Improved Underwater Vehicle Control and Maneuvering Analysis with Computational Fluid Dynamics Simulations

Ryan G. Coe

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Aerospace Engineering

Wayne L. Neu, Chairman
Daniel J. Stilwell
Craig A. Woolsey
Danesh K. Tafti

August 8, 2013
Blacksburg, Virginia

Keywords: CFD, AUV, Maneuvering

Improved Underwater Vehicle Control and Maneuvering Analysis with Computational Fluid Dynamics Simulations

Ryan G. Coe

ABSTRACT

The quasi-steady state-space models generally used to simulate the dynamics of underwater vehicles perform well in most steady flow scenarios, and are therefore acceptable for modeling today's fleet of endurance-focused autonomous underwater vehicles (AUVs). However, with their usage of numerous assumptions and simplifications, these models are not well suited to certain unsteady flow situations and for use in the development of AUVs capable of performing more extreme maneuvers. In the interest of better serving efforts to design a new generation of more maneuverable AUVs, a tool for simulating vehicle maneuvering within computational fluid dynamics (CFD) based environments has been developed. Unsteady Reynolds-averaged Navier-Stokes (URANS) simulations are used in conjunction with a 6-degree-of-freedom (6-DoF) rigid-body kinematic model to provide a *numerical test basin* for vehicle maneuvering simulations. The accuracy of this approach is characterized through comparison with experimental measurements and quasi-steady state-space models. Three state-space models are considered: one model obtained from semi-empirical database regression (this is the method most commonly used in application) and two models populated with coefficients determined from the results of prescribed motion CFD simulations. CFD analyses focused on supporting the design of the GPAUV are also presented.

for Brittney

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The ongoing development of a series of autonomous underwater vehicles (AUVs) by a group of researchers at Virginia Tech has sparked interest in hydrodynamic modeling and control design for underwater vehicles in general. Early efforts to characterize the performance of potential AUV and control algorithm designs using standard quasi-steady state-space models inspired ideas about how to improve those models and use CFD-based maneuvering simulations when necessary to achieve higher fidelity results.

The quasi-steady state-space models traditionally used in control system design make substantial assumptions to simplify the complex physical system presented by unsteady viscous fluid flow. While these assumption are well reasoned, the resulting model is not necessarily based on first principles. Thus, while state-space models are in general quite useful and sufficient for mostly steady flows, they are not well equipped to model certain viscous fluid flow regimes and the vehicle dynamics that result. Nonlinear fluid dynamic phenomena, such as transient flow separation, vortex shedding and unsteady lift effects, can result in nonlinear vehicle dynamics not easily linked to a quasi-steady state (the vehicle's instantaneous orientation, velocity and acceleration). Inaccuracies in a state-space model can pose an impediment to control algorithm testing and design. In many cases, discrepancies between a state-space model and physical reality are small enough for a sufficiently robust control system to function without difficulty. Vehicles tasked with performing fast, high-angle maneuvers are, however, not well served by these reduced-order models.

An additional limitation to the usage of state-space models in vehicle and control algorithm design is the need to populate the model with coefficients specific to the vehicle geometry, as well as its operating regime. Before the coefficients of a model can be determined, the formulation of that model (i. e., the means with which it will model control inputs, viscous damping, etc.) must be chosen so as to best capture the phenomena important to that vehicle.

There is an increasing need for an economical means of populating vehicle maneuvering models, prior to the completion and testing of a prototype, that avoids the limitations of semi-empirical databases.

1.2 Approach

To support the design of next generation fluid-based vehicles, this dissertation explores the ability of maneuvering simulations based in CFD to supply engineers with accurate information for potential vehicle designs and control algorithms. In this approach, an unsteady viscous CFD simulation and coupled rigid-body kinematic model serve as a high-fidelity numerical test basin for open and closed-loop maneuvering analysis [1]. Although some published research in this area does exist, no common name for this approach has been adopted. Thus, in reference to the free-running reduced-scale models (FRMs) often used to predict full-scale vehicle maneuvering performance, the class of tools developed and tested in this study are referred to as *virtual free-running model* (VFRM) simulations.

To better characterize the effectiveness of VFRM simulations, substantial effort has also been focused on the study and development of quasi-steady state-space models. These models are intended for end-use analysis as well as to provide a point to which the predictions of VFRM simulations can be compared. An assessment of common modeling practices and their theoretical foundations has lead to the development of a novel state-space model for AUVs. Advancements have also been made in the process of developing state-space models from the results collected in steady and unsteady CFD simulations.

1.2.1 Applied Computational Fluid Dynamics

Traditionally, much of the academic research in the field of computational fluid dynamics has focused on the advancement of CFD methods, algorithms and codes. Thanks to this work, CFD has progressed from a subject of mostly theoretical research to a tool capable of analyzing real engineering problems. This development has opened the field of applied CFD research, where CFD is used as a *tool* to investigate complex problems.

Much in the same way that tow tanks, wind tunnels and digital particle velocimetry (PIV) systems enable experimental researchers to analyze complex engineering systems, CFD can offer a means to better understand real life systems. This mode of usage does not dismiss the need for careful attention to the underlying theories and methods on which CFD simulations rely. As with any complex tool, ample consideration must be paid to its proper usage. In that interest, this study emphasizes the need for an advanced understanding of CFD methods along with numerous verification and validation analyses to substantiate the accuracy of the presented tool-set and results.

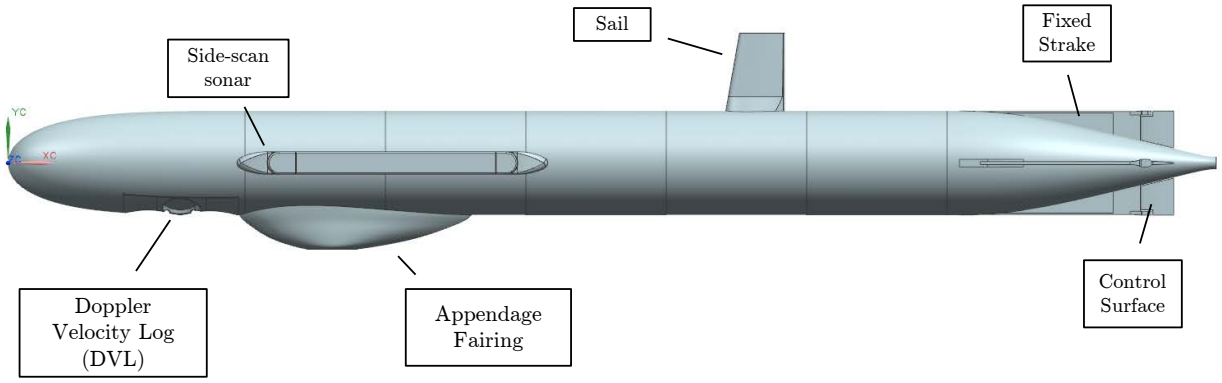


Figure 1.1: GPAUV side-view with appendage labels.

1.2.2 General Purpose Autonomous Underwater Vehicle (GPAUV)

While the applicability of the methods presented in this study spans many vehicles, discussion in this dissertation will focus almost entirely on AUVs and submarines. Specifically, the so-called *general purpose AUV* (GPAUV), shown in Figure 1.1, currently under development by a group of researchers at Virginia Tech, is considered as the vehicle of interest [2]. Major specifications of the GPAUV are given in Table 1.1.

The vehicle's hull has a length of 2.03 m and a major diameter of 0.175 m. It is designed to operate at a design speed of approximately 2 m/s (3.89 knots). This condition, which is used in the majority of analyses of this study, gives the vehicle a length-based Reynolds number, Re, of approximately 4.5×10^6, which suggests a transitional/turbulent flow regime. While the vehicle is designed in a modular fashion, allowing it to be outfitted for a wide variety of missions, the general configuration considered in this study includes a number of standard external appendages, which are shown and labeled in Figure 1.1.

As this study is concerned with the maneuvering characteristics of this vehicle, its control surfaces (shown in Figure 1.2) are of particular interest. The GPAUV is designed with relatively small control surfaces (Control Surface Flap in Figure 1.2) and fixed strakes. This configuration has been shown to supply the vehicle with the necessary control authority, while avoiding the larger control surfaces with which many AUVs are equipped (see Section 3.6.1 for a full explanation).

Table 1.1: GPAUV geometric and operational specifications.

Parameter	**Symbol**	**Value**	**Units**	**Notes**
Length	L	2.033	m	
Diameter	D	0.175	m	
Mass	m	41.2	kg	
Displacement	$\forall$	0.0414	m^3	Reflects partially flooded hull
Nominal operating speed	U	2.0	m/s	
Density of water	ρ	997.561	kg/m^3	freshwater
Viscosity of water	μ	8.89×10^{-4}	Pa-s	freshwater
Center of buoyancy	CoB	[92.5, 0, 0]	cm	measured from nose (aft, starboard, up)
Center of gravity	CoG	[92.5, 0 -0.1]	cm	measured from nose (aft, starboard, up)
Rigid-body inertia	$\mathbf{I}_{RB}$	diag(0.155, 8.71, 8.68)	kg-m^2	

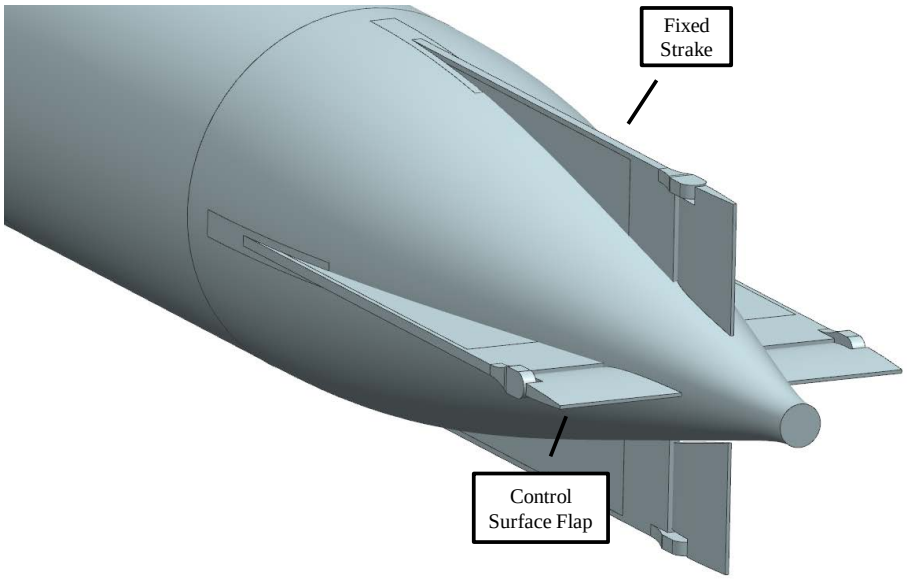


Figure 1.2: GPAUV tail geometry.

1.3 Background

1.3.1 State-Space Models

Quasi-steady state-space models (often referred to as simply state-space models or SSMs within this dissertation) represent the current standard method of analyzing vehicle maneuvering and control algorithm performance. State-space models use a vehicle's state, which typically includes its orientation, velocity and acceleration, to represent its dynamics as a set of coupled first-order differential equations. Using a numerical integration scheme, these equations can be used to predict the vehicle state for successive time-steps.

To accomplish this prediction in an efficient manner, a number of assumptions and simplifications are necessary. These assumptions, along with other limitations inherent to this style of modeling, are the central motivating factors for the research discussed in this dissertation.

1. Superposition - This assumes multiple phenomena accounted for with separate coefficients do not interact. Take, for example, a vehicle with simultaneous motions in sway and yaw: forces on the vehicle will arise from phenomena due to both of these motions, but it is unlikely that the flow structures created by each of these will act independently of the other. In many models, this is accounted for by including so-called cross terms. As with other coefficients used in state-space models, the magnitude of each cross term is unique to the body geometry and flow regime. Accounting for every effect and interaction of effects, which must involve identification of the parameters that describe the interactions, is not feasible.

2. Quasi-steady - This assumes that the vehicle's instantaneous state describes its surrounding fluid field and therefore the forces and moments experienced. In an unsteady flow, this assumption is invalid. The magnitude of the errors incurred from this assumption is proportional to vehicle acceleration and change in acceleration, but also dependent on vehicle geometry. This is well demonstrated by the unsteady pitch-up maneuver analyzed in Section 2.10.2.3. Terms for unsteady and memory effects can be included, but for an arbitrary geometry, such phenomena are quite complex and not easily represented by a finite number of coefficients.

3. Complexity of the physical system - As is generally the case for any model, this group of models is tasked with reducing a complex physical system with no analytic solution (viscous fluid flow) to a set of readily solvable mathematical equations. By nature, a model will not have a direct relationship with the actual governing physics of the system. To provide a formulation that can feasibly be populated with values, these models must limit the number of phenomena they attempt to address. While some models do include terms that account for wakes being shed from upstream appendages (e. g., sails and external sensors), fully modeling the interaction of various factors would involve an

unfeasibly large number of coefficients. Generally speaking, the larger the number of coefficients included in a model, the harder the task of defining each coefficient becomes.

4. Model population - A state-space model is specific to a vehicle's geometry as well as its operating regime. Although a number of methods exist to obtain the coefficients that comprise a maneuvering model, each has its own limitations. If the model is to be populated experimentally, this can present a large and expensive testing workload. Another factor when using experimental population methods is the need to use a scaled model, which may create issues in maintaining dynamic similitude. Semi-empirical databases offer an alternative to experimentally population methods, but these can be less accurate and limited to use within a relatively narrow set of similar geometries.

Within the field of marine vehicles, most modern state-space models can trace their lineage back to the equations presented by Gertler and Hagen [3], which were later revised by Feldman [4]. These equations decompose the dynamics of a submarine into contributions from rigid-body dynamics, added mass, damping, control surfaces, hydrostatics and propulsion.[1] The development of these models was supported by the introduction of new experimental methods for characterizing vehicle hydrodynamics, such as planar motion mechanisms (PMMs) and rotating arm basins [5–9]. The work by a number of researchers has served to provide the hydrodynamic formulations that are the basis of these models [10–13]. While publication of research in this field has been somewhat limited due to the clandestine nature of naval submarine work, the recent surge in the development of AUVs has stimulated extensive research with increased dissemination. Many modern models are based closely on the vectorized formulation presented by Fossen [14–17].

Although a large variety of modeling approaches can be observed in the literature, most state-space models are constructed using fairly similar concepts and components. A detailed discussion of the concepts important to state-space modeling is presented in Chapter 3.

