complex or too poorly understood to make suitable simplifications and more sophisticated PBLM methodologies are required.

7.2.2 Froude Scaling of Ship Forces

Froude scaling represents a more sophisticated physics-based learning model, which allows for detailed examination. In the early 1870s, Froude proposed the decomposition of ship resistance into two parts, the frictional resistance and the residuary resistance [16]. In modern terminology Froude decomposition states that there are two primary scales on which the ships resistance depends and that these factors act independently. These two scales are the viscous scales parameterized by the Reynolds number and the gravity wave scales parameterized by the Froude number. In other words, the resistance coefficient C_T is given by

$$Y = C_T = f(R_e, F_L, X_{geo}) \approx C_f(R_e) + C_r(F_L, X_{geo}) \tag{7.15}$$

where C_f and C_r are the frictional and residuary resistance coefficients and X_{geo} are the inputs defining the ship geometry.

It is well known that this decomposition is the basis for model-basin testing, and has been used since Froude's first basin in 1868. The breakdown of resistance into these two parts allows the resistance results from scale-models of a ship to predict the full-scale ship results, and the process is depicted in figure 7-5. The compound model is made up of three main parts: the model for C_r the model for C_f and the inference engine used to determine the free parameter in the frictional model.

Because C_f does not depend on the geometry or Froude number, these parameters are fixed using data from (flat)plate tests, $\{Y|X\}_p^i$ in the figure. As detailed in the previous section this data is easily obtained and the friction model parameters are optimized for a full range of Reynolds numbers. The full-scale ship inputs X_s are used by the frictional and residuary models, in parallel, to determine to full scale output

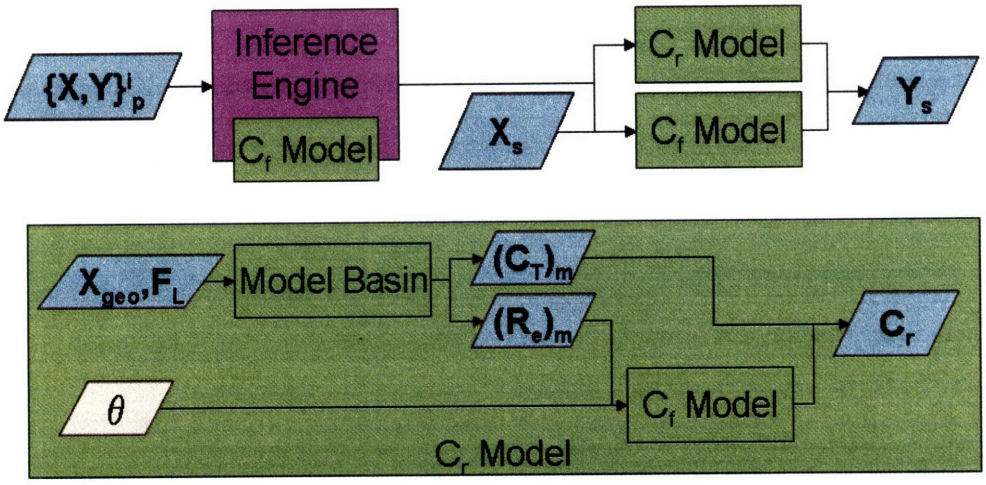


Figure 7-5: Learning diagram for Froude-scaling of ship resistance. The major components are shown in the upper figure and the lower figure details the workings of the residuary resistance C_r model.

via equation 7.15.

The internal workings of the residuary model are shown on the lower portion of the figure. The model-basin itself acts as a model unit, mapping the Froude number and geometry to the model-scale total resistance and Reynolds number, $(C_T, R_e)_m$ in the figure. Equation 7.15 and the frictional model are then used to determine the Reynolds number independent residuary resistance.

There are two key features of this model that are worth revisiting from a functional point of view. First note that model-basin mapping is a measurement of a true physical system, and therefore there is little to no capacity in that unit. Thus, the sole source of capacity is in the frictional mapping of equation 7.14 with $h = 2$. The second key feature is the split of the input space; Froude's decomposition. The frictional mapping is only dependent on Reynolds number (not Froude number or geometry) allowing for the use of a vast independent training set (the plate data). As such, the cost is effectively limited to the cost of the model-basin experiments and the VC gap W_η approaches zero.

This model has obviously been a historic success, but the simple nature of the decomposition leads to empirical errors. There are various methods employed to combat these errors (basin specific adjustment factors) but always at the cost of

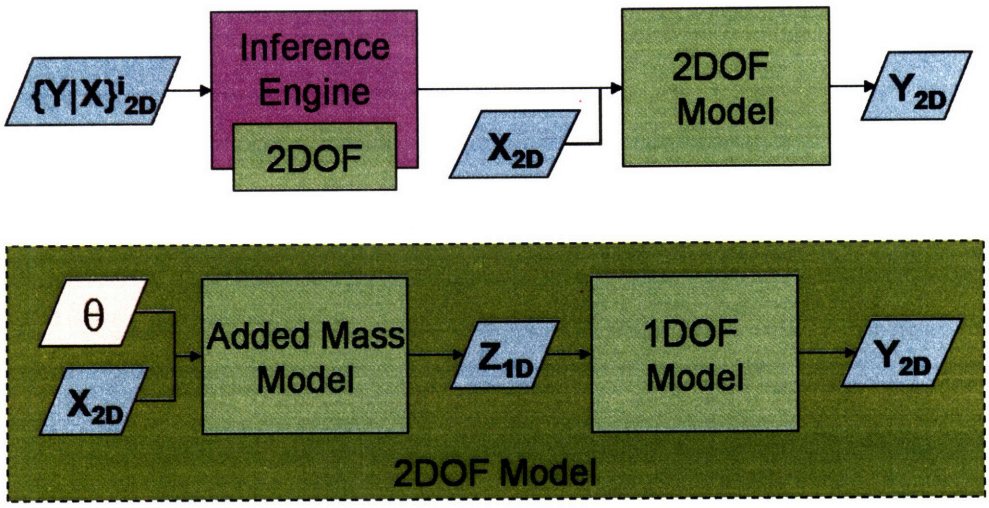


Figure 7-6: Learning diagram for a 2 degree-of-freedom (DOF) vortex-induced-vibration model based off of a 1DOF model and a model trained to output the effective 1D added mass.

increased generalization error. There is also the large cost (time and monetary) of running model-basin tests. These factors and the increased availability of computers have resulted in the industry shift towards computational models for ship resistance as well as other ocean engineering systems such as marine risers.

7.2.3 Modern Computational PBLMs

Modern examples of PBLMs typically involve the extension of an existing physics-based computational model to a new genre of problems. This extension is accomplished by incorporating new physics-based and/or learning-based models and training the model with existing experimental results. Two recent examples from the literature are sketched to demonstrate current efforts in this direction.

First is the work of Mukundan [37] on the extension of a single degree of freedom (1DOF) vortex-induced-vibration (VIV) computational model of marine riser motions, to the prediction of two degree of freedom (2DOF) motions. While there are many aspects to this work, the physics-based learning model is sketched in figure 7-6.

The higher level diagram shows the use of existing 2DOF experimental data for the VIV of marine risers to train the 2DOF model which can then make predictions

for new input cases. The second level shows the internal workings of the 2DOF model, which consists of two parts. First the 2DOF inputs X_{2D} are fed into an added mass model, which outputs a set of effective 1DOF inputs Z_{1D}. These inputs are used by the 1DOF solver to predict the VIV characteristics. The 1DOF solver is a well established model with no additional free parameters, therefore only the added mass parameters must be determined.

As the example data is for a 2DOF system the entire 2DOF model must be incorporated into the inference engine. The cost of the added mass model is insignificant, but the 1DOF model is somewhat costly and highly nonlinear and this is the limiting factor for this PBLM architecture. In order to moderate the learning costs, a relatively small set of parameters θ are employed. By clever choice of these parameters, the empirical risk of the 2DOF model is minimized but a reduction in learning cost may enable even greater optimality of the compound model [37].

A second PBLM found in the literature is a Recursive Neural Network (RNN) model of ship maneuvers published by Faller and Hess. The underling concept is to extend a steady-state physics-based flow solver to predict unsteady motions without excessive computational cost. The work includes predicting the effect of control surfaces [10] hull shape changes [11] and environmental effects [22] on ship maneuvering. A sketch of the learning model is given in figure 7-7, though even this simplified diagram requires some effort to analyze.

The basic structure is pictured at the top level of the diagram, where a set of measured full-scale maneuvering data is fed into a inference engine for the RNN. Once trained, the RNN predicts the time history of ship motions based on the geometry and input maneuvering commands among other factors.