1.3.2 Free-Running CFD-based Maneuvering Simulations

Advances in computing power and algorithm development in the past decades have allowed a number of researchers to experiment with free-running CFD simulations, in which a vehicle is propelled through an infinite fluid and controlled in a dynamic manner. This dissertation uses the term *virtual free-running model*s (VFRMs) to refer to these simulations.

McDonald and Whitfield presented the first VFRM simulations of submarines, which focused on the rapid descent maneuver of the Defense Advanced Research Projects Agency (DARPA) SUBOFF geometry [18].[2] These simulations, which were computed in serial, relied on a

[1] For the purposes of later discussion, the Gertler-Hagen and Feldman equations of motion are reprinted in Appendices A.1.2 and A.1.3 respectively.

[2] The DARPA SUBOFF geometry was employed for CFD methodology validation in this study and is discussed in Section 2.10.1

change in ballast to initiate the submarine's maneuver (the control surfaces of the submarine were fixed with no deflection throughout the simulation). Two approaches were used to represent the submarine's propeller. In addition to simulating the actual rotation of a propeller, McDonald and Whitfield also completed simulations using an actuator disk.[1] While they were unable to make a direct comparison of results from these propeller representations due to time constraints, McDonald and Whitfield were able to show that this reduced complexity approach of modeling the propeller by an actuator disk lowered computational needs by approximately one order of magnitude.

Zierke et al. completed a study of VFRM simulations and performed validations on many of the CFD methodologies essential to these simulations [19]. The authors present preliminary work with deforming meshes to allow for dynamic deflection of control surfaces. These simulations used a single control surface rotating adjacent to a flat plate and were focused on demonstrating and analyzing the methodology for use in future VFRM simulations. Zierke et al. also presented the analysis of maneuvers similar to those studied by McDonald and Whitfield using propeller inputs. In these simulations, a loose coupling of the fluid dynamics and rigid-body kinematics was enforced, in which solutions were exchanged between the two systems at each time-step. A rapid stopping maneuver, commonly referred to as a "crashback," in which the vehicle is slowed using the reverse thrust of its propeller was conducted to exhibit the utility of VFRM simulations over traditional state-space models. Noting the significant computational expense of VFRM simulations, the authors devoted significant effort to improving computational efficiency and cited the need for additional work in future studies in this area. The authors also noted the need to implement control algorithm coupling in order to avoid the compounding accumulation of path errors over the course of a maneuver.

Since these two earliest studies, a number of researchers have performed VFRM simulations with maneuvers influenced by external forces applied to the vehicle so as to replicate those generated by deflected control surfaces [20–24]. These simulations avoid the substantial computational cost imposed by dynamic mesh methods needed to deflect the control surfaces throughout the simulation by altering the computational grid.[2] Instead,"fin-models" that describe the additional reaction created by the deflection of a control surface are constructed from theory, empirical data or steady simulations.[3] This allows for forces and moments to be applied "externally" during VFRM simulations. The fin-models developed in these studies showed fair-to-good agreement with force measurements from static testing [18, 20, 24].

Poremba employed an interesting approach, in which fluid momentum was affected in regions surrounding the control surfaces to replicate the presence of a deflected control surface [24]. This allowed for fore-mounted control surfaces to affect fluid flow on downstream structures. Unsteady maneuvering simulations were compared to free-running experimental tests. Discrepancies between the VFRM simulations and model tests were attributed to errors in

[1]This approach, which is discussed in Section 2.7, uses a source term in the conservation of momentum equations to add velocity to the flow without the presence of a rotating propeller.

[2]See Section 2.8.1 for more on the subject of dynamic meshes.

[3]See Section 3.6.1 for more on the topic of fin models.

measurement of the rigid-body inertial properties and the relatively simple manner in which control forces were calculated.

Dreyer and Boger performed a comparison of a VFRM simulation and experimental model driven by a control algorithm [22]. This study focused on a scenario in which a shallowly submerged submarine (which is under way) is overtaken by a large surface ship. The pressure disturbance created by the surface ship caused a change in the path of the submarine. Experimental measurements were taken using a free-running model submarine and carriage mounted surface ship. The path of the surface ship was set in a pre-prescribed fashion, while the submarine was guided by an autopilot. This scenario was replicated in the numerical simulations. In general, the numerical simulations compared quite favorably with the experimental results. While certain discrepancies were evident, some of these were apparently due to inconsistencies in controller gain settings. Another point of concern cited by the authors was the need to explore the relationship between time discretization within the CFD-simulation (i. e., time-step size) and the sampling rate of the autopilot controller.

A series of studies by Pankajakshan et al. employed deforming mesh techniques, with iterative remeshing to allow for larger deflections of control surfaces [25–27]. In one study, VFRM simulations of horizontal and vertical overshoot maneuvers were compared with similar maneuvers performed by a free-running experimental model. The trajectories predicted by the VFRM simulation compare quite well with those observed in the experiment.

Although most VFRM simulation research has been confined to the application of underwater vehicles, a number of researchers at the University of Iowa have conducted maneuvering simulations on surface combatants [28–32]. These studies have focused mostly on zig-zag maneuvers and broaching events. Research on surface ship maneuvering from a number of groups was presented at the inaugural SIMMAN Workshop on Verification and Validation of Ship Manoeuvring Simulation Methods (`http://www.simman2008.dk/`) in 2008 [33]. These conferences are focused on the comparison of numerical maneuvering predictions methods with experimental data. A similar conference is planned for 2014 (`http://www.simman2014.dk/`).

1.4 Outline of Dissertation

While the usage of VFRM simulations in maneuvering analysis is likely to grow in years to come with a continuing increase in computing power, it will none-the-less remain an expensive option in comparison to state-space modeling. There remains a need to better understand VFRM simulations as an analysis tool and to characterize their performance in comparison with state-space models as well as experimental data. In that interest, the remainder of this dissertation is divided into a series of chapters as follows:

1. Computational Fluid Dynamics Methodology - A general overview of the CFD methodology employed in this study is discussed. This chapter covers numerical formulation

and solution of Reynolds-averaged Navier Stokes equations as well as some higher level methods such as propeller modeling. Validations of these methodologies are also presented.

2. Quasi-Steady State-Space Models - In this chapter, the theory and formulation of quasi-steady state-space models is discussed. A number of modeling approaches and formulations are selected for further consideration. The population of state-space models for the GPAUV is presented.

3. Virtual Free-Running Model (VFRM) Simulations - Within this chapter, the execution of VFRM simulations and analyses is presented along with comparisons to field test data and state-space models. Details pertaining to the development of VFRM simulations beyond the CFD methodology covered in Chapter 2 are also discussed.

4. Discussion - Implications of the presented research and suggestions for future work are discussed.

1.5 Contributions

This dissertation presents a number of contributions to the field of maneuvering analysis. Based on an exploration of the theoretical concepts that are the basis for most state-space models used in underwater vehicle maneuvering analysis, a novel viscous damping formulation for a port-starboard symmetric vehicle was conceived and analyzed. This model compares favorably with the widely-used Gertler-Hagen model [3]. A system identification process using the results from CFD simulations, was developed to provide the coefficients for state-space models. This type of approach does not rely on the presence of information about similar vehicles (as is the case for database regression methods) and can be used before the completion of an experimental model. An additional advantage of this numerically based approach is its ability to be used to evaluate the effect of configuration changes before deployment (e. g., adding some additional external sensor).

The VFRM simulations presented in this model are among the first maneuvering simulations of their kind. These simulations employ an overset mesh configuration to allow for the dynamic deflection of a vehicle's control surfaces. This method of accounting for control surface contributions is more directly linked to the physical system than the fin-models which are often used instead. In addition to developing VFRM simulations, this dissertation also focuses on characterizing their performance and accuracy. To the author's knowledge, this study is the first in which VFRM simulations, state-space model(s) and experimental trial data are compared side-by-side. The "prescribed motion" comparison approach used to analyze these different maneuvering prediction methods is also unique to this dissertation.

Chapter 2

Computational Fluid Dynamics Methodology

Much of this dissertation is based on the application and analysis of computational fluid dynamics simulations. Steady and unsteady Reynolds-averaged Navier-Stokes (RANS) simulations were performed using CD-adapco's STAR-CCM+ software package [34]. This commercial code based approach has allowed research to focus on the application of a range of fluid dynamic formulations, solution algorithms, meshing schemes and post-processing. As this body of research focuses more on the use of CFD as tool than of an area of study in and of itself, an extended explanation of numerical formulations and methods is omitted. However, a brief explanation of the employed formulations and methods is essential for certain topics of discussion.

2.1 The Finite Volume Method

In the finite volume (FV) method, a discretized version of governing equations is solved within a set of finite control volumes (see Figure 2.1) that make up a computational domain. Each control volume (i. e., "cell"), contains a node at which flow variables are calculated (this node is denoted by P in Figure 2.1). The calculation of a node values requires that volume and surface integrals be taken over its cell. A number of methods are available for approximating these integrals. Second-order accurate methods are employed in the simulations presented in this dissertation(see [34, 35] for formulations). In unsteady simulations, a second-order central difference scheme was used for temporal discretization. By enforcing conservation laws within each cell, global conservation within the entire domain is also achieved. This system allows for the creation of a set of equations to be solved via some numerical method.

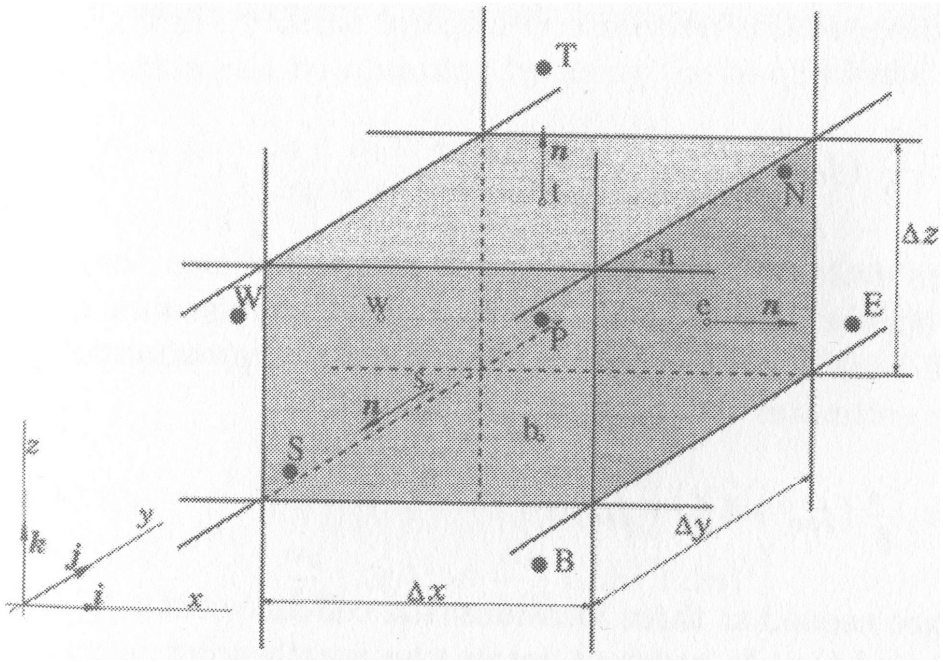


Figure 2.1: Diagram of finite volume element [35].

2.2 Reynolds-Averaged Navier-Stokes Formulation

At present, a range of formulations for simulating viscous fluid dynamics are available. While all simulations in this dissertation were executed using a RANS formulation, methods developed and applied throughout the study can be used with any level of numerical simulation (i.e., large-eddy simulations (LES) and detached-eddy simulations (DES)) with minor alterations.

2.2.1 Reynolds-Averaging

RANS formulations take advantage of the concept of statistical steadiness. Although a turbulent flow is by nature unsteady, this unsteadiness is often centered about some mean. A RANS formulation can be developed from the Navier Stokes equations through an averaging process, to operate on these mean values in a turbulent flow. The Navier Stokes equations for a incompressible fluid can be written as

$$\vec{\nabla} \cdot \vec{v} = 0 \tag{2.1}$$

$$\rho\left(\frac{\partial\vec{v}}{\partial t}+\vec{v}\cdot\vec{\nabla}\vec{v}\right)=-\vec{\nabla}p+\mu\vec{\nabla}^2\vec{v}+\mathfrak{b}. \tag{2.2}$$

Considering a three-dimensional Cartesian coordinate system, with components $x_i, (i=1,2,3)$ and velocities in those directions $\vec{v}=[u_1\ u_2\ u_3]^T$, (2.1) and (2.2) can be written in Einstein notation as

$$\frac{\partial u_i}{\partial x_i}=0 \tag{2.3}$$

$$\rho\left(\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}\right)=-\frac{\partial p}{\partial x_i}+\frac{\partial t_{ji}}{\partial x_j}, \tag{2.4}$$

The body force term in (2.2), $\mathfrak{b}$, has been dropped for simplicity. The term t_{ji} is the viscous stress tensor, defined by

$$t_{ji}=2\mu s_{ij}. \tag{2.5}$$

where s_{ij} is the strain-rate tensor

$$s_{ij}=\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right). \tag{2.6}$$

By applying (2.3), the convective term in (2.4) can be rewritten and reduced.

$$\begin{aligned} u_j\frac{\partial u_i}{\partial x_j}&=\frac{\partial}{\partial x_j}\left(u_ju_i\right)-u_i\frac{\partial u_j}{\partial x_j}\\ &=\frac{\partial}{\partial x_j}\left(u_ju_i\right)\end{aligned} \tag{2.7}$$

Applying these changes to (2.4) yields

$$\rho\frac{\partial u_i}{\partial t}+\rho\frac{\partial}{\partial x_j}\left(u_ju_i\right)=-\frac{\partial p}{\partial x_i}+\frac{\partial t_{ij}}{\partial x_j}. \tag{2.8}$$

The velocity, u_i, can be represented by the combination of a mean,[1] $\bar{u}_i$, and a fluctuation about that mean, u_i'.

$$u_i\left(x,t\right)=\bar{u}_i\left(x\right)+u_i'\left(x,t\right) \tag{2.9}$$

[1]Time averaging is used in steady simulations and ensemble averaging is used in unsteady simulations.

Looking back to (2.8), it becomes apparent that in addition to representing the velocity as a combination of a mean and fluctuation, we must also represent the mean of velocity products.

$$\overline{u_i u_j} = \overline{\left(\bar{u}_i + u_i'\right)\left(\bar{u}_j + u_j'\right)} = \overline{\bar{u}_i\bar{u}_j + \bar{u}_i u_j' + \bar{u}_j u_i' + u_i' u_j'} \tag{2.10}$$

The means of two of these fluctuating quantities are zero ($\overline{u_i'} = \overline{u_j'} = 0$). Therefore the terms which are the mean of a product of a mean quantity and a fluctuating quantity are also zero ($\bar{u}_i u_j' = \bar{u}_j u_i' = 0$). Thus, (2.10) reduces to

$$\overline{u_i u_j} = \bar{u}_i\bar{u}_j + \overline{u_i' u_j'} \tag{2.11}$$

If a similar approach is adopted for the pressure, p, the Reynolds-averaged Navier-Stokes equations can be written as

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \tag{2.12}$$

$$\rho\frac{\partial \bar{u}_i}{\partial t} + \rho\frac{\partial}{\partial x_j}\left(\bar{u}_j\bar{u}_i + \overline{u_j' u_i'}\right) = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu\bar{s}_{ij}\right) \tag{2.13}$$

The term $\bar{s}_{ij}$ is the Reynolds-averaged version of the strain-rate tensor.

$$\bar{s}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{2.14}$$

These averaged equations are similar to the full Navier-Stokes equations, with the addition of the *Reynolds stress tensor* term, $\overline{u_j' u_i'}$. Often, this term is moved to the right-hand side of the momentum equation and the convective term is written in fashion of (2.4).

$$\rho\frac{\partial \bar{u}_i}{\partial t} + \rho\bar{u}_j\frac{\partial \bar{u}_i}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu\bar{s}_{ij} - \rho\overline{u_j' u_i'}\right) \tag{2.15}$$

2.2.2 Turbulence Closure Models

Observing (2.12) and (2.15), there exist ten unknowns for a three-dimensional problem (P, U_x, U_y, U_z and the six elements of symmetric matrix $u_i' u_j'$) and only four equations. There is thus a need to devise a closure for the system, which gives rise to what are generally referred to as turbulence closure models (TCM), or simply turbulence models.

The turbulence models applied in this study rely on the Boussinesq eddy-viscosity assumption, which is based on the link between viscosity and turbulence: turbulence in a flow causes

dissipation and transport (of energy, momentum and mass) in directions normal to flow streamlines. These phenomena are also directly related to viscosity. In recognition of this connection, many turbulence models rely on the concept of an increased viscosity in turbulent flow, referred to as the *eddy-viscosity*. As both turbulence and viscosity increase the transport of flow properties, an increase in turbulence can be modeled with an increase in viscosity. While this concept is physically inaccurate, methods that rely on this assumption can none-the-less provide realistic results. The Boussinesq assumption can be written as

$$-\rho\overline{u_i'u_j'} = 2\mu_t\bar{s}_{ij} - \frac{2}{3}\rho\delta_{ij}k \tag{2.16}$$

where μ_t is the eddy-viscosity, δ_{ij} is the Kronecker-delta function and k is the turbulent kinetic energy, which can be given by

$$k = \frac{1}{2}\overline{u_i'u_i'}. \tag{2.17}$$

With this assumption, the number of unknowns is reduced from from ten to six ($\bar{p}$, $\bar{u}_x$, $\bar{u}_y$, $\bar{u}_z$, μ_t and k). The job of a chosen TCM is to provide a means of solving for μ_t and k. A detailed description of turbulence closure modeling and the formulation of the k-ω SST model used for most of the simulations in this study is given by Wilcox [36].

2.3 SIMPLE Solution Algorithm

All simulations in this study employed a semi-implicit method for pressure-linked equations (SIMPLE) solution algorithm [37]. This method has been shown to be an efficient means of solving the incompressible RANS equations. The iteration scheme of the SIMPLE algorithm and similar predictor-corrector style algorithms can be outlined as follows [34, 35].

1. Set velocity and pressure to values from previous iteration.
2. Solve the momentum equation for intermediate velocity field.
3. Solve the pressure-correction equation for pressure correction field.
4. Correct the velocity field so that it satisfies the continuity equation and the corrected pressure field.
5. Progress to the next iteration.

2.4 Hydrodynamic Forces and Moments

2.4.1 Relation to Pressure and Shear

The forces and moments experienced by a body as a result of fluid dynamic pressure and shear stress are of central importance to this study. These are readily available from the fluid field solution. The pressure and shear force on each cell face of a rigid body can be expressed as

$$\vec{F}_c^{\text{pressure}} = p_c \vec{a}_c \tag{2.18}$$

$$\vec{F}_c^{\text{shear}} = -T_c \cdot \vec{a}_c. \tag{2.19}$$

Here, p_c, T_c and $\vec{a}_c$ are the pressure, viscous stress tensor and cell area vector. The total force on the body in the $\hat{n}_i$ outward direction can then be determined by the sum

$$F_i = \sum_c \left(\vec{F}_c^{\text{pressure}} + \vec{F}_c^{\text{shear}} \right) \cdot \hat{n}_i. \tag{2.20}$$

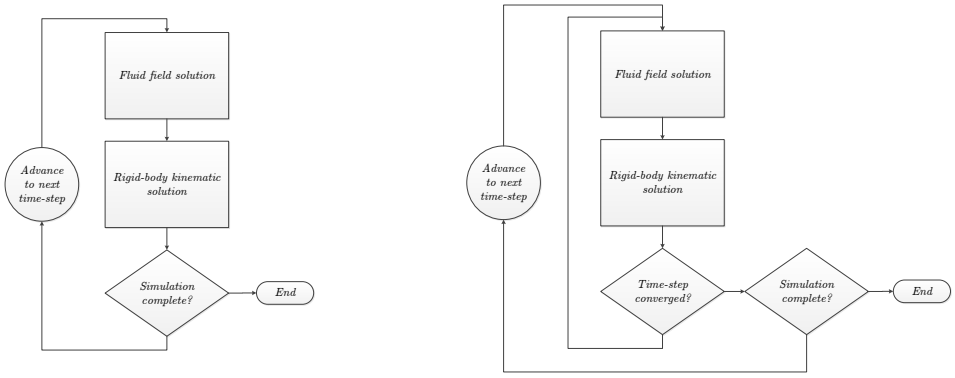
Likewise, the moment about an axis $\hat{n}_i$ passing through the point x_0 is given by

$$M_i = \sum_c \left[\vec{r}_c \times \left(\vec{F}_c^{\text{pressure}} + \vec{F}_c^{\text{shear}} \right) \right] \cdot \hat{n}_i, \tag{2.21}$$

where $\vec{r}_c$ is the position of the each cell face relative to x_0.

2.4.2 Effect on Rigid-Boy Dynamics

In a 6-DoF CFD simulation, the forces and moments due to the fluid are coupled to a rigid-body kinematic model, which uses the equations given in Appendix A.1.1. The linking of these models can be a loose style of coupling (Figure 2.2a) or a tight coupling (Figure 2.2b). In a loose coupling method, the fluid solution is applied to the rigid-body kinematic model at the end of each time-step. In the tight coupling procedure, the fluid solution and rigid-body kinematic solution are solved repeatedly during each time-step until the change in each solution is sufficiently small. While more computationally intensive, the tight coupling scheme is better suited to simulations where high accelerations are possible. STAR-CCM+ implements a tight coupling scheme.



(a) Loose coupling **(b)** Tight coupling

Figure 2.2: CFD and rigid-body kinematic coupling.

2.5 Mesh Generation

Mesh generation for this study was performed within STAR-CCM+. Simulations were conducted on unstructured polyhedral meshes, with body-fitted prismatic cells applied to walls in order to better resolve boundary layer flow, generally resulting in a wall y^+ value in the first layer of cells less than 10 (often this value was less than 1, specific information is presented for each simulation's mesh). Regions predicted or observed to experience high gradients were further resolved through local increases in mesh resolution.

2.6 User-coded Functions

STAR-CCM+ allows for three levels of user-coding. The most basic of these approaches are referred to as Field Functions, which use a syntax based on C++ and are compiled within the STAR-CCM+ graphic user interface (GUI). This approach is useful for most basic functions, but the syntax can make multi-part functions fairly cryptic. Program macros, written in Java^{TM}, provide a better means of writing more complex functions. Java macros have an added benefit in their capability to employ any of the many openly available libraries. The third form of user-code, referred to as User Code within STAR-CCM+, is in some ways the most comprehensive, as it can be used to control field variables within the simulation and can thus be used to create supplementary physics formulations.

While some of the preliminary simulations described in this dissertation use Field-Functions for user-coding needs, Java macros are the preferred user-coding method applied this study. A number of the macros used for various purposes throughout the study are provided in Appendix B.

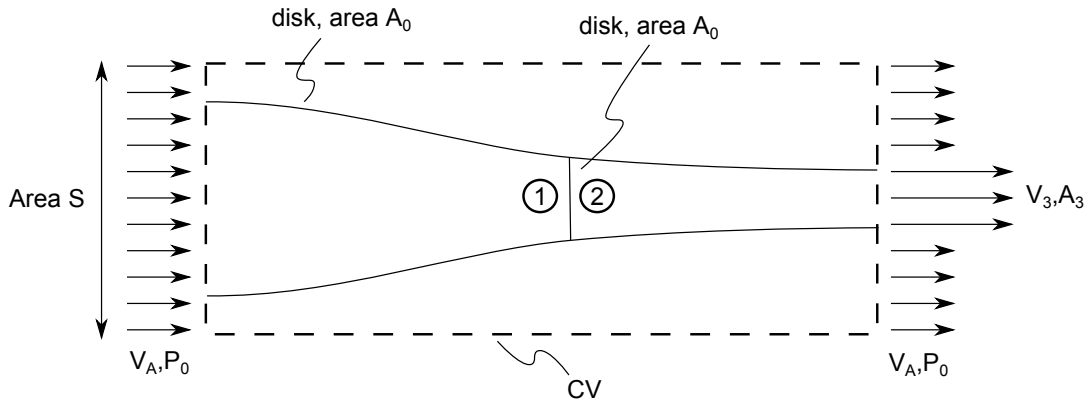


Figure 2.3: Actuator disk concept illustration [39].

2.7 Propeller Modeling

In order to accurately represent the flow over the stern of an underwater vehicle, it is necessary to model the upstream influence of the vehicle's propulsor. Simulating a rotating propeller is computationally expensive in the it: (1) generally requires additional cells be added the computational domain to capture the propeller's geometry and (2) puts additional constraints, based on the propeller's RPM, on the time-step size of unsteady simulations. Rather than imposing the computational cost of a rotating propeller, an actuator disk model was developed.

2.7.1 Actuator Disk Theory

Actuator disks are simple, traditionally two-dimensional, representations of propellers, where the effect of a propeller on the fluid is achieved via a discontinuity in pressure [38]. The propeller is idealized as a simple disk, and a sudden increase in pressure is used to accelerate the fluid. Consider the system depicted in Figure 2.3, where an actuator disk is located within a control volume large enough to have a uniform pressure distribution, P_0, over its entire surface.

If continuity is enforced for the control volume, the increased volume flow rate downstream of the actuator disk ($Q_{\text{downstream}} = A_3 V_3 + (S - A_3)\, V_A$) requires an increased flow enter upstream of the disk. This flow rate difference can be defined as

$$\begin{aligned} \Delta Q &= [Q_{\text{downstream}}] - [Q_{\text{upstream}}] \\ &= [A_3 V_3 + (S - A_3) V_A] - [S V_A] \\ &= A_3 \left(V_3 - V_A\right). \end{aligned} \tag{2.22}$$

In (2.22), A_i and S refer to areas while V_i denote fluid velocities. The momentum equation for the control volume in Figure 2.3 can be written as

$$\iint T dA = \iint \vec{V} \left(\rho \vec{V} \cdot d\vec{A} \right) + \frac{\partial}{\partial t} \iiint V \rho \, d\forall, \tag{2.23}$$

where T is the force on the fluid and ρ is its density. Applying (2.23) in the axial direction and solving for T gives

$$\begin{aligned} T &= \rho \left[-S{V_A}^2 - \Delta Q V_A + A_3 {V_3}^2 + (S - A_3) {V_A}^2 \right] \\ &= \rho \left[-A_3 (V_3 - V_A) V_A + A_3 {V_3}^2 + A_3 {V_A}^2 \right] \\ &= \rho A_3 V_3 (V_3 - V_A) \end{aligned} \tag{2.24}$$

If conservation of mass is enforced in the streamtube such that $\rho A_3 V_3 = \rho A_0 V_1$, (2.24) can be reduced.

$$T = \rho A_0 V_1 (V_3 - V_A) \tag{2.25}$$

The force on the fluid can be set equal to that on the actuator disk due to the pressure difference across the disk. Letting p_1 and p_2 represent the fluid pressure upstream and downstream of the disk respectively, the force on the disk and fluid is given by

$$T = A_0 (p_2 - p_1) \tag{2.26}$$

Setting (2.26) equal to the result of (2.25) gives

$$(p_2 - p_1) = \rho V_1 (V_3 - V_A), \tag{2.27}$$

As Bernoulli's equation cannot be applied across the discontinuity at the actuator disk, separate expressions for the fluid upstream and downstream of the disk must be considered.

$$p_0 + \frac{1}{2} \rho V_A^2 = p_1 + \frac{1}{2} \rho V_1^2 \tag{2.28}$$

$$p_0 + \frac{1}{2} \rho V_3^2 = p_2 + \frac{1}{2} \rho V_2^2 \tag{2.29}$$

Taking the difference between (2.28) and (2.29) gives

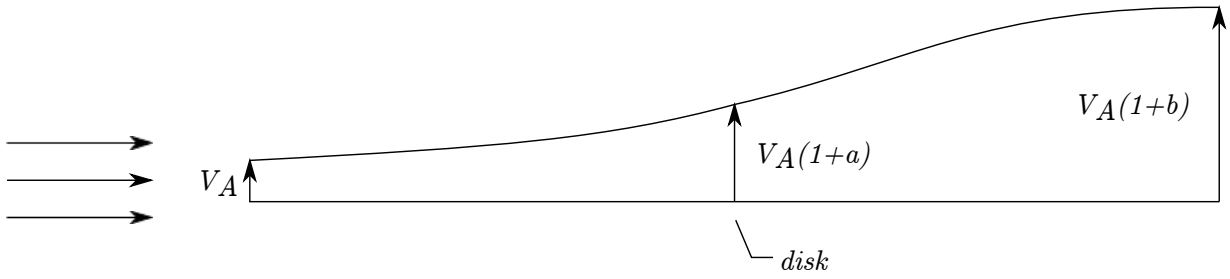


Figure 2.4: Velocity variation across a theoretical actuator disk [39].

$$\begin{aligned} p_2 - p_1 &= \frac{1}{2}\rho\left(V_3^2 - V_A^2\right) \\ &= \frac{1}{2}\rho V_2^2\left(V_3 + V_A\right)\left(V_3 - V_A\right). \end{aligned} \tag{2.30}$$

This result can be further reduced by applying (2.27).

$$V_1 = \frac{1}{2}\left(V_3 + V_A\right) \tag{2.31}$$

Thus, (2.31) shows that half of the velocity increase created by the actuator disk occurs upstream of the disk, while the remaining fluid acceleration occurs downstream.