The second level is a more detailed view of the inner workings of the RNN which is made up of two parts. The geometry and current orientation X_G, X_O are fed into a neural network (NN) force prediction model. The forces Y_F, along with the geometry and orientation and old values of these parameters, are fed into a motion prediction model which updates the orientation Y_O. These output are fed back into the input of the recursive model and the cycle repeats. The single recursive modeling unit thereby

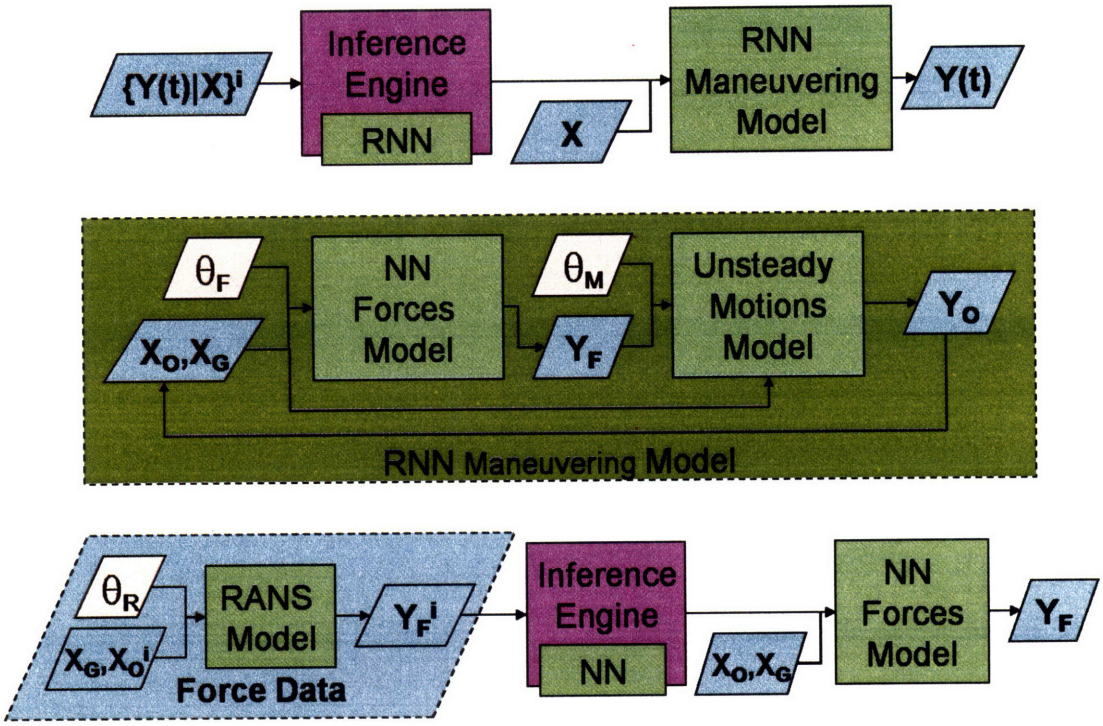


Figure 7-7: Learning diagram for the ship maneuvering RNN.

predicts the full time history of the ship motions.

There are two sets of model parameters in the second level diagram; the motion set θ_M and the force set θ_F. The motion set are learned using the motion data, as depicted in the level above. Because the model is recursive, each time step acts as a new set of data, effectively controlling the generalization error of the motions model. The low evaluation cost of the NN force model helps moderate the training cost of the complete RNN in contrast to the VIV example. However, the force model parameters must be determined first.

The training of the force model is detailed on the lowest level of the figure. A set of example data are used to train the NN model to predict the forces based on the ship orientation. The authors use a physics-based Reynolds-Averaged Navier-Stokes (RANS) flow model to generate this data, which itself has a set of modeling parameters θ_R. As stated previously, such flow solvers typically have millions of model parameters and internal variables. Therefore, they require an experienced user, the evaluation cost is high, and the generalization error is unbounded.

The authors show this physics and learning-based model is much more optimal

than an unsteady RANS simulation. However, the high cost and generalization risk of the imbedded steady RANS model does propagate upward, limiting the optimality of the compound model. Using reduced capacity physics-based models eliminates this source of risk and cost.

Despite some limitations in cost and learning capacity, both of these examples show the promise of physics and learning based computational models for ocean engineering systems. However, as shown in figures 7-6 and 7-7, both architectures are complex and system-specific. Additionally, both require a set of training data for the system to already exist. As such, neither other these architectures is applicable to new general systems. The need for such a architecture is addressed in the following section and the key innovation is the concept of physics-based basis functions.

7.3 New PBLM Approach: Physics-based Basis Functions

The goal of this chapter is to achieve a highly optimal PBLM through low computational cost, minimal empirical risk, and controlled generalization risk. In addition, the architecture should be generally applicable to different types of physical systems, the model units should be modular (and capacity controlled), and the PBLM should not depend on pre-existing data.

As is often the case is science, asking the right questions is the most difficult step; with many of the answers following naturally. In the case of PBLM design, the previous sections and chapters have made us aware of these goals and metrics and most of the solutions are straight forward. With our eyes on the goal of reduced *total* cost, not just evaluation cost, we can easily minimize the training costs which plagued the VIV predictions of section 7.2.3 by avoiding such high-cost architectures. Instead we will prefer cases where all learning is done after the data is available, as in the ship maneuvering method. Additionally, now that we are aware of the excessive capacity and interface costs of fitted-grid methods we can incorporate better physics-based

model units such as Cartesian grid methods.

The remaining problem is how to *generally* incorporate information from both learning-based and physics-based methods? One option is to create a model architecture which is adaptive, such as in the field of genetic programming [20]. In this 'unsupervised' learning method the architecture itself would evolve based on some supplied fitness criterion. However, unsupervised learning is better suited to modeling human intelligence than physical systems as it has larger costs and risks [20]. Instead, a simple fixed architecture which is highly general will be explored in this section; that of physics-based basis functions. To present that method we first revisit direct curve-fitting models.

7.3.1 Direct Curve-Fitting Revisited

As stated above, the remaining problem is how to *generally* incorporate information from both learning-based and physics-based methods? The most obvious method is to use a physics-based model to generate a set of example data, and then preform a direct curve-fit on that data. A computational version of this approach is the basis for the resistance prediction unit of the maneuvering model, the lowest level in figure 7-7. However, this is fundamentally the same as the classic approach of preforming a set of experiments and fitting a curve through the results. More computationally advanced methods are used to generate the data and preform the curve-fit but the game hasn't changed and neither have the limitations. As in the classic case, the final model will only be reliable if you have completely mapped out the design space with a reliable prediction tool.

In the terminology of information theory; the risk of the compound model will be a function of the empirical and generalization risk of the components. Therefore, if an advanced neural network (NN) with capacity $h > 25$ is used, the model will require something like $l = 400$ data points to reduce the generalization risk. Additionally, computational models with very low empirical risk have proportionally high computational costs. For a ship-flow CFD simulation $C = 1day$ is reasonable, meaning the model will require a year of computational effort before confident predictions can be

made.

The dependance of the direct curve-fit method on large amounts of high-fidelity data is it's shortcoming. This is common knowledge in the engineering community and has already been discussed in section 6.3. *It is a weakness in the architecture itself* which is why the shortcoming is not avoided by using RANS codes and NN learning machines. The architecture itself must be modified to alleviate this limitation.

7.3.2 Semi-Empirical Models Revisited: Basis Functions

The architecture of the semi-empirical model (fig 7-4) is only subtly different than that of the direct-curve fit; the generic curve-fitting tool (spline, NN, etc) being replaced with a system-specific PBLM. While this approach is not generally applicable to new and complex engineering systems, its advantage is that a physics-based learning tool requires much less capacity to achieve similar levels of empirical error compared to a generic learning tool. This is not surprising. A generic curve-fitting tool doesn't know anything about the system other than the $\{Y|X\}^i$ data it is given. It is equally capable of fitting stock market trends as it is of fitting resistance curves. The difference between a generic model and the semi-empirical model are the basis functions used inside the model.

The polynomial example regression model from section 6.2.4 for resistance takes the form

$$C_D = \sum_{j=0}^{3} \theta_j Re^j \tag{7.16}$$

where Re is the Reynolds number and θ_j are the model parameters (four in this case). The model is linear in the coefficients (no non-linear combination of *theta* is present) which allows simple learning methods to be used and ensures unique optimum values of the coefficients for a given set of sample data. The basis functions for this model are the powers of the Reynolds number, i.e. $1, Re, Re^2$ and Re^3. However, this choice was somewhat arbitrary; any orthogonal basis could be used such as a Fourier series or Tchebychev functions.

On the other hand, the basis functions of a semi-empirical model are specific to

the engineering system. Equation 7.14 in section 7.2.1

$$C_f = f(Re|\theta) = \frac{\theta_1}{(\log Re - \theta_2)^2} \tag{7.17}$$

gives the ITTC model for the resistance of a flat plate. The model is still only a function of Re but the form of the function is more complex and the model is nonlinear in the parameters. There is no clear 'basis' which the function is a part of and it is not intended to fit any other physical system trends. This particular function was chosen by the ITTC from physical arguments and therefore *its form includes additional information*, making it more robust to limited training data.

The comparison between the models highlights the point that a semi-empirical model gains it's advantage over generic models through advanced basis functions. It also reminds us that semi-empirical models are not optimum for studying new and complex systems because they require explicit knowledge of the basis functions for the physical system's solution - something which is virtually never available. A method of constructing physics-based basis functions for new and complex systems would further the goal of developing optimal PBLMs.