Often, the stream tube velocity, V_3, is redefined as

$$V_3 = V_A\left(1 + b\right) \tag{2.32}$$

and the velocity directly upstream of the disk, V_1, is redefined as

$$\begin{aligned} V_1 &= V_A\left(1 + \frac{b}{2}\right) \\ &= V_A\left(1 + a\right); \text{ where } a = \frac{b}{2}, \end{aligned} \tag{2.33}$$

Figure 2.4 shows an illustration of the velocity variation across the actuator disk using the parameters defined in (2.32 - 2.33). In this style, the thrust can be written as

$$T = \rho A_0 V_A^2\left(1 + a\right) b \tag{2.34}$$

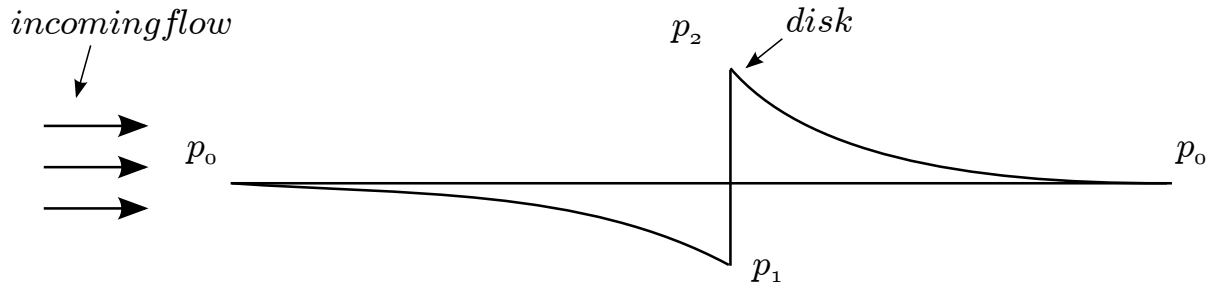


Figure 2.5: Pressure variation across a theoretical actuator disk [39].

Returning to (2.28) and (2.29), the pressure variation upstream and downstream of the disk can be defined by

$$p_1 = P_0 - \rho V_A^2 a \left(1 + \frac{a}{2}\right) \tag{2.35}$$

$$p_2 = P_0 + \rho V_A^2 a \left(1 + \frac{3a}{2}\right). \tag{2.36}$$

This pressure distribution is depicted in Figure 2.5.

A similar derivation can be used to deduce the method for adding a rotational energy to flow via an actuator disk.

2.7.2 Actuator Disk Implementation within CFD Simulations

The actuator disk used for this study is similar to that described in Section 2.7.1, with some alterations to allow for application within a CFD environment. The implementation of an actuator disk within a CFD environment has been demonstrated within a number of studies [18, 19]. While no explicit module for this ability exists within STAR-CCM+, the software does allow for the prescription of a momentum source within a designated volume, which, in turn, permits for relatively straightforward implementation of a three-dimensional actuator disk.

The momentum source term, $\mathfrak{b}^{\text{prop}}$, with units $\left[\frac{\text{Force}}{\text{Volume}}\right]$ for axial flow and $\left[\frac{\text{Moment}}{\text{Volume}}\right]$ for circumferential flow, is given in cylindrical coordinates by

$$\mathfrak{b}^{\text{prop}} = \left[\begin{array}{ccc} 0 & \frac{T}{\forall} & \frac{Q}{\forall}, \end{array}\right] \tag{2.37}$$

where T and Q are the propeller's thrust and torque as determined from the propeller performance curves. The first entry in the momentum source term, $\mathfrak{b}_1^{\text{prop}}$, in the radial

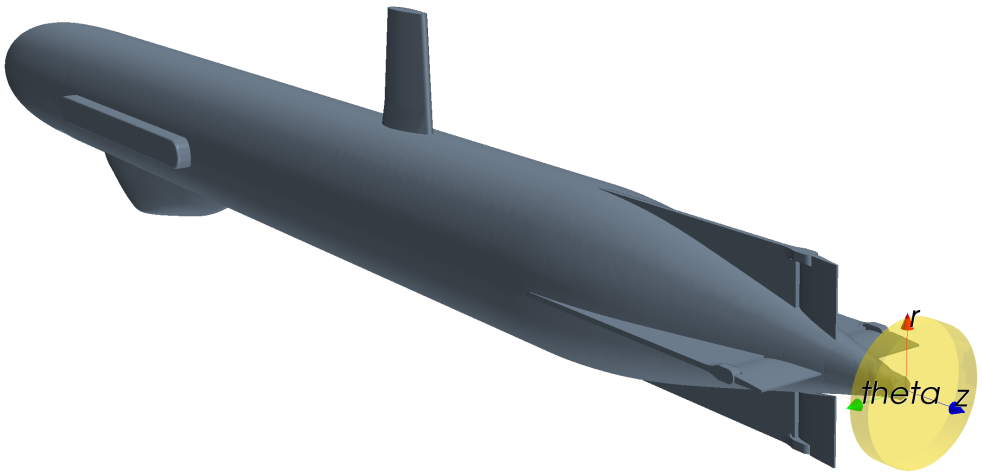


Figure 2.6: Local coordinate system for actuator disk (*ActuatorDiskCsys*).

direction is zero. The momentum source term for the actuator disk can be applied to the numerical simulation via (2.2).

It is important to note that while the actuator disk can add momentum to the fluid, it contains no surface on which pressure forces can act and therefore does not supply a thrust force (or torque) to a vehicle. The thrust and torque from the propeller are therefore applied separately via an external force and moment in the 6-DoF rigid-body kinematic model.

2.7.2.1 Local Coordinate System

A local cylindrical coordinate system, titled *ActuatorDiskCsys*, is used to designate the actuator disk's operation. The orientation of *ActuatorDiskCsys* is shown in Figure 2.6. Here, the positioning of the θ axis is irrelevant as the loading of the actuator disk is assumed to be axisymmetric.[1]

2.7.2.2 Propeller Loading Distribution

The actuator disk momentum output presented in (2.37) represents a flat (constant) radial loading distribution. Marine propellers are known to have a non-uniform radial distribution of loading. If the radial distribution of the momentum added by a marine propeller is considered,

[1] A time-accurate version of an actuator disk generally called *actuator lines*, can be used to simulate the effect of each blade individually. This approach has been used by a number of researchers to model ship propulsion [40] as well as wind and tidal turbines [41–44].

the maximum momentum is added to the flow near three-fourths the maximum radius while almost no momentum is added at the blade tip or root. Experimental data on the radial loading distribution, $\xi(r)$, for the GPAUV's propeller was unavailable. Instead, it was assumed to be similar to other marine propellers and therefore satisfy the following conditions.

$$\begin{aligned} \xi(r = R_{hub}) &\cong 0 \\ \xi(r = R_{blade}) &\cong 0 \\ \left.\frac{d\xi}{dr}\right|_{r=\frac{3}{4}R_{blade}} &\cong 0 \end{aligned} \tag{2.38}$$

For the GPAUV, the limits of the propeller blades, R_{hub} and R_{blade}, are 0.9 cm and 6 cm respectively.

To insure that the overall thrust delivered remains unchanged by the loading distribution (i. e., the thrust produced by the radially distributed load actuator disk matches that produced by a constant load actuator disk), the loading distribution must satisfy

$$\int_0^{2\pi}\int_{R_{hub}}^{r_{blade}} (1)\, r\, dr d\theta = \int_0^{2\pi}\int_{R_{hub}}^{r_{blade}} \xi\left(\frac{r}{R}\right) r\, dr d\theta. \tag{2.39}$$

Based on these constraints, the following function was chosen to represent the radial loading distribution of the propeller.

$$\xi\left(\frac{r}{R}\right) = \begin{cases} 0, & \text{for } \left(\frac{r}{R}\right) \leq 0.2 \\ \left(0.945\left(\frac{r}{R}\right)^6 + 1.182\left(\frac{r}{R}\right) + 0.236\right)\pi, & \text{for } \left(\frac{r}{R}\right) \geq 0.2 \end{cases} \tag{2.40}$$

This distribution is shown in Figure 2.7. The application of this radial loading distribution leads (2.37) to be rewritten as

$$\mathfrak{b}^{\text{prop}}\left(\frac{r}{R}\right) = \xi\left(\frac{r}{R}\right)\left[\begin{array}{ccc} 0 & \frac{T}{\forall} & \frac{Q}{\forall} \end{array}\right]. \tag{2.41}$$

2.7.2.3 Propeller Performance

The actuator disk implemented in CFD simulations is based directly on the propeller used by the GPAUV. The GPAUV's propeller has two blades with a maximum diameter of $D = 0.12\,\text{m}$. It is located approximately $0.15D$ aft of the tail and $1D$ aft of the quarter chord of each

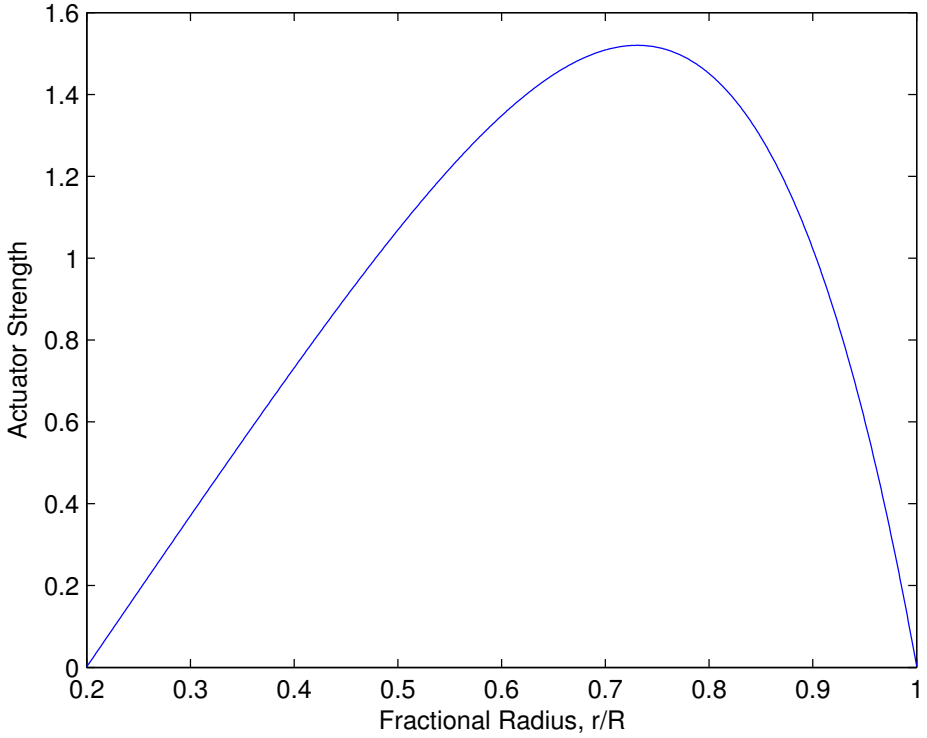


Figure 2.7: Radial distribution of the actuator disk momentum flux.

control surface. The propeller was characterized during the design process using analytic methods [45]. A rendering of the propeller geometry is shown in Figure 2.8.

The propeller performance curves for this propeller, based on the advance ratio

$$J = \frac{V_A}{nD} \tag{2.42}$$

were shown to be

$$\begin{aligned} K_T(J) &= 0.0477 + 0.1705J - 0.4404J^2 + 0.2072J^3 \\ K_Q(J) &= 0.0042 + 0.0058J - 0.0117J^2 \end{aligned} \tag{2.43}$$

The coefficients K_T and K_Q are nondimensional representations of the thrust and torque created by the propeller.

$$\begin{aligned} K_T &= \frac{T}{\rho n^2 D^4} \\ K_Q &= \frac{Q}{\rho n^2 D^5} \end{aligned} \tag{2.44}$$

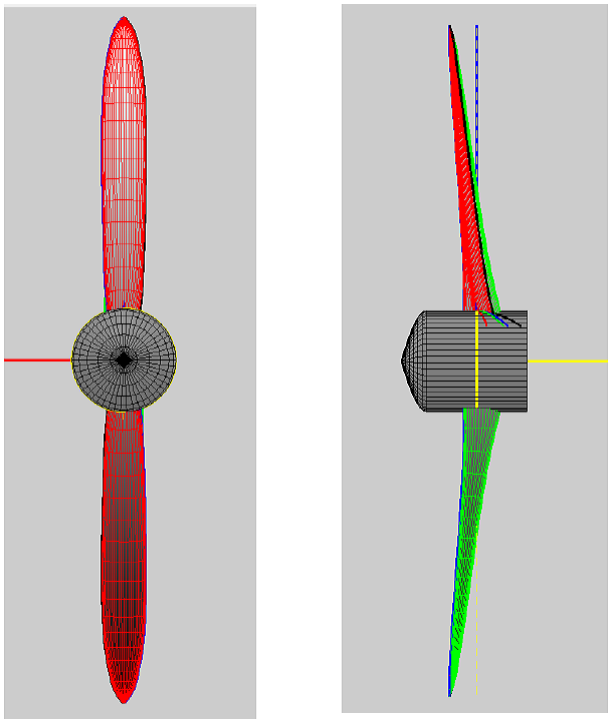


Figure 2.8: GPAUV propeller rendering [45].

The propeller performance curves defined by (2.43) are plotted in Figure 2.9.

2.7.2.4 Control

The output of the actuator disk is controlled by the propeller performance curves shown in Figure 2.9. These curves are a function of advance ratio, J, defined in (2.42). With the propeller diameter remaining constant, the instantaneousness output of the actuator disk is a function of inlet velocity, V_A, and rotational rate, n. No active speed control is currently employed on the GPAUV. Instead the propeller RPM is set directly. As shown in Appendix B, the GPAUV's speed is taken from the state vector, ν, and a wake fraction is imposed to find the inlet velocity for the actuator disk.[1]

2.7.2.5 Results

Figure 2.10 shows the cylindrical actuator disk (with no hub) operating in steady flow. The effect of the propeller loading distribution function is evident in both figures. As noted in Section 2.7.1, there is both a downstream increase and upstream decrease in the pressure

[1] Another method of obtaining the inflow velocity would be to monitor it directly during the simulation. Early tests in this study using this approach suffered from instability issues.

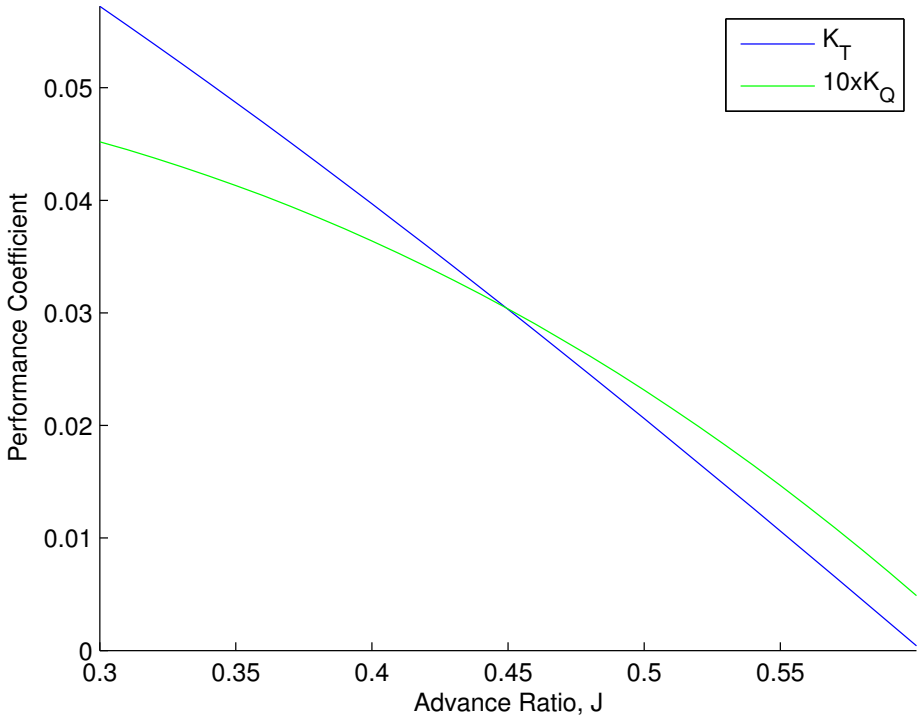


Figure 2.9: GPAUV propeller performance curves.

field. This effect on the pressure is visible in Figure 2.10b as well as Figure 2.11. This plot can be compared to the theoretical illustration shown in Figure 2.5.

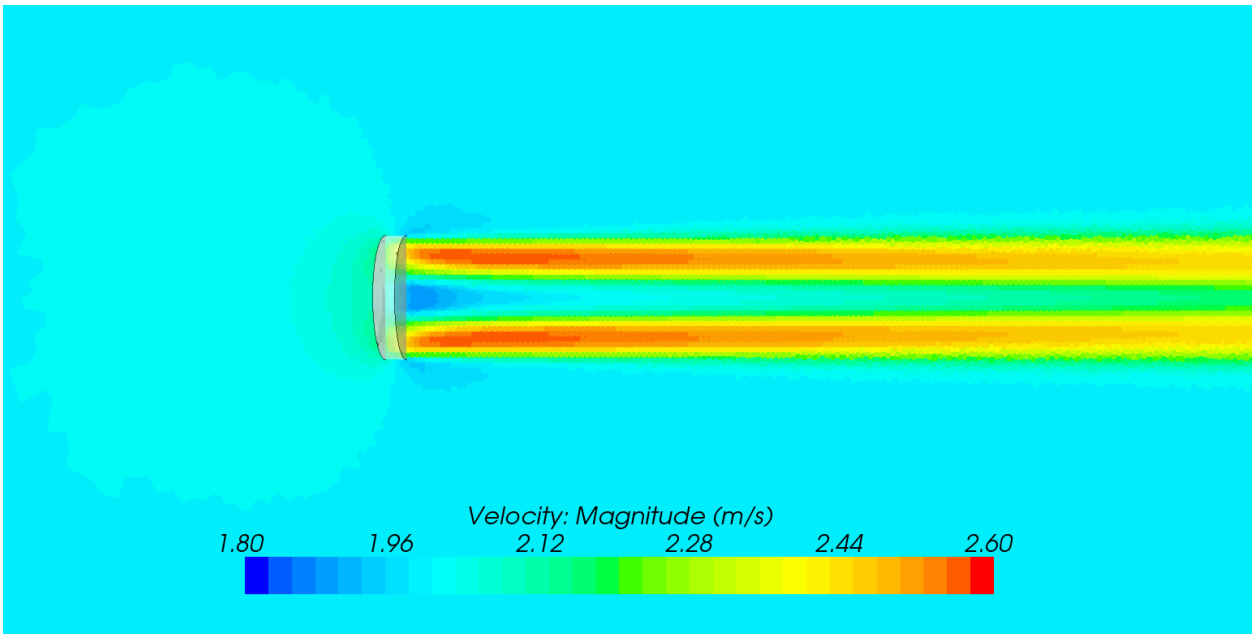
2.7.2.6 Validation

To ensure that the actuator disk performed as intended, a simple test test-rig simulation was devised in which the actuator disk was enclosed in a control volume, which is shown in Figure 2.12.

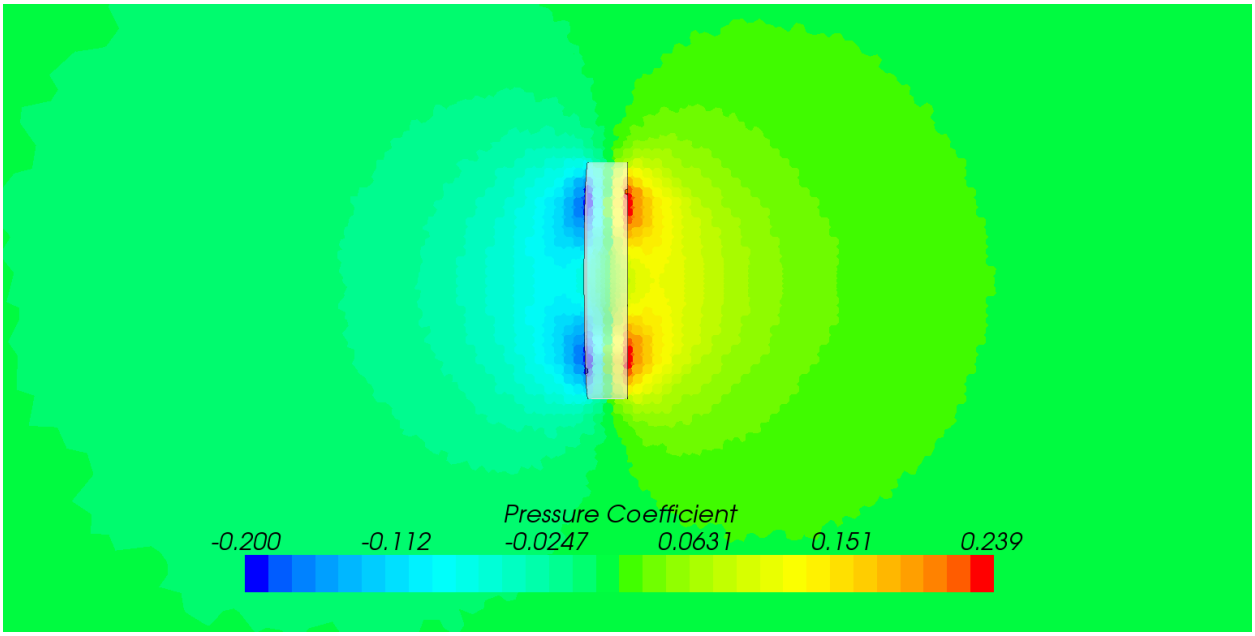
The thrust produced by the propeller was measured by taking the following expression over the control surface shown in Figure 2.12.

$$T = \iint \vec{V} \left(\rho \vec{V} \cdot d\vec{A} \right) + \iint p d\vec{A} \tag{2.45}$$

This expression for thrust, T, contains a momentum integral as well as a pressure integral, so that the control volume could be of a reasonably small size (to discard the pressure integral, the control volume would need to be drawn large enough so that the surfaces were all at a constant pressure). This approach allows for a comparison to be made between the thrust commanded and the thrust produced.



(a) Velocity magnitude



(b) Pressure coefficient

Figure 2.10: Actuator disk operating with plane to show velocity magnitude and pressure coefficient scalar.

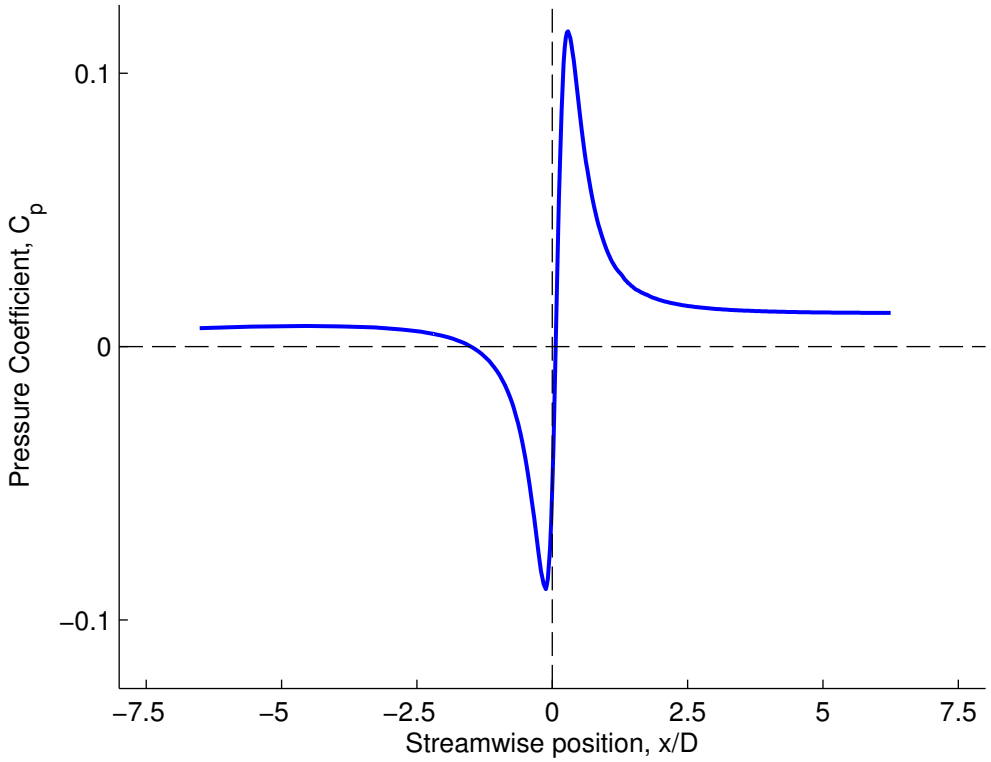


Figure 2.11: Streamwise pressure coefficient in simulation due to operating actuator disk in CFD simulation.

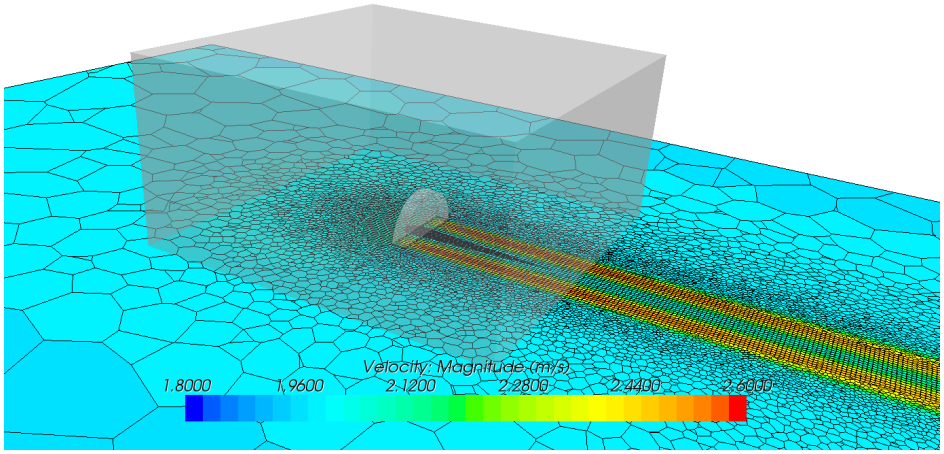
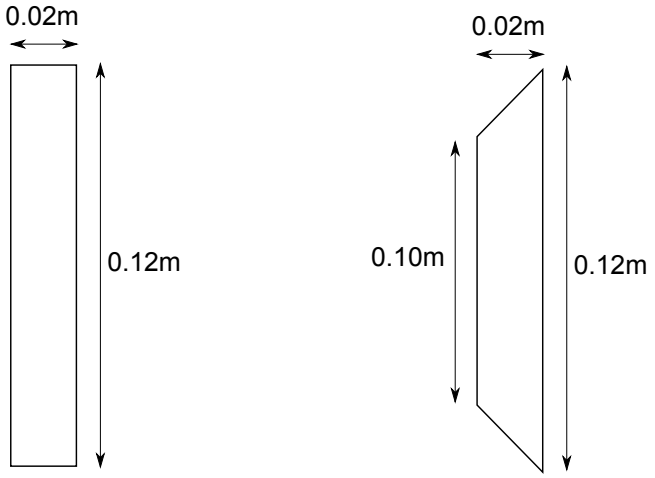


Figure 2.12: Actuator disk operating in control volume.



(a) Cylinder **(b)** Conical frusum

Figure 2.13: Momentum source region geometries considered for actuator disk.

Table 2.1: A performance comparison of actuator disk geometries.

Geometry	Commanded	Measured	% Error
Conical Frustum	1.12E+01	1.18E+01	5.18%
Cylinder	1.12E+01	1.12E+01	0.12%
Cylinder w/ hub	1.12E+01	1.10E+01	-1.86%

Four actuator geometries were analyzed using this test rig simulation. While a simple cylinder is the most straightforward shape to represent a propeller, early simulations completed for this project showed the presence of a fluid interface parallel to the freestream flow direction in a cylindrical shaped actuator disk to be somewhat unfavorable for numerical stability. While the interface can easily handle flow passing through at a nearly normal angle, flow crossing at very small angles can be problematic. Thus a conical frustum was also considered. Figure 2.13 shows the two actuator disk geometries considered in this study. Each was analyzed with and without a hub at their centers.