7.3.3 Construction of Physic-Based Basis Functions via Intermediate Models

The role of basis functions in a learning model can be thought of as preconditioning the input set X such that it correlates better with the output set Y. In the polynomial resistance model, the input $X = \{Re\}$ is preconditioned to the intermediate set $Z = \{1, Re, Re^2, Re^3\}$ in the (rather blind) hope that a linear combination of this set will correlate with the observed resistance trend. In other words the model

$$Y = \theta \cdot Z(X) \tag{7.18}$$

has it's model parameters optimized by minimization of a loss function. In the limit of Z including all of the positive and negative powers of Re this correlation would

be ensured because the capacity h of a complete set of polynomial basis functions is infinite. However, we know that predictions of high capacity models can not be trusted when given limited data because of the generalization risk.

On the other hand, if the conditioned set Z is based on physical knowledge of the system, then far fewer elements are needed. This corresponds to a much lower capacity and generalization risk for a given amount of data. Consider the case of a phase-shifted harmonic oscillator as a toy example.

$$y = a\sin(kx + \phi) \tag{7.19}$$

Given only a few sample points $\{y|x\}^i$ from this system we might use a polynomial basis which would result in large predictive errors. However, if we knew from physical arguments to precondition x such that $Z = \{\sin(kx), \cos(kx)\}$ then we could fit the model exactly

$$y = \theta_1 \sin(kx) + \theta_2 cos(kx) = a\cos\phi\sin(kx) + a\sin\phi\cos(kx) \tag{7.20}$$

where the few sample points would allow us to establish the coefficients $\theta_1 = a\cos\phi, \theta_2 = a\sin\phi$. This is the advantage of semi-empirical models over generic curve-fitting stated yet another way, but how can this concept be extended to general systems?

For our simple harmonic oscillator we might assume that a model of the form

$$\tilde{y} = b\sin(kx) \tag{7.21}$$

is at hand. Direct comparison of the model to the data would reveal error because of the approximate nature of the model, but this error can be corrected by learning from data. One method of correction is the so called semi-parametric or nuisance model. In this method, the difference $y^i - \tilde{y}(x^i)$ is fit with a generic learning model to compensate for the error. A slight modification proposed in [53] is to fit the simple model and generic model simultaneously. In our example, this would have the example data $\{y|x\}^i$ fit by a linear combination of $Z = \{\tilde{y}, 1, x, x^2, x^3, \ldots\}$. The reference shows

simple examples which demonstrate that this modified semi-parametric model can more accurately fit data because it allows all the basis functions maximum flexibility in reducing the model error.

Semi-parametric models thus generally incorporate information from both training data and physics-based system models. However, the semi-parametric model will only reduce the overall risk of the model if the intermediate model is a very good description of the system. In our toy example, large values of ϕ will render the model no better than a generic learning model. More physic-based functions are needed in the basis to improve the model.

There are many ways to develop a basis from a given function, but one of the simplest is the derivative basis from Taylor. In our example the Taylor basis is

$$Z = \{\tilde{y}, \frac{\partial \tilde{y}}{\partial x}, \frac{\partial^2 \tilde{y}}{\partial x^2}, \ldots\} \tag{7.22}$$

$$= \{b\sin(kx), kb\cos(kx), \ldots\} \tag{7.23}$$

which recovers the exact function by linearly combining the first two terms. This process of building a larger-set of physics-based basis functions through derivatives is similar to the process of increasing the order of accuracy of a numerical method through use of higher order polynomials. Supplementing a generic model with a set of physics-based basis functions reduces the required capacity of the learning model while increasing its physical relevance and accuracy.

The drawbacks of using physics-based basis functions are the cost of the intermediate model and it's derivatives. The first element of that cost is development. Luckily, fast simple engineering models abound for many systems. In addition there are many analysis tools which apply to general systems after some level of approximation. Examples from ocean engineering include potential flow methods, and the 2D+T models of chapter 5. These and other similar models may be used as the intermediate models which precondition the input X before learning is done on any relevant ocean engineering system.

Development of an exact derivative basis would be significantly more complex

and the evaluation cost would likely be much greater. However, the derivative basis is easily approximated numerically. For example, instead of basis 7.22 we can use the difference basis

$$Z(x^i) = \left\{ \tilde{y}(x^i), \frac{\tilde{y}(x^i + \Delta x) - \tilde{y}(x^i - \Delta x)}{2\Delta x}, \right. \tag{7.24}$$

$$\left. \frac{\tilde{y}(x^i + \Delta x) - 2\tilde{y}(x^i) + \tilde{y}(x^i - \Delta x)}{\Delta x}, \ldots \right\} \tag{7.25}$$

which (through linear combinations) is equivalent to the basis

$$Z(x^i) = \{\tilde{y}(x^i), \tilde{y}(x^i - \Delta x), y(x^i + \Delta x), \ldots\}. \tag{7.26}$$

the first two terms of which will fit to the function y for general values of Δx. Of course, this phase shifted basis is identical to the derivative basis in the limit of $\Delta x \to 0$ and is trivially computable using the known intermediate model $\tilde{y}$.

Therefore the phase-shifted basis is the one used in the remainder of this thesis, and it leads to a new computational PBLM architecture, the physics-based basis function (PBBF) model. The learning diagram for the PBBF architecture is shown in figure 7-8 which consists of three key components, a high-fidelity model, an intermediate model and a generic learning model. As stated above, the intermediate model is typically an existing low-capacity physics-based model of the system. Models such as potential flow or the Cartesian-grid 2D+T model are excellent candidates. The cost of the models C_I must be sufficiently low to allow fast generation of the physics-based basis functions.

The high-fidelity model is a low-capacity physics-based model which trades increased cost ($C_H >> C_I$) for increased performance. Candidates would be experimental measurements or resolved 3D Cartesian-grid solutions. The specific type of machine learner can be based on the application, but the learning model used in this thesis is the Support Vector Machine (SVM) proposed by Vapnik [64]. SVMs are notable because they are linear (with unique parameter solutions) and control their model capacity automatically [53]. Conceptually, this is done through selecting only

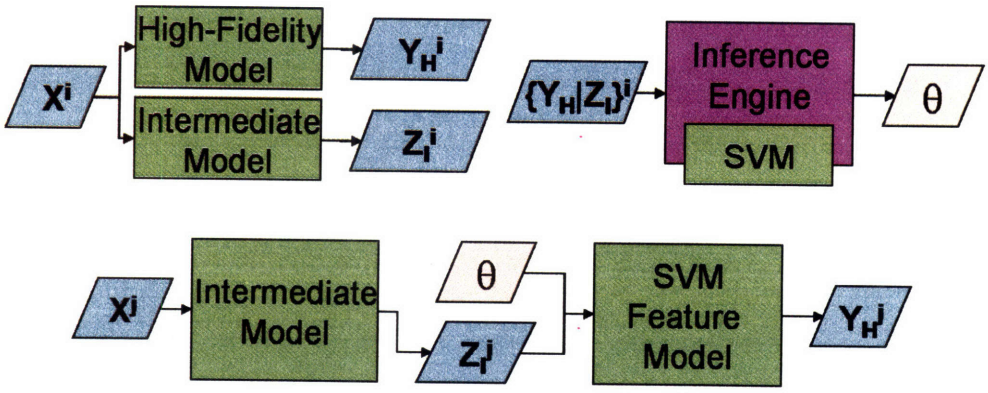


Figure 7-8: Learning diagram for the physics-based basis function architecture. Note that the model makes use of parallel components and learning takes place after model evaluations.

elements of the basis set which minimize the error and removing the others. However other generic learning methods could be used to within the PBBS architecture; polynomials, fourier series, Neural Networks, etc.

Summarizing figure 7-8; the high-fidelity and intermediate model are run in parallel to generate a data sets $\{Z_I, Y_H|X\}^i$. The intermediate model is treated as a preconditioner generating a phase-shifted basis which augments the SVMs generic basis in the manner of semi-parametric models. Note that the phase-shifted basis requires the computation of a number n of additional solutions to the intermediate model[3]. Thus for every one of the l high-fidelity points which are run, there are n intermediate model evaluations. Because the processes are in parallel the cost of the data generation is $C_{data} = max(C_H l, C_I ln)$ and as $C_H >> C_I$ the high-fidelity cost is likely to be the limiting factor.

Once the data is generated, a linear combination of the complete basis Z_I is fit to the high-fidelity data Y_H an in equation 7.18. Because the learning takes place after the physics-based models have run the learning cost is not dependant on the model evaluation cost. Also, because the model is linear the learning cost is negligible compared to C_{data}.

A new set of m input cases X^j are run through the intermediate model producing

[3]In general, n depends on the dimensions of X and the desirednumber of Taylor components in the basis.

new physics-based output Z^j. As this is independent of learning, this step too, may be done in parallel to the high-fidelity evaluations. Once trained, the learning model maps this intermediate output to the best estimate of the high fidelity solution Y_H^j, an operation of insignificant cost. Therefore, the total cost of m evaluations of the PBBS model is $C = \max(C_{data}, C_I mn)$ and the total capacity is $h = h_{SVM} + n$.