The results of this analysis, comparing the commanded thrust to the thrust obtained using (2.45), are shown in Table 2.1. The cylindrical actuator disks outperformed those with a conical frustum shape. The addition of a solid hub at the center of the disk, so as to mimic the hub located as the center of a propeller, degraded the accuracy of the propulsor. This reduction in measured thrust is believed to be due to viscous losses incurred as a result of the hub. Later consideration also revealed that the inclusion of a hub duplicates an already accounted for effect, as the propeller performance curves on which the output of the actuator disk is based were derived analytically using a propeller with a hub.

2.7.3 Propeller and Wake Analysis

A series of CFD simulations, analyzing the wake flow characteristics behind a predecessor of the GPAUV using an actuator disk, served as the basis for the GPAUVs propeller design [45].

2.7.3.1 Wake Fraction

The inflow to a propeller, and therefore its performance, is altered when operating in the wake of a vehicle (as opposed to uniform freestream flow). When operating within a wake, the inflow "seen" by the propeller will be of a different magnitude than the freestream flow and have a complex velocity distribution. The ratio of the flow velocity upstream of the propeller to the velocity of the freestream flow is referred to as the wake fraction, w_t.

$$w_t = \frac{V - V_A}{A} \tag{2.46}$$

This relation allows the inflow to the propeller to be determined from vehicle speed.

$$\begin{aligned} V_A &= V\,(1 - w_t) \\ &= u\,(1 - w_t) \end{aligned} \tag{2.47}$$

Numerical simulations of this predecessor to the GPAUV indicated a wake fraction of $w_t \cong 0.3$.

Figure 2.14 shows the radial distribution of flow seen by the propeller as a percentage of the freestream flow. This data was approximated by the following polynomial fit, which was used in the design process of the propeller [45].

$$\frac{u}{U_\infty} = 2.02\left(\frac{r}{R}\right)^3 - 5.02\left(\frac{r}{R}\right)^2 + 4.27\left(\frac{r}{R}\right) - 0.332 \tag{2.48}$$

2.7.3.2 Thrust Deduction Factor

Just as a propeller's performance is influenced by the presence of a vehicle, the drag on the vehicle is increased by the propeller. As noted in Section 2.7.1, the region upstream of a propeller experiences a substantial drop in pressure. The result of the lowered pressure on the tail of the vehicle is an increase in drag. This phenomenon is often accounted for with a thrust deduction factor, t_T.

$$t_T = \frac{T - R}{T} \tag{2.49}$$

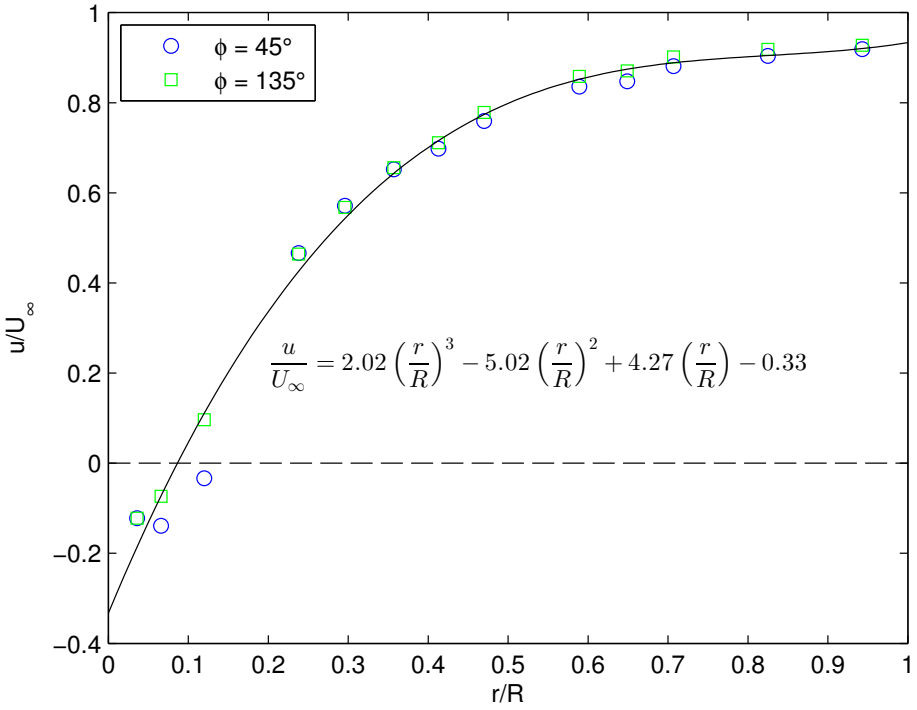


Figure 2.14: Radial propeller inflow distribution from CFD simulations, shown for two azimuthal locations: $\phi = 35°$ and $\phi = 135°$.

Here, T is the thrust required to propel the vehicle and R is the resistance experienced by the vehicle without an operating propeller (i. e., the "naked hull" resistance). Instead of altering the drag on the vehicle, the thrust deduction factor accounts for a decrease in the thrust produced by the propeller when operating behind a vehicle.

$$R = T\,(1 - t_T) \tag{2.50}$$

The thrust deduction factor for a predecessor of the GPAUV was found to be $t_T = 0.064$.

It should be noted that the thrust deduction factor does not need to be applied in a CFD simulation that incorporates an actuator disk, as the effect is replicated by the suction created by the actuator disk.

2.8 Dynamic geometry

Two levels of rigid-body motion are required for simulating the dynamic maneuvering of an AUV. The first is the motion of the vehicle as a whole, with respect to the fluid environment.

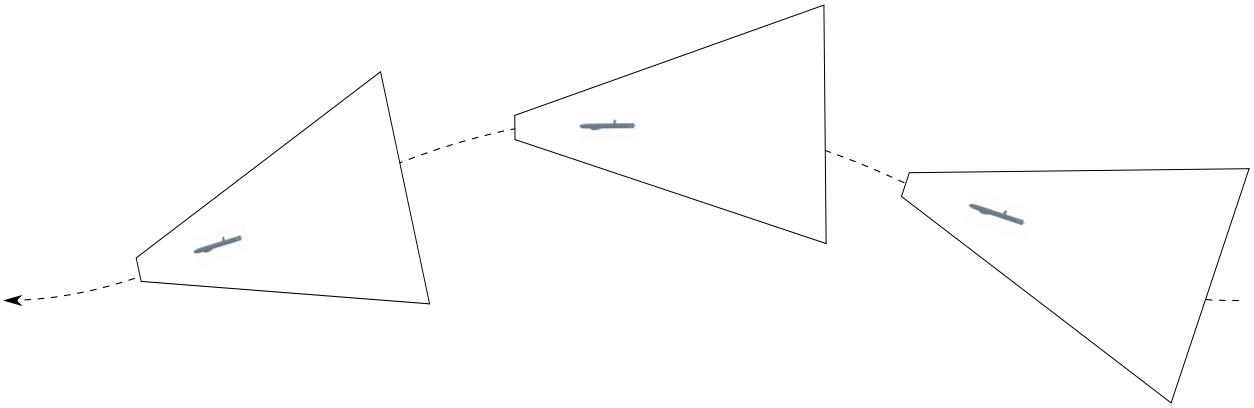


Figure 2.15: Illustration of the movement of the entire computational domain through an "infinite fluid."

In all cases considered for this study, this level of motion is accomplished by manipulating the boundary conditions of the computational domain. As the body and domain move in any arbitrary manner, the boundaries of the domain are set to always present a fluid at rest with respect to the simulation's inertial reference frame. In effect, the entire computational domain moves rigidly with the vehicle, throughout some infinite body of resting fluid (see Figure 2.15).[1]

The second motion of concern for a VFRM simulation, is that of the vehicle's control surfaces. These rigid appendages must move with the vehicle as a whole, but also rotate about axes to provide control forces to the AUV. Figure 2.16 shows the GPAUV's upper control surface during multiple stages of a deflection. Unlike the motion of the vehicle as a whole, the rotation of the control surfaces presents a change in the domain geometry that requires an alteration of the finite volume mesh.

2.8.1 Dynamic Mesh Approaches

A number of methods were considered to perform simulations involving relative motion (often referred to as two-body motion). The simulation domain shown in Figure 2.17 is used in the following sections to provide a means of demonstrating and discussing these methods. In all cases, the "mixer blade," colored purple in Figure 2.17, rotates in the positive direction about the z-axis at its center. The domain boundaries, colored gray in Figure 2.17, are set as stationary walls.

[1]This method of simulating the motion of the vehicle is appropriate when considering maneuvers in an unconfined environment (e. g., open ocean). To simulate the maneuvering of a vehicle around other objects or within a confined space, an overset mesh approach (discussed in Section 2.8.1.3) would be required.

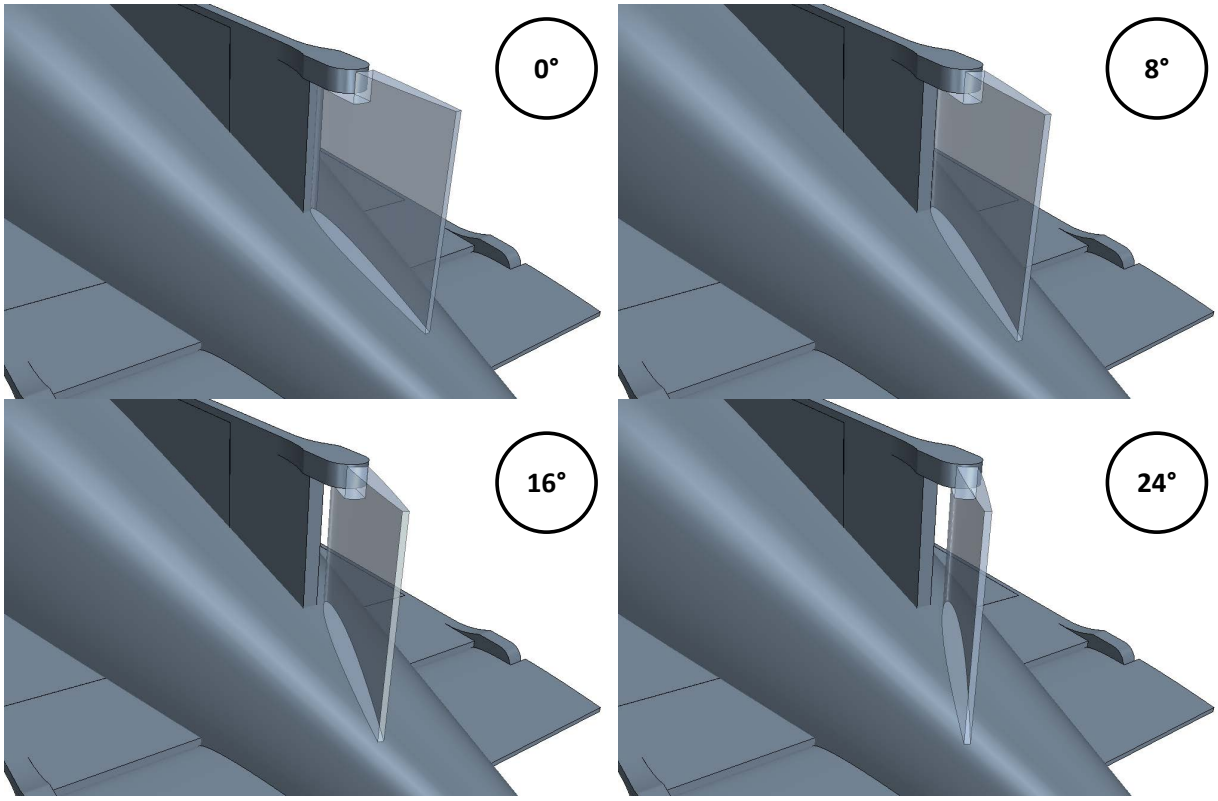


Figure 2.16: GPAUV upper rudder at a range of deflections.

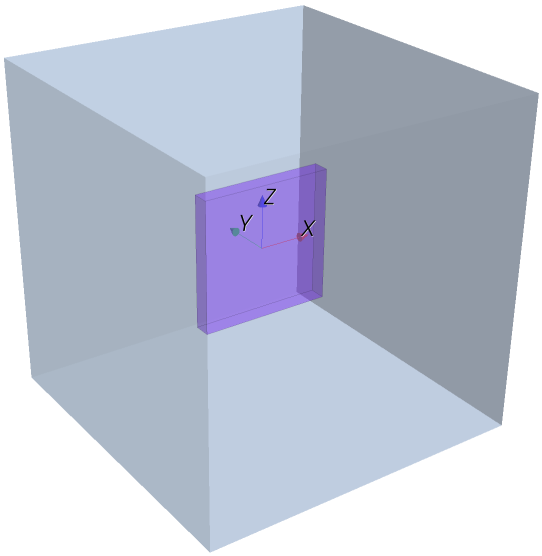
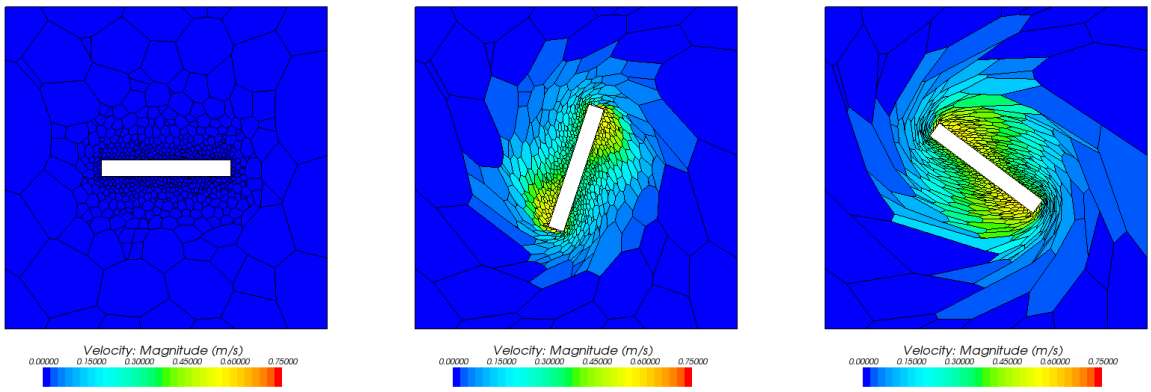


Figure 2.17: Mixer simulation geometry for demonstration of dynamic mesh methods.