Two example applications of this architecture demonstrate its performance in fluids systems modeling; prediction of waterlines and breaking waves.

7.3.4 Waterline Predictions for the Wigley Hull

The physics-based basis function architecture introduced above is first used to predict the waterline elevation on the Wigley hull form. In this introductory example the high-fidelity and intermediate model are both taken from existing sources. Experimental measurements for the waterline elevation are used as the high fidelity model f_{exp},

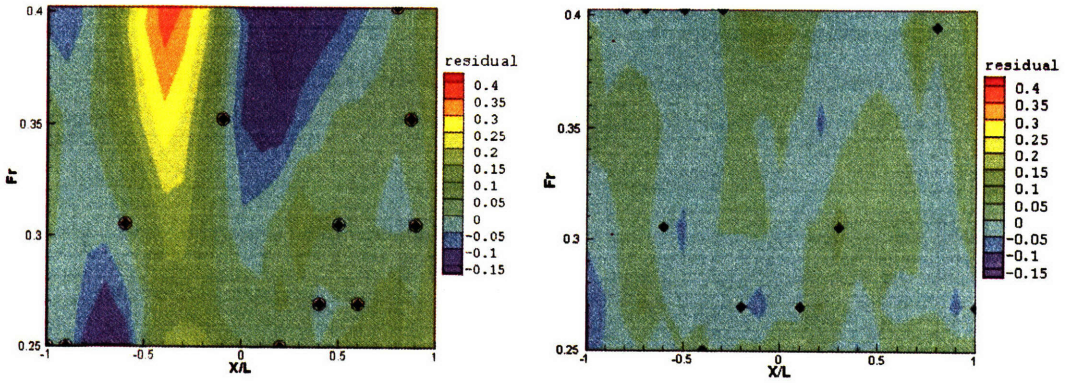
$$Y_H = \eta_H = f_{exp}(x/L, Fr) \tag{7.27}$$

where η is the elevation, x/L is the scaled longitudinal distance (L is the half length), and Fr is the length based Froude number. A potential flow method serves as the intermediate model f_{PF},

$$Y_I = \eta_I = f_{PF}(x/L, Fr). \tag{7.28}$$

The potential flow model features a simple linearized free surface boundary condition and all of its modeling parameters (related to the integration and inversion method) are fixed. This intermediate model is used to generate the phase shifted basis of equation 7.26 which is used in the physics-based learning model 7.18.

The experimental data η_H is taken from DTMB sources for 5 Froude numbers ($Fr = 0.25, 0.27, 0.305, 0.35, 0.45$ and 20 section positions from $x/L = -1, \ldots, 1$). The linear potential flow model is used to generate the full corresponding intermediate data set η_I. With this data in hand the learning is essentially a post processing step, and a number of cases are tested. In each test, the loss is taken with respect to the



(a) Generic SVM

(b) PBLM

Figure 7-9: Residual loss of the generic SVM compared to the physics-based learning model. The PBLM loss is controled in the absence of data by the internal physical model.

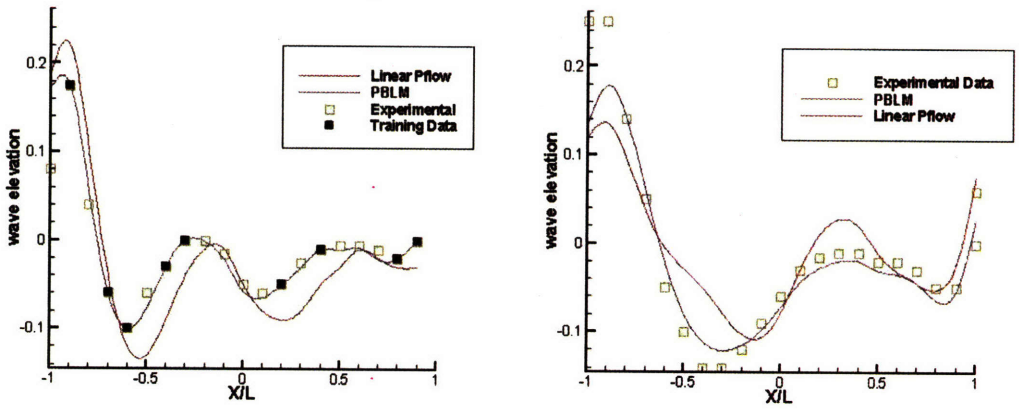
high fidelity data,

$$L = Y - Y_H \tag{7.29}$$

and the trained PBLM loss L_{PBLM} is compared to the error of the intermediate model alone L_I and a trained generic SVM mapping with no physics-based basis functions, L_{SVM}.

In the first test, a training set of 11 points is taken i.i.d. from the complete database of 100 points. The PBLM and generic SVM are trained off this data and the losses are calculated for every point in the database. The results from two instances of this test are shown in figure 7-9 where the black dots are the given data points and the contours denote the residual (equal to the loss scaled by the maximum wave elevation). A typical result for the generic SVM is shown in the first figure. The residuals are very low in regions with example data but because the data is limited and there is no internal physics-based model the errors range from 40% to -20%. The second figure shows the residuals using the PBLM, and these results are also typical. The error is still smallest where training data is available, but the intermediate model has limited the loss to less than 7% in data sparse regions.

It is somewhat artificial to randomly select the Froude number and station points for the training set. To increase the applicability of the results 15 data points are selected i.i.d. from $Fr = 0.305, 0.45$ and the PBLM is forced to generalize to the



(a) Fr=0.305

(b) Fr=0.35

Figure 7-10: Waterlines for the Wigley hull for $Fr = 0.305, 0.35$. The green box denote experimental measurements and the filled boxes are training points given to the PBLM. The red lines are the intermediate solutions, and the blue lines are the PBLM model solutions. Even on the 'unseen' Froude number the PBLM results are superior.

unseen Froude numbers $Fr = 0.25, 0.35$. A set of predicted wave elevations are shown in figure 7-10. The green boxes in that figure are the experiential points, the filled boxes are in the training set, the red lines are the intermediate model results and the blue lines are the PBLM results. As expected, the data helps the PBLM to reduce the empirical error of the intermediate model, as shown in the first figure. This result also carries over the to unseen Froude number $Fr = 0.35$ which is intermediate to the training data. The $Fr = 0.25$ case is outside the training sets and the PBLM results are virtually identical to the internal model results. This means that the data is not enabling greater predictive accuracy in that region but it also means the capacity control is effectively limiting the generalization error of the PBLM.

7.3.5 PBLMs using an Intermediate 2D+T Model

Chapter 5 demonstrated that 2D+T cannot model the flow of a 3D ship with quantitative accuracy unless that ship is slender $L/B > 16$ and moving at high speeds $F_{2D} > 0.2$. However, the model does give good qualitative predictions for 3D flows at lower lower speeds and slenderness values. Additionally, 2D+T does capture the non-

linear boundary conditions of the hull with a simulation many orders of magnitude faster than the 3D simulations.

As such the 2D+T model makes a fine candidate for the physics-based intermediate model of a PBLM. Using the same PBBF architecture as in the previous section, a SVM learning machine is trained to correct the 2D+T output based on 3D examples over a range of speeds and slenderness values. In the terminology above, the high-fidelity model is the 3D CFD

$$Y_H = \{z_{contact}, z_{crest}\} = f_{3D}(x/L, F_{2D}, \tan\alpha) \tag{7.30}$$

where $\tan\alpha$ is the entrance angle, inversely proportional to L/B. The intermediate model is 2D+T

$$Y_I = \{z_{contact}, z_{crest}\} = f_{2D+T}(x/L, F_{2D}) \tag{7.31}$$

and this allows for the generation of basis functions with a rich physical background based on nonlinear boundary conditions. Additionally, the 2D+T solution becomes the exact solution in the limit of $\tan\alpha \to 0$. This intermediate model is used to generate the phase shifted basis of equation 7.26 which is used in the physics-based learning model 7.18.

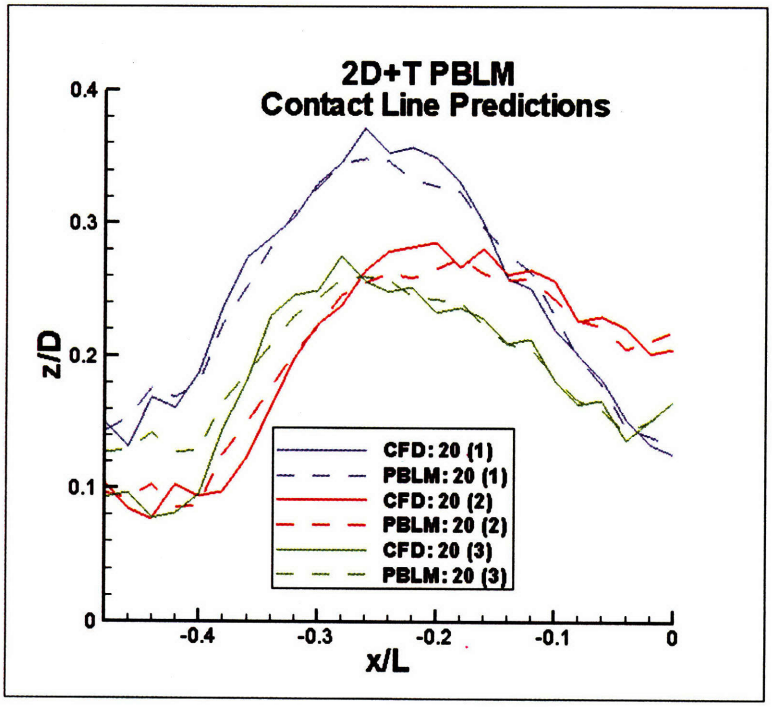
The 2D+T PBLM maps Z_I to Y_H, and is therefore able to minimized the training error of the 3D breaking wave with a lower capacity than a comparative direct curve-fitting prediction tool. Additionally, the physics-based models are all Cartesian-grid methods with only the single ϵ model parameter from the BDIM. Therefore, the complete model is more optimal than any individual component or any other prediction tool for breaking ship bow waves we are aware of.

This PBLM is trained using the data presented in the previous chapter and the results are shown in figure 7-11. For this test the PBLM is trained using the 2D+T model and 3D results for the hull with $L/B = 8.3, 33.2$. To teach the PBLM that 2D+T is a limiting solution for the 3D flow the 2D+T is labeled as $\tan\alpha = 0$ and given as an additional training solution. As such

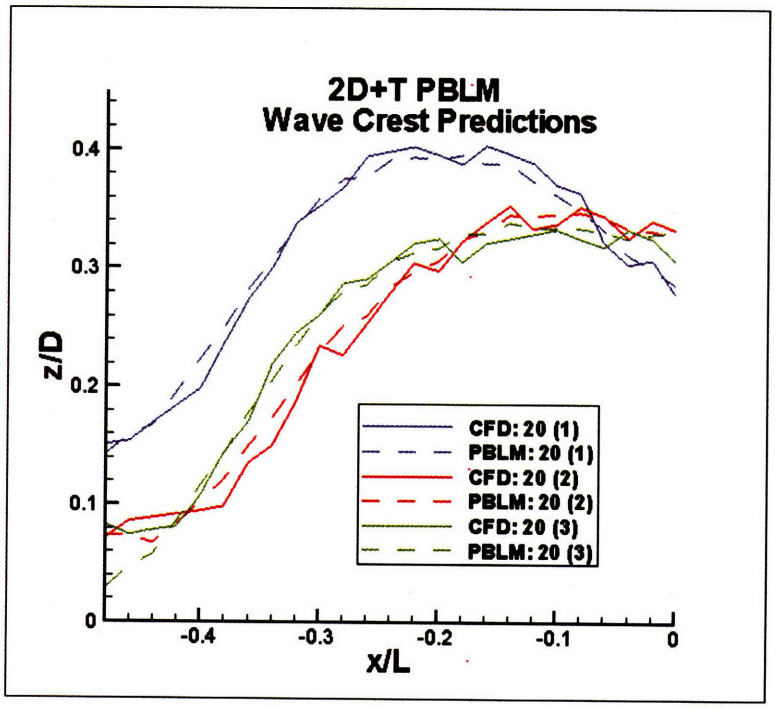
The results show that the trained PBLM makes excellent quantitative predictions

for $L/B \geq 8.3$. Such a model is therefore likely to be extremely useful in design and analysis, predicting the wave characteristics for any value of slenderness using a few 3D examples and the 2D+T solution.

In conclusion, this chapter has developed and tested a new architecture for physics and learning based models; the physics-based basis function architecture. The method is a generalization and extension of previous methods such a semi-empirical models and semi-parametric models which allows physics-based solution methods to be combined with learning methods to produce highly optimal prediction models. Unlike many PBLMs in the literature, the PBBF architecture is simple and self-contained, allowing for rapid application to new engineering problems. The models are fast and capacity controlled and show excellent predictive performance for sample ocean engineering systems.



(a) Contact Line



(b) Wave Crest

Figure 7-11: PBLM learning model for the contact line and wave crest elevation for $F_{2D} = 0.191$. The PBLM is trained on only the 3D data from the $L/B = 8.3, 33.2$ case and $L/B = 16.6$ being test case. The PBLM makes excellent quantitative predictions for these breaking ship bow wave measurements.

Chapter 8

Concluding Remarks

This thesis presented three key contributions to the study of computational modeling for complex physical systems. The theory of modeling presented in chapter 6 and expanded in chapter 7 illustrated the use of universal metrics and model units and architectures. These enabled preliminary arguments be be developed supporting the research on Cartesian-grid methods based on their reduced capacity, risk and cost. The conceptualization of a physics-based learning model is grounded by developing bounds on the metrics of compound models based on their component units and model architectures. This allowed previous examples of PBLM to be analyzed and a new effective architecture, the physics-based basis function to be developed. This architecture is used to produce optimal predictions tools for ship waves, both in terms of high-accuracy and low cost.

Technical advancements to the current state-of-the-art boundary immersion approaches constitutes the second key contribution. Chapter 2 presented a new conservative Volume-of-Fluid transport method for the simulation of 2D and 3D free surface flows. The standard VOF transport tests are shown to be inadequate because of their lack of stretching. Tests with stretching, such as a 2D corner and a 3D breaking wave, revealed conservation errors in the previous methods. The strictly conservative new method avoids such errors and is at least as simple as the methods currently used in the literature.

Another set of technical advancements are presented in chapters 3 and 4 for the

problem of computing flows with general solid bodies on a Cartesian-grid. Previous methods suffered from a lack of generality or computational simplicity. A new Boundary Data Immersion Method is introduced based on the combination of multiple governing equations into a single meta equations, valid throughout the system domain. These meta-equations offer the generality to model any type of body/fluid system; examples treated include solids in 2D potential flow, moving bodies in 2D Euler flow, ships and moving bodies in 3D flow, as well as numerical exits. The formulation and results are also shown to match standard fitted-grid solutions when applied to that subset of problems. Indeed, the Boundary Data Immersion method appears to constitute a completely general method for the solution of partial differential equations on numerical grids of any type.

Simple implementations are proposed for the resulting meta-equations based on a finite-width transition from one set of governing equations to the next. As such, the method is not a 'sharp-interface' method, but side steps all the numerical order of accuracy and stability problems common to such methods. Indeed, the width ϵ is a transparent model parameter, easily optimized and fixed for complete sets of simulations, greatly reducing the model capacity and user interface costs. By utilizing a body equation which allows for slip, the effects of the smoothing width may be further reduced. Methods for determining the pressure force, including a simple kernel method, are proposed and validated. The convergence of the smooth solution to available analytic benchmarks is found to be second order in the smoothing width, and 'internal flows' are found to have an negligible effect on the external solution.

The third key thesis contribution is the application of this general solution tool to the validation of 2D+T models in chapter 5. Simulations of modern 2D+T wavemaker experiments are found to have excellent agreement when fully resolved. Using a free-slip boundary condition allowed lower resolution runs to achieve similar accuracy. The geometries from different wavemaker runs and the ideal 5415 geometry are superimposed to generate a test hull for the 2D+T validation. The small changes in the wavemaker geometry are found to result in correspondingly small changes in the resulting wave hydrodynamics. A wide range of equivalent Froude number 2D+T

simulations are run and matched well to the experimental results. A set of 3D runs are performed on the hull geometry for the same speeds and a range of slenderness values. Because the same meta-equation-based Cartesian-grid method is used for both simulations direct quantified comparisons are easily made. The results showed that 2D+T accurately predicts the bulk flow for high speeds and fine entrance angles. The 2D+T model also predicts the correct class of breaking. However, the onset of breaking is predicted too early by 2D+T due to local violation of the 2D+T assumptions in the true 3D breaker. However, the BDIM 2D+T model is still a fast and relatively accurate tool for breaking wave prediction and a PBLM is developed in chapter 7 used that model and 3D data to optimally predict the breaking wave characteristics over a range of Froude numbers and slenderness values.

There are still many interesting questions on all fronts of this research. From the technical standpoint, the reliance on Cartesian background grids greatly limits the achievable resolution of the method. While adaptive-grid methods often have an array of numerical complications, the volume fraction f and interpolation function δ_ϵ could be used to locally refine the grid in the region where the equations transition from one medium to another. Combining a recursive oct-tree adaptive-grid method with a multi-grid solver could reduce the computational effort by an order of magnitude, limiting the number of computational points and inverting for the pressure much faster.

The investigation of 2D+T methods has not been fully completed either. While the thesis demonstrated that small variations to a hull has small changes in the 2D+T flow, large changes such as the addition of a bulbous bow are not considered. The question of application of 2D+T to multi-hull vessels is also interesting, and the flexible Cartesian-grid framework described above enables further research into this topic.