(a) **(b)** **(c)**

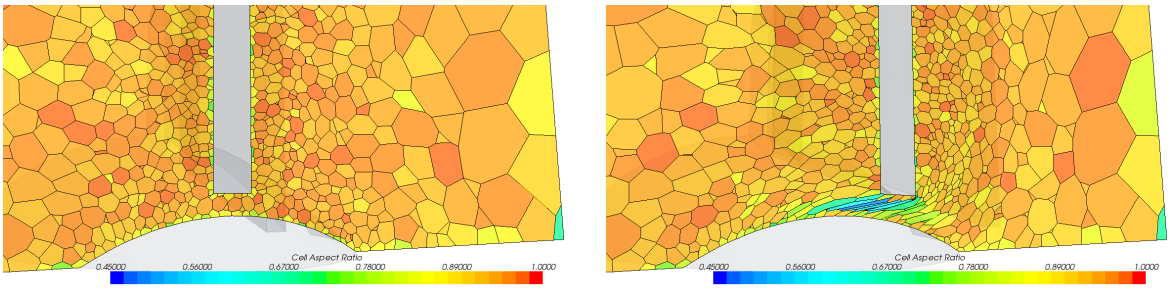
Figure 2.18: Progression of mixer simulation using mesh morphing method.

2.8.1.1 Mesh morphing/deformation method

Morphing meshes (a.k.a. deformable meshes) achieve two-body motion within a computational domain by altering (morphing) cell geometry. Figure 2.18 shows a series of images in which a morphing mesh approach is used to simulate the motion of the mixer simulation from Figure 2.17. The deformation of each finite volume cell is based on an interpolation function governed by the motion of a set of control points. For rigid-body motion such as this, the control points are generally set at mesh vertices on the surface of the moving boundary (the mixer blade in this case).

For the application of the GPAUV, and fluid-based vehicles in general, using a mesh morphing approach for the deflection of control surfaces proves to be somewhat impractical. This is due to the comparatively small gaps between bodies which must move relative to one and other. For this relative motion to occur, finite volume cells in gaps must stretch and deform, but can only stretch so far before their aspect ratios become unacceptably large. This problem is well illustrated by Figure 2.18c, where many cells in the domain have become extremely narrow.

This issue of limited cell deformation is in fact very restricting in the case of the GPAUV, as its control surfaces are designed to fit closely to the adjacent hull and fixed strakes. This tight-fitting geometry allows for maximum efficiency in the creation of maneuvering forces (see Section 3.6.1), but makes the use of a mesh morphing approach for the deflection of its control surface impractical. In preliminary simulations, control surface deflections of on the order of 5° resulted in unacceptably low mesh qualities. Figure 2.19 shows the application a morphing mesh to one of the GPAUV's control surfaces. A transverse plane that intersects the control surface and vehicle hull shows the finite volume mesh with a scalar to indicate each cells aspect ratio. Figure 2.19a shows the mesh before deflection of the control surface



(a) $\delta = 0°$ **(b)** $\delta \approx 3°$

Figure 2.19: Transverse cross section of morphing mesh during deflection of one the of the GPAUV's control surfaces with scalar showing finite volume cell aspect ratio.

($\delta = 0°$) and Figure 2.19b shows the mesh just before failure when the control surface is deflected to approximately 3°.

One option to alleviate the issue of poor quality meshes created over the course of deformation is iterative remeshing. This describes an approach in which a new finite volume mesh is created intermittently during the solution process. This approach can be used in combination with a mesh morphing approach to rebuild the mesh when cells have been deformed to their limits. Since building a new mesh and interpolating the solution onto that mesh can be computationally expensive, this approach is best suited for systems in which the geometry deforms from some initial condition to a final condition about which it oscillates. This is generally not the case for an AUV.

2.8.1.2 Embedded Mesh Methods

Embedded meshes involve one rigid mesh region able to slide within another rigid mesh region. A rigid interface between these regions allows for interpolation between the solution in each. The embedded mesh region must be axisymmetric about its center of rotation. These meshes are particularly useful for simulations involving rotation geometry, such as propellers. An embedded mesh approach for the mixer simulation, shown in originally Figure 2.17, uses a cylinder to enclose the mixer blade within its own rotation region (this configuration is depicted in Figure 2.20).

Unlike the mesh morphing approach, an embedded mesh approach does not deform finite volume cells within the domain, and is therefore not limited to a finite rotation. The mixer simulation can thus run indefinitely, as is shown in Figure 2.21.

While an embedded mesh approach works well in the mixer simulation, it is not feasible for the deflection of the GPAUV's control surfaces. As is shown in Figure 2.22, a properly devised interface, with its center located at the rudder's axis of rotation, must intersect the stationary strake and hull. Thus any rigid rotation of the embedded region would alter the geometry of the vehicle inappropriately.

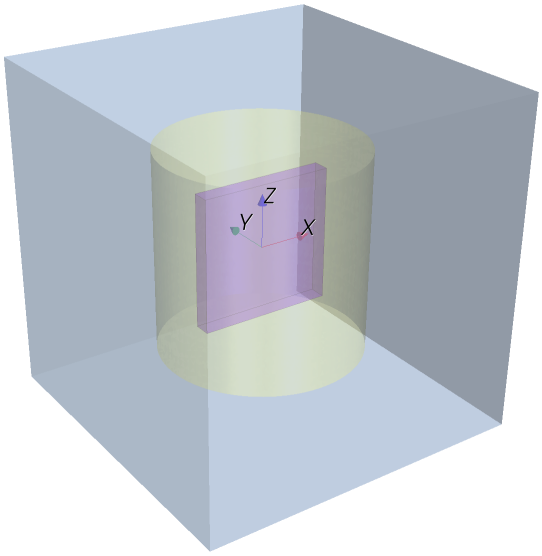
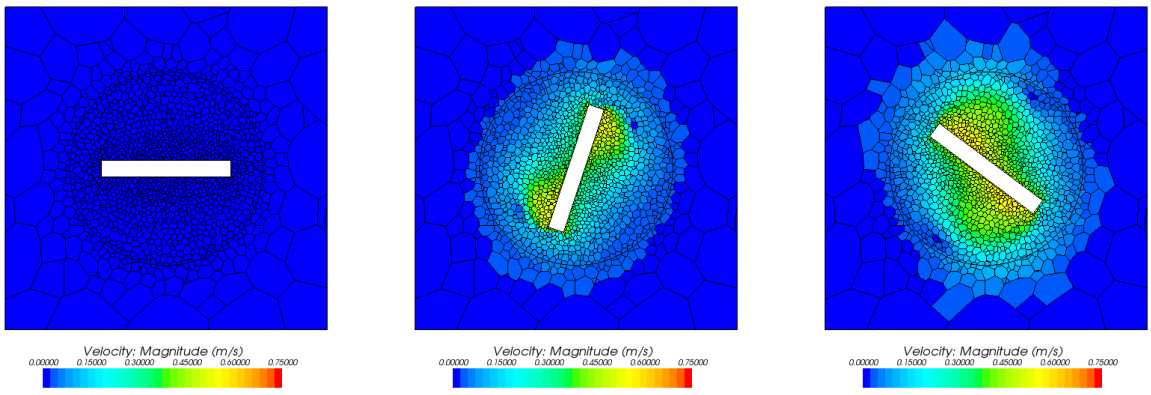


Figure 2.20: Geometry configuration for mixer simulation, with cylindrical interface shown in yellow.



(a) **(b)** **(c)**

Figure 2.21: Progression of mixer simulation using embedded mesh method.

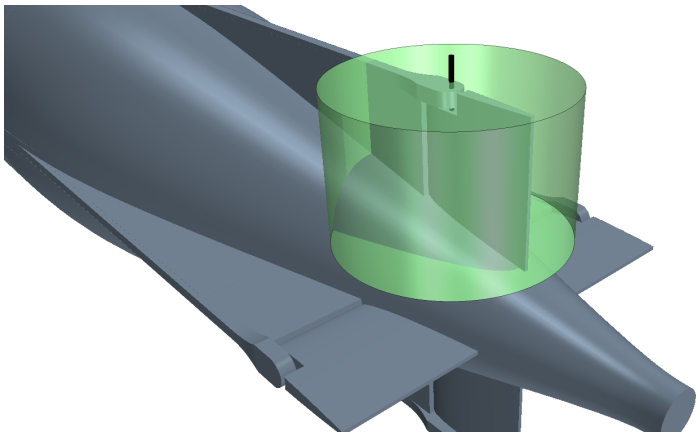


Figure 2.22: GPAUV with embedded mesh boundary (green) necessary for rotation of upper control surface about its axis (black).

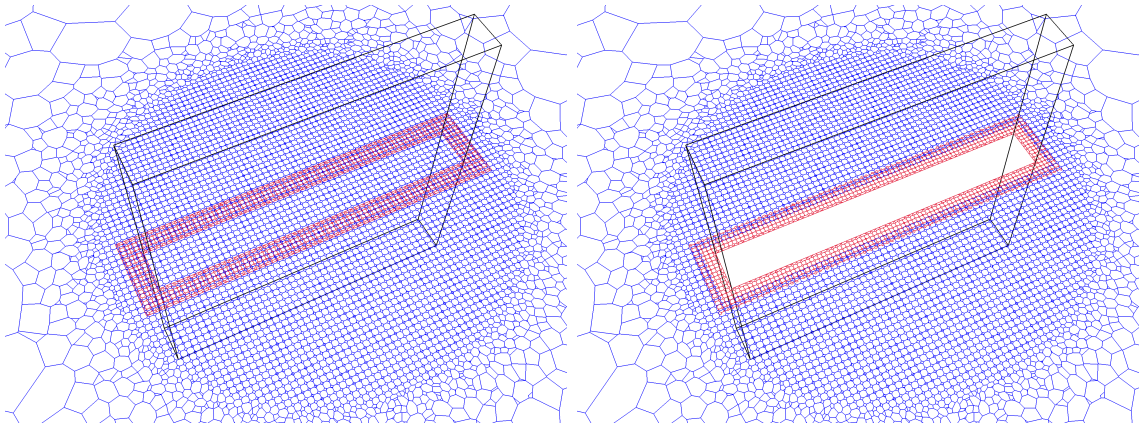
2.8.1.3 Overset Mesh Methods

Overset meshes, which are sometimes also referred to as overlapping or Chimera grids, are composed of a series of finite rigid volume meshes that overlap each other to form a single computational domain. Overset meshes have two major benefits: (1) they allow for a variety of structured meshes to be used in conjunction for better handling of complex geometries (e. g., an orthogonal mesh can be used with a cylindrical mesh) and (2) they allow for the relative motion of bodies in dynamic simulations. As the meshes employed within this study are all of an unstructured nature, the latter of these advantages is mainly of interest here.

An extended discussion of the overset mesh methodologies employed within STAR-CCM+ is presented by Hadzic [46]. In general, a background mesh, which comprises the majority of the computational domain, is supplemented by an overset mesh, which surrounds a moving body. The cells within a simulation employing an overset mesh can function as active, inactive or acceptors depending on their position within the computational domain. While the discretized governing equations are solved in active cells, no solution is computed within the inactive cells. Acceptor cells are located between active and inactive cells of a given region, and facilitate interpolation of the solution between the overset and background regions.

Figure 2.23 shows an overset mesh applied to the mixing paddle discussed in the previous sections. Here, the mixing paddle (black) is shown with a cross-sectional plane to display the finite volume meshes. The overset region, shown in red, forms a rectangular prism around the paddle; the background mesh, shown in blue, fills the remainder of the domain. Figure 2.23a shows the both complete meshes. Figure 2.23b shows the meshes used for computation of the solution, with inactive cells in the background domain excluded.

The process of removing inactive cells from the domain is referred to as “hole cutting.” In a dynamic simulation, each iteration must begin with a hole cutting procedure to exclude inac-



(a) Generated meshes **(b)** Computational domain

Figure 2.23: Mixer simulation with overset mesh, with overset (red) surrounding a mixing paddle (black outline) within a larger background region (blue).

tive cells from the computational domain. This is followed by the formulation of interpolation stencils for each acceptor cell.

As with any computational mesh, changes in cell size must occur gradually to avoid damping of flow gradients. This requires that the acceptor cells of the background and overset regions be of a similar size. This necessity is doubly important due to the need for acceptor cells to perform a robust interpolation of flow variables, using multiple nearby cells from the adjacent region.

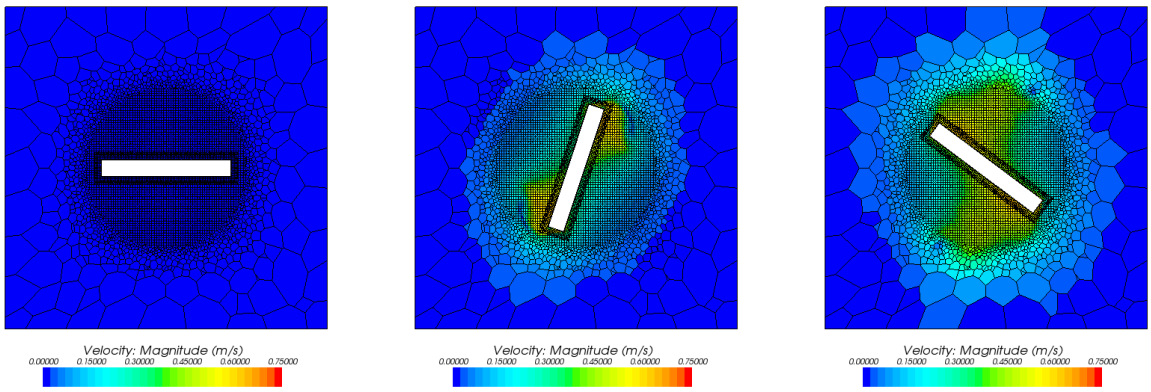
Figure 2.24 shows the progression of the mixer simulation using an overset mesh. Note that while the overset mesh can support the motion of the mixer paddle for an indefinite number of rotations, it is more computationally expensive than the embedded mesh.

2.8.2 Application of Overset Mesh to GPAUV

2.8.2.1 GPAUV Overset Mesh "Test-Rig"

In order to avoid computationally expensive full VFRM simulations, a GPAUV overset mesh "test-rig" simulation was used for development of mesh configuration and methodology [47]. These overset mesh test-rig simulations used the reduced domain shown in Figure 2.25. Only the upper control surface is modeled here to increase computational efficiency. The coordinate system visible in Figure 2.25 above the control surface is used to directly indicate its orientation. The domain contains a total 11.2×10^6 finite volume cells.[1]

[1]When considering the number of cells used to compute the fluid solution, the cell count is actually significantly less than this at any given time, as some of these cells are located outside of the computational domain and are therefore inactive. For the mesh used in VFRM simulations of the GPAUV, roughly 20% of total cells were inactive at any given point during a simulation.