Finally, the model theory and optimized physics-based learning models presented in this work are still in their infancy. There are an almost daunting number of research questions and possibilities yet to be investigated. One example that seems promising is an extension of the physics-based basis function architecture to the internal variable

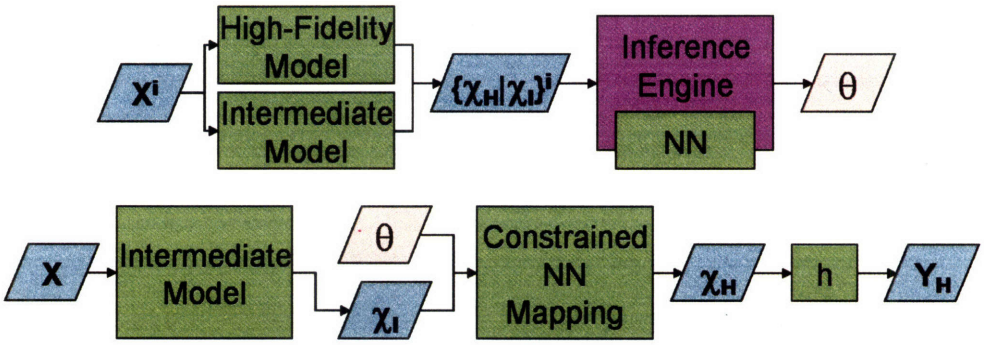


Figure 8-1: A more advanced physics-based basis function model, utilizing the the internal variables χ of the intermediate and high-fidelity model.

level.

As presented in the model theory, physics-based models typically operate on a set of internal variables χ, the variational generalized coordinates of the system. The conservation laws are used by a physics-based model to solve for these internal variables, the variables are measured by the operator h for the values of the output set Y. Therefore, a feature mapping based on these more fundamental variables could be constrained to obey some simplified version of these conservation laws, and this process is shown in figure 8-1.

Inclusion of general conservation laws into the mapping would greatly reduce the require capacity of the system, and likely increase the predictive accuracy. However, this would come at the cost of a much more specialized learning machine, such as some form of a constrained NN. This idea could lead toward a physics-based learning model capable of adaptable levels of accuracy, or possibly be a complete dead end. Further work in the field is required to research this possibility among the countless others.

Appendix A

Proof of VOF conservation

This appendix details the proof of fluid volume conservation for the Volume-of-Fluid (VOF) transport equation presented in section 2.2.2. That section presented and discussed three basic requirements for a conservative scheme:

1. Conservative treatment of the volume flux
2. Zero sum divergence term
3. Enforcement of the volume fraction constraint $0 \leq f \leq 1$ at every point in the algorithm

The section showed that the first two requirements are easily met but demonstrating the third is non-trivial, particularly in three spacial dimensions.

Recall that the general transport algorithm is

$$(f - f_0)\frac{\Delta\Omega}{\Delta t} = \sum_{d=1}^{\mathcal{N}} \left(\Delta_d F_d + g\frac{\partial u_d}{\partial x_d} \right) \tag{A.1}$$

where f is the volume fraction, $\Delta\Omega$ is the cell volume, Δt is the time step, $\mathcal{N}$ is the number of spacial dimensions, F is the flux of dark fluid , $\Delta_d F_d$ is the net flux in the d direction, and u is the velocity. Equation A.1 has conservative flux treatment and a zero sum divergence term. The factor g in that equation must be set such that

$$0 \leq f \leq 1 \tag{A.2}$$

at all times. In chapter 2 it shown that setting $g = 0$ results in a transport equation which cannot conserve fluid volume in a flow with stretching. On the other hand, that chapter claims that by setting $g = \bar{f}_c$, the color function at cell center, the method conserves volume exactly. This appendix proves the volume fraction constraint is met at all times by bounding the flux F in terms of the velocity u and local volume fraction f. Once bounds are established which ensure that no volume can overfill or 'over-empty' the conservation of volume is guaranteed.

Constructing general bounds on the flux of dark fluid F is difficult because of the varieties of orientations which the fluid interface may take as well as the varieties of velocity fields the volume fraction may be transported in. To simplify the initial analysis we first consider only the flux through the right face of the cell when transported by a positive velocity. In this case a general lower bound on the flux of dark fluid is

$$\frac{\Delta t}{\Delta x} F \geq \max\left(0, \frac{\Delta t}{\Delta x} u - \hat{f}\right). \tag{A.3}$$

where $\hat{f} \equiv 1 - f$ is the volume of light fluid in the cell and, as in section 2.2.2, the velocities and fluxes are scaled by $\frac{\Delta t}{\Delta x}$ making them local Courant numbers. For the rest of the proof this scaling is assumed, and the lower bound is written as

$$F \geq \max\left(0, u - \hat{f}\right). \tag{A.4}$$

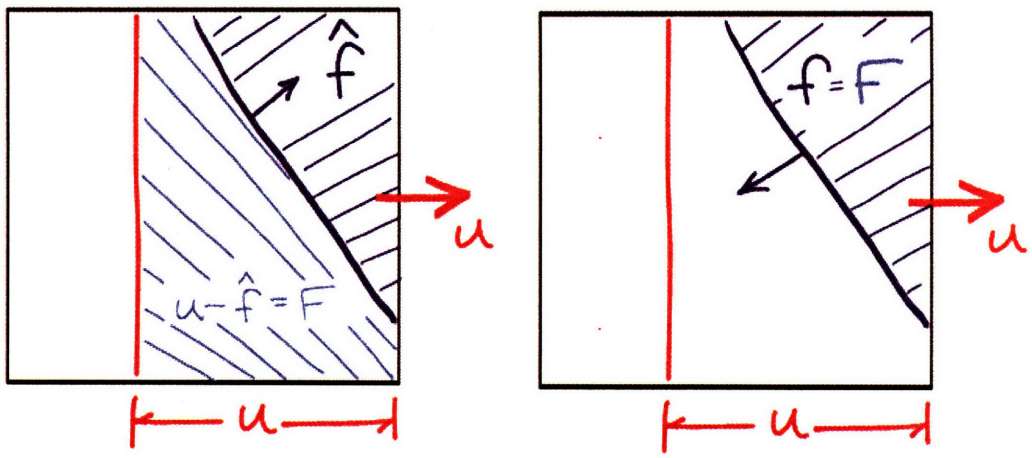
Figure A-1(a) illustrates the lower bound, which states that if $u - \hat{f} > 0$ then at least that volume of dark fluid is fluxed. For instance, if $u = 0.4$ and $f = 0.9$ then $\hat{h} = 0.1$ and $F \geq 0.3$ regardless of the interface orientation.

A general upper bound is

$$F \leq \min\left(u, f\right) \tag{A.5}$$

which is illustrated in figure A-1(b) and states that the flux must not exceed the lesser of f and u. For instance, if $f = 0.1$ and $u = 0.4$ then $F \leq 0.1$.

Equations A.4 and A.5 are valid for any interface orientation and positive velocity on the right face. The next task is to extend these bounds to more general velocity



(a) Lower Bound (b) Upper Bound

Figure A-1: Illustrations of the lower and upper bound on the flux. Note that these figures consider only positive velocity components on the right face.

conditions.

The first case is simple; when the left face velocity is negative the same bounds apply with negatives multiplying the velocities. As shown in figure A-2 the dark volume is always fluxed 'downwind'. In the case of a negative right velocity u_r or positive left velocity u_l this implies that the flux depends on the volume fraction of the neighboring cells. However, a time constraint on the VOF transport equation must not reference the left or right cell volume fractions. A set of bounds which do not reference the volume fraction are

$$0 \leq |F| \leq |u| \tag{A.6}$$

but these bounds are not as tight as bounds A.4 and A.5. In order to prove conservation the bounds must be as tight as possible in every velocity field. There are four general cases; case (a) $u_r, u_l > 0$, case (b) $u_r > 0, u_l < 0$, case (c) $u_r < 0, u_l > 0$, and case (d) $u_r, u_l < 0$. The bounds for the net flux $\Delta F \equiv F_r + F_l$ in each case are derived by combining equations A.4,A.5 and A.6 as appropriate, giving

$$\text{case (a)} \quad -\min(u_r, f) \leq \Delta F \leq u_l - \max(0, u_r - \hat{f}) \tag{A.7}$$

$$\text{case (b)} \quad -\min(u_r - u_l, f) \leq \Delta F \leq -\max(0, u_r - u_l - \hat{f}) \tag{A.8}$$

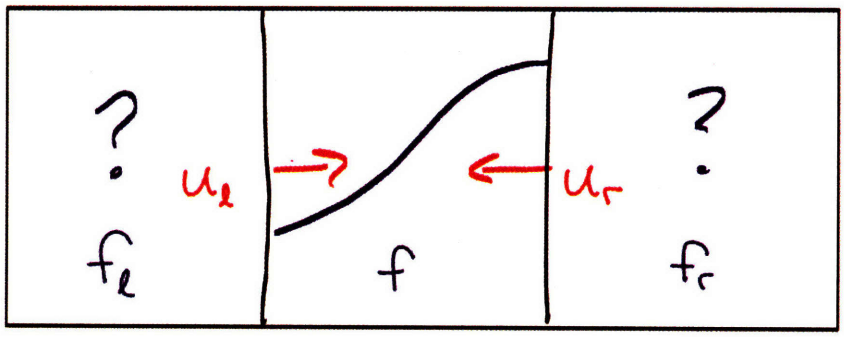


Figure A-2: Illustration of the downwind flow of volume fraction. A looser set of local bounds are needed in this case.