(a) **(b)** **(c)**

Figure 2.24: Progression of mixer simulation using overset mesh method.

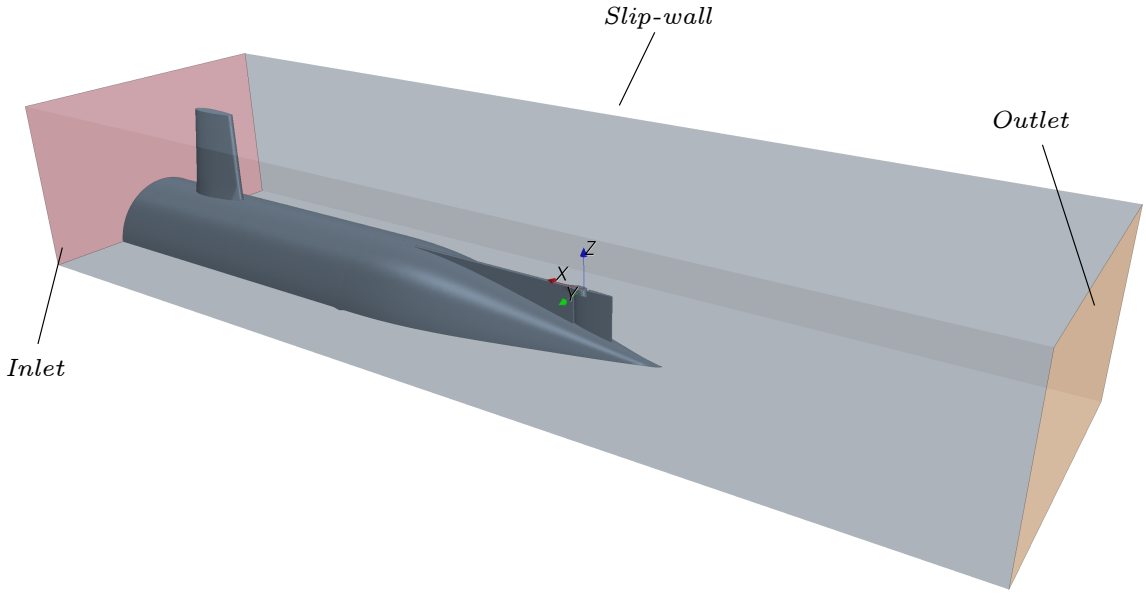
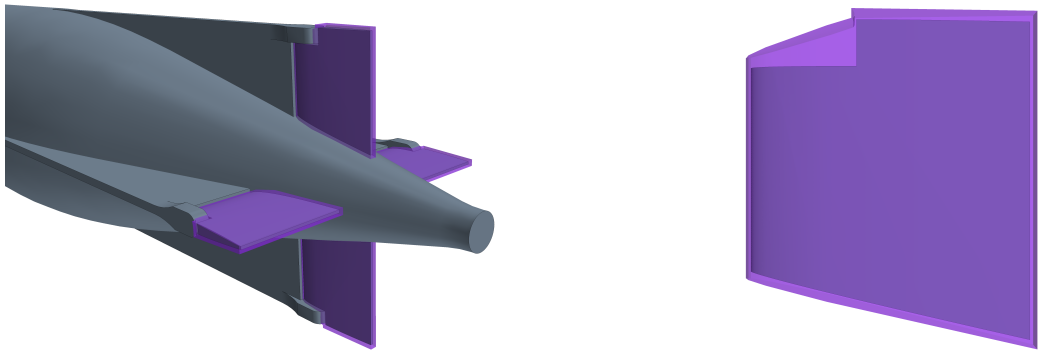


Figure 2.25: GPAUV overset mesh "test-rig" computational domain.

Testing of overset mesh configurations was accomplished by running simulations in which the control surface deflected back and forth throughout each possible position within its range. The overset mesh configuration was required to be capable of functioning at control surface orientations of $\delta = \pm 18°$.



(a) **(b)**

Figure 2.26: Control surface (grey) and its surrounding overset region (purple) with and without surrounding geometry.

2.8.2.2 Overset Region Geometry

The size and shape of an overset region must be set so as to entirely encompass all bodies that move with it. The overset region boundaries used for dynamic control surface simulations of the GPAUV are shown in Figure 2.26. As shown in Figure 2.26b, the overset region boundary surrounding each control surface was designed as a basic inflation of the control surface itself (these regions are shown with the rest of the GPAUV in Figure 2.26a). This geometry is large enough to allow for the boundary layer and flow separation to be captured mostly within the overset region, yet small enough to avoid unnecessary computational costs.

2.8.2.3 Small Gap Mesh Refinement

Small gaps between bodies that move relative to each other, like those between the GPAUV's control surfaces and fixed strakes as well as those between the control surfaces and hull, present a significant challenge in overset mesh implementation. To allow for interpolation between solutions in the background and overset regions of the simulation, these gaps must include multiple layers of cells (4 in the current version of STAR-CCM+) from both regions (overset and background). This requirement has been satisfied by local mesh refinement in these gaps. Figure 2.27 shows one of the GPAUV's control surfaces and the refinement volumes used to dictate local cell size. A cross section of the mesh in the gap between a fixed strake and control surface is shown in Figure 2.28. Here, it can be seen that a higher density of prism-layer cells has been used at the control surfaces's leading edge to insure robust interpolation.

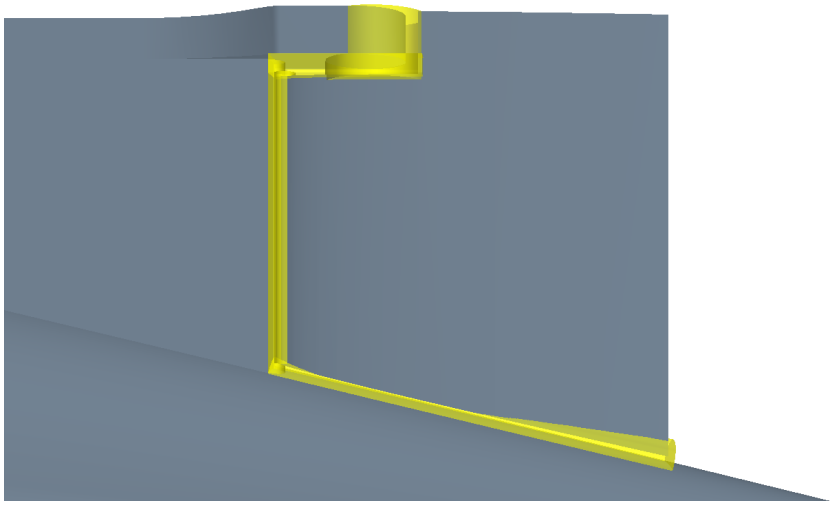


Figure 2.27: Mesh refinement volumes (yellow) used to increase mesh density in control surface gaps.

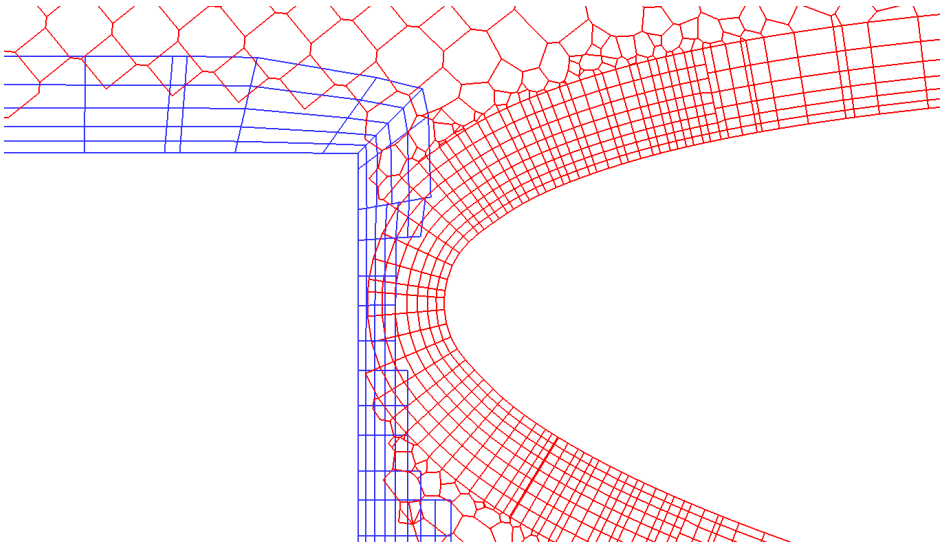


Figure 2.28: Overset (red) and background (blue) meshes in the gap between one of the GPAUV's fixed strake and moveable control surface.

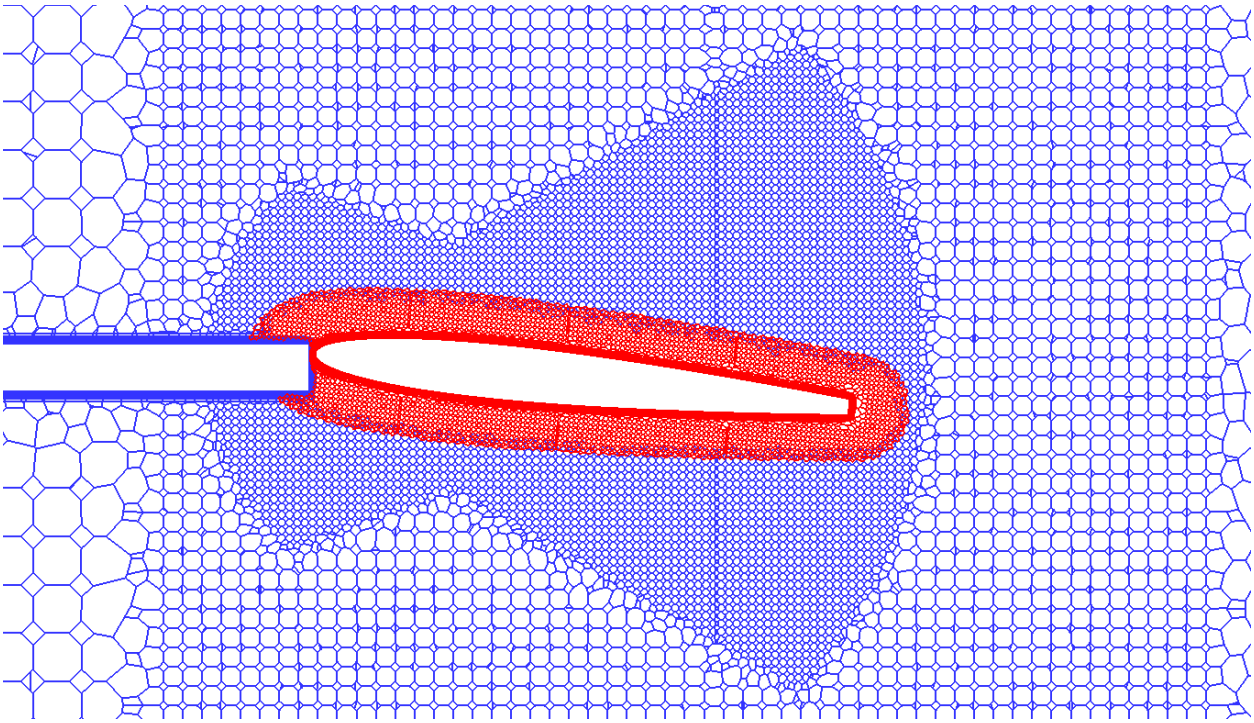


Figure 2.29: Overset (red) and background (blue) meshes with control surface slightly deflected. Local "sweep" refinement visible in background mesh.

2.8.2.4 Deflection Sweep Mesh Refinement

Interpolation between the overset and background region must be possible at the full range of positions that a control surface can occupy. Thus, mesh refinement must be designed to facilitate the deflection of the control surfaces. Special "sweep" refinement volumes, somewhat resembling the shape of a "snow-angel" when viewed on a plane normal to the control surface's axis of rotation (see Figure 2.29), were used to apply localized mesh refinement to the background mesh. This insures that the background region's acceptor cells are sufficiently fine for all possible deflections of the control surfaces ($-18^\circ \leq \delta \leq 18^\circ$). Figure 2.30 shows these refinement volumes with the GPAUV geometry.

2.8.2.5 Reduction of Possible Control Surface Deflections

The success of the overset mesh interfaces around the control surfaces of the GPAUV was found to be quite sensitive to mesh configuration. On the actual GPAUV, the control surfaces are capable of achieving a very large number of discrete positions within their operational limits. This presents a somewhat problematic situation when attempting to rigorously test

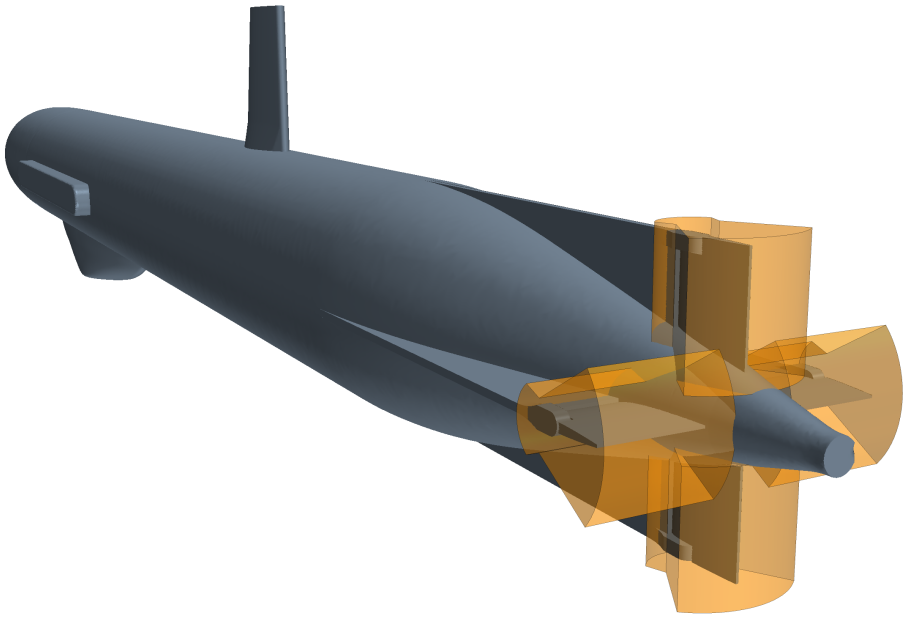