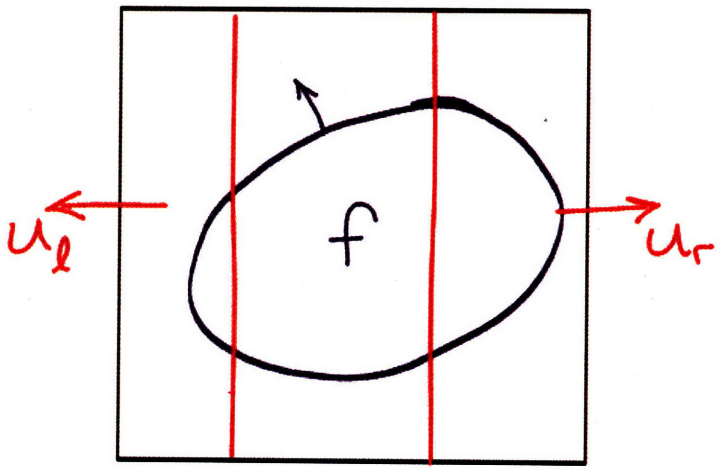


Figure A-3: Illustration of case (b) where the velocity and flux are entirely out of the cell. The bound $-f \leq \Delta F \leq 0$ is self evident.

$$\text{case (c)} \qquad 0 \leq \Delta F \leq u_l - u_r \tag{A.9}$$

where case (d) is the same as case (a) swapping u_l and u_r. As the cases are equivalent we do not consider case (d) separately. Also note that applying equation A.4 and A.5 blindly to case (b) gives

$$-\min(u_r, f) - \min(-u_l, f) \leq \Delta F \leq -\max(0, u_r - \hat{f}) - \max(0, u_l - \hat{f}) \tag{A.10}$$

but this bound is not tight. Combining it with $-f \leq \Delta F \leq 0$ (shown in figure A-3) results in the bound given above.

As stated above these bounds are used to test equation A.1 against constraint A.2. In other words, we determine if

$$f_o + \Delta F_{min} + g(u_r - u_l) \geq 0 \tag{A.11}$$

$$f_o + \Delta F_{max} + g(u_r - u_l) \leq 1 \tag{A.12}$$

for all u, f and some choice of g. First we choose $g = 0$ which is known to be deficient. Plugging in the bounds we gives

$$\text{case (a)} \quad f_0 \geq \min(u_r, f_0) \tag{A.13}$$

$$\text{case (a)} \quad \hat{f}_0 \geq u_l - \max(0, u_r - \hat{f}_0) \tag{A.14}$$

$$\text{case (b)} \quad f_0 \geq \min(u_r - u_l, f_0) \tag{A.15}$$

$$\text{case (b)} \quad \hat{f}_0 \geq -\max(0, u_r - u_l - \hat{f}) \tag{A.16}$$

$$\text{case (c)} \quad f_0 \geq 0 \tag{A.17}$$

$$\text{case (c)} \quad \hat{f}_0 \geq u_l - u_r. \tag{A.18}$$

For example, the first bound states that if $f_0 \geq \min(u_r, f_0)$ then case (a) cannot over-empty the cell. This bound is trivially ensured and so setting $g = 0$ never results in over-emptying due to case (a). However, the second and last equality cannot be ensured because u_r can be either large or smaller than u_l. For example, in case (a) if $\hat{f}_0 = 0.2, u_r = 0.1$ and $u_l = 0.3$ then the cell overfills. A time step restriction does not help enforce the bounds because $\hat{f}$ may equal zero. Therefore we have shown formally that $g = 0$ allows overfilling and does not generally conserve fluid volume.

Next, setting $g = 1$ to activate the dilatation term gives

$$\text{case (a)} \quad f_0 \geq \min(u_r, f_0) + u_l - u_r \tag{A.19}$$

$$\text{case (a)} \quad \hat{f}_0 \geq u_r - \max(0, u_r - \hat{f}_0) \tag{A.20}$$

$$\text{case (b)} \quad f_0 \geq \min(u_r - u_l, f_0) + u_l - u_r \tag{A.21}$$

$$\text{case (b)} \quad \hat{f}_0 \geq -\max(0, u_r - u_l - \hat{f}) + u_r - u_l \tag{A.22}$$

$$\text{case (c)} \qquad f_0 \geq u_l - u_r \tag{A.23}$$

$$\text{case (c)} \qquad \hat{f}_0 \geq 0. \tag{A.24}$$

Now the second and last inequalities are met for any general flow but the first and fifth are not. Therefore setting $g = 1$ does not allow the volume to overfill but does allow it to over-empty. Note that setting $g = f_0$, does not help help. Indeed, neither of the case (a) bounds

$$\text{case (a)} \qquad f_0 \geq \min(u_r, f_0) - f_0(u_l - u_r) \tag{A.25}$$

$$\text{case (a)} \qquad \hat{f}_0 \geq u_r - \max(0, u_r - \hat{f}_0) - \hat{f}_0(u_r - u_l) \tag{A.26}$$

can be guaranteed in a general flow.

As discussed in section 2.2.2 the term $g(u_r - u_l)$ comes from a quadrature of the dilatation term

$$\int_\Omega \bar{f} \frac{\partial u_d}{\partial x_d} \, \mathrm{d}v \tag{A.27}$$

and as such the natural choice for g is the cell centered value of the color-function $\bar{f}$. In this case

$$g = \begin{cases} 1 & \text{if} \quad f_0 > 1/2 \\ 0 & \text{else} \end{cases} \tag{A.28}$$

To use this g we split the cases where f_0 is less than $1/2$ (f^-) from those where f_0 is greater than $1/2(f^+)$. Clearly, the bounds for f^- are the same as when $g = 0$ and the bounds for f^+ are the same as when $g = 1$. However, there are now bounds on the values of f_0 and $\hat{f}_0$. Examining the prior violations give

$$\text{case (a)} \qquad f_0^+ \geq \frac{1}{2} \geq \min(u_r, f_0) + u_l - u_r \tag{A.29}$$

$$\text{case (a)} \qquad \hat{f}_0^- \geq \frac{1}{2} \geq u_l - \max(0, u_r - \hat{f}_0) \tag{A.30}$$

$$\text{case (c)} \qquad f_0^+ \geq \frac{1}{2} \geq u_l - u_r \tag{A.31}$$

$$\text{case (c)} \qquad \hat{f}_0^- \geq \frac{1}{2} \geq u_l - u_r. \tag{A.32}$$

which are all achievable through a time step restriction.

Effectively, the use of $g = \bar{f}_c$ activates the dilation term only when there is a sufficient amount of dark fluid in the cell to assure no over-emptying. This result is easily extended to a second advection sweep by further restriction of the time step. A time step restriction of the form

$$|u|, |\Delta u| \leq \frac{\Delta x}{\Delta t 2(\mathcal{N}-1)} \tag{A.33}$$

guarantees the bounds above are met for $\mathcal{N}-1$ advection sweeps. The divergence-free velocity field ensures the final sweep will 'undo' the stretching of the earlier steps.

Therefore, equation A.1 with g defined by A.28 and time step bounded by A.33 ensures constraint A.2 at all times and thus exact conservation of volume.

Bibliography

[1] Yann Andrillon and Bertrand Alessandrini. A 2d+t vof fully coupled formulation for the calculation of breaking free-surface flow. *J Mar Sci Technol*, 8:159168, 2004.

[2] Eugenio Aulisa, Sandro Manservisi, Ruben Scardovelli, and Stephane Zaleski. A geometrical area-preserving volume-of-fluid advection method. *J. Comp. Phys.*, 192:355–364, 2003.

[3] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38, 1977.

[5] Douglas G. Dommermuth, G.E. Innis, T. Luth, E. Novikov, E. Schlageter, and J.C. Talcott. Numerical simulation of bow waves. In *22nd Symposium on Naval Hydrodynamics*, 1998.

[6] Douglas G. Dommermuth, Thomas T. OShea, Donald C. Wyatt, Mark Sussman, Gabriel D. Weymouth, Dick K.P. Yue, Paul Adams, and Randall Hand. The numerical simulation of ship waves using cartesian-grid and volume-of-fluid methods. In *26th Symposium on Naval Hydrodynamics*, 2006.

[7] Douglas G. Dommermuth, Mark Sussman, Robert F. Beck, Thomas T. O'Shea, Donald C. Wyatt, Kevin Olson4, and Peter MacNeice. The numerical simulation of ship waves using cartesian grid methods with adaptive mesh refinement. In *25th Symposium on Naval Hydrodynamics*, 2004.

[8] Douglas G. Dommermuth, Dick K. P. Yue, W. M. Lin, R. J. Rapp, E. S. Chan, and W. K. Melville. Deep-water plunging breakers: a comparison between potential theory and experiments. *Journal of Fluid Mechanics*, 189:423–442, 1988.

[9] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:150, 1999.

[10] Wil Faller and David Hess. Simulation based design: A real-time approach using recursive nueral networks. *AIAA*, 0911, 2005.

[11] Wil Faller and David Hess. Real-time simulation based design part ii: Changes in hull geometry. *AIAA*, 1485, 2006.

[12] J.H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics.* Springer, third edition, 2002.

[13] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons Ltd., second edition, 1987.

[14] E. Fontaine and R. Cointe. A slender body aproach to nonlinear bow waves. *Phil. Trans. Royal Society of London*, A 355:565–574, 1997.

[15] E. Fontaine, O. M. Faltinsen, and R. Coite. New insight into the genertion of ship bow waves. *J. Fluid Mech.*, 421:1538, 2000.

[16] W. Froude. *The Papers of William Froude.* London: Instritution of Naval Architects, 1955.

[17] Anvar Gilmanov and Fotis Sotiropoulos. A hybrid cartesian/immersed boundary method for simulating flows with 3d, geometrically complex, moving bodies. *J. Comp. Phys.*, 207:457–492, 2005.

[18] Dan Givoli, Beny Neta, and Igor Patlashenko. Finite element analysis of time-dependent semi-infinite wave-guides with high-order boundary treatment. *Int. J. for Numerical Methods in Engineering*, 58:1955–1983, 2003.

[19] C. Harpham, C.W. Dawson, and M.R. Brown. A review of genetic algorithms applied to training radial basis function networks. *Neural Computing and Applications*, 13:193201, 2004.

[20] Dalton J. E. Harvie and David F. Fletcher. A new volume of fluid advection algorithm: the defined donating region scheme. *Int. J. Numer. Meth. Fluids*, 35:151172, 2001.

[21] Kelli Hendrickson and Dick K.-P. Yue. Navier-Stokes simulations of unsteady small-scale breaking waves at a coupled air-water interface. In *26th Symposium on Naval Hydrodynamics*, 2006.

[22] David Hess, Wil Faller, J. Lee, Thomas Fu, and Edward Ammeen. Ship maneuvering simulation in wind and waves: Nonlinear time-domain approach using recursive neural networks. In *26th Symposium on Naval Hydrodynamics*, 2006.

[23] R.L. Higdon. Radiation boundary conditions for disperive waves. *SIAM J. Numerical Analysis*, 31:24–462, 1994.

[24] C. W. Hirt. Addition of wave transmitting boundary conditions to the flow-3d program. *FLOW-3D Documentation*, FSI-99-TN49, 1999.

[25] C.W. Hirt and B.D. Nicholas. Volume of fluid (vof) method for the dynamics of free boundaries. *J. Comput. Phys.*, 39:201–225, 1981.

[26] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffry E. Hinton. Adaptive mixture of local experts. *Neural Computation*, 3:79–87, 1991.

[27] A. Karion, T. Sur, T. C. Fu, D. Furey, J. R. Rice, and D. D. Walker. Experimental study of the bow wave of a large towed wedge. In *8th International Conference Numerical Ship Hydrodynamics*, 2003.

[28] B.V. Korvin-Kroukovsky and W. R. Jacobs. Pitching and heaving motions of a ship in regular waves. *Transactions SNAME*, 65:590632, 1957.

[29] Ludwik Kostro. Albert einstein's hypothetism. *Science and Education*, 7:317–322, 1998.

[30] Cornelilus Lanczos. *The Variational Principles of Mechanics.* Dover, 1970.

[31] B. P. Leonard and Simin Mokhtari. Beyond first-order upwinding: The ultra-sharp alternative for non-oscillatory steady-state simulation of convection. *International Journal for Numerical Methods in Engineering*, 30:729–766, 1990.

[32] W.M. Lin, J.N. Nueman, and D.K. Yue. Nonliear forced motion of floating bodies. *Proc. 15th Int. Symp. on Naval Hydrodynamics*, 1984.

[33] Woei-Min Lin. *Nonliear Motion of the Free Surface Near a Moving Body.* PhD dissertation, Department of Ocean Engineering, Massachusetts Institute of Technology, 2005.

[34] S. Marella, S. Krishnan, H. Liu, and H.S. Udaykumar. Sharp interface cartesian grid method I: An easily implemented technique for 3d moving boundary computations. *J. Comp. Phys.*, 210,1:1–31, 2005.

[35] Al Martinich and E David Sosa, editors. *A Companion to Analytic Philosophy.* Blackwell, 2001.

[36] J.H. Michell. The wave resistance of a ship. *Phil. Mag.*, 45:106–123, 1898.

[37] H. Mukundan. *Vortex-Induced Vibration of Marine Risers: Motion and Force Reconstruction from Field and Experimental Data.* PhD thesis, Massachusetts Institute of Technology, April 2008.

[38] Max M. Munk. The gas flow past slender bodies. *Journal of Applied Physics*, 24:584–589, 1953.

[39] S. Muzaferija and M. Peric. Comuputation of free-surface flows using the finite-volume method and moving grids. *Numerical heat transfer*, 32, PartB:369–384, 1997.

[40] F. Noblesse, D. Delhommeau, M. Guilbaud, D. Hendrix, and G. Karafiath. Analytic and experimental study of unsteady and overturning ship bow waves. In *26th Symposium on Naval Hydrodynamics*, pages 57–72, 2006.

[41] I. Orlanski. A simple boundary condition for unbounded hyperbolic flows. *J. Comp. Phys.*, 21:251, 1976.

[42] P. Persson and J. Peraire. Sub-cell shock capturing for discontinuous galerkin methods. *AIAA Aerospace Sciences Meeting and Exhibit*, 44, 2006.

[43] Charles S. Peskin. The immersed boundary method. *Acta Numerica*, pages 1–39, 2002.

[44] Les A. Piegl and Wayne Tiller. *The Nurbs Book.* Springer, 1997.

[45] J.E. Pilliod and E.G. Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *J. Comput. Phys.*, 199:465–502, 2004.

[46] E.G. Puckett, A.S. Almgren, J.B. Bell, D.L. Marcus, and J.D. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *J. Comput. Phys.*, 130(2):269–282, 1997.

[47] William J. Rider and Douglas B. Kothe. Reconstructing volume tracking. *J. Comp. Phys.*, 141:112–152, 1998.

[48] PJ Roache. *Verification and validation in computational science and engineering.* Hermosa Publishers, 1998.

[49] Damian W.I. Rouson and Yi Xiong. Design metrics in quantum turbulence simulations: How physics influences software architecture. *Scientific Programming*, 12, 2004.

[50] R. Scardovelli and S. Zaleski. Analytical relations connecting linear interfaces and volume fractions in rectangular grids. *J. Comp. Phys.*, 164:228–237, 2000.

[51] R. Scardovelli and S. Zaleski. Interface reconstruction with least-square fit and split eulerianlagrangian advection. *Int. J. Numer. Meth. Fluids*, 41:251274, 2003.

[52] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond.* MIT Press, 2001.

[53] M. Shakeri. *An experimental 2D+T investigation of breaking bow waves.* PhD dissertation, Department of Mechanical Engineering, University of Maryland, 2005.

[54] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–424,623–656, July,October 1948.

[55] J.R. Shearer and J.J. Cross. The experimental determination of the components of ship resistance for a mathematical model. 1965.

[56] Mark Sussman. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comp. Phys.*, 187:110–136, 2003.

[57] W. M. Thorburn. The myth of occam's razor. *Mind*, 27:345–353, 1918.

[58] Y.H. Tseng and J.H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *J. Comp. Phys.*, 192:593623, 2003.

[59] M.P. Tulin. The theory of slender surfaces planing at high speed. *ShiKstechnik*, 4:125–133, 1956.

[60] M.P. Tulin and M. Wu. Divergent bow waves. *Symp. on Naval Hydrodynamics*, 21:99–117, 1996.

[61] H. S. Udaykumar, Heung-Chang Kan, Wei Shyy, and Roger Tran-Son-Tay. Multiphase dynamics in arbitrary geometries on fixed cartesian grids. *J. Comp. Phys.*, 137:366–405, 1997.

[62] H. S. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna. A sharp interface cartesian grid method for simulating flows with complex moving boundaries. *J. Comp. Phys.*, 174:345–380, 2001.

[63] Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[64] Gabriel D. Weymouth, Douglas G. Dommermuth, Kelli Hendrickson, and Dick K.-P. Yue. Advancements in cartesian-grid methods for computational ship hydrodynamics. In *26th Symposium on Naval Hydrodynamics*, 2006.

[65] G.D. Weymouth, K. Hendrickson, T. OShea, D.G Dommermuth, D.K.P. Yue, P. Adams, and R. Hand. Modeling breaking ship waves for design and analysis of naval vessels. In *HPCMP Users Group Conference*, 2006.

[66] Jianming Yang and Elias Balaras. An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving bodies. *J. Comp. Phys.*, 215:12–40, 2006.

[67] T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy. An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries. *J. Comp. Phys.*, 156:209–240, 1999.

[68] Luoding Zhu and Charles S. Peskin. Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method. *J. Comp. Phys.*, 179,2:452–468, 2002.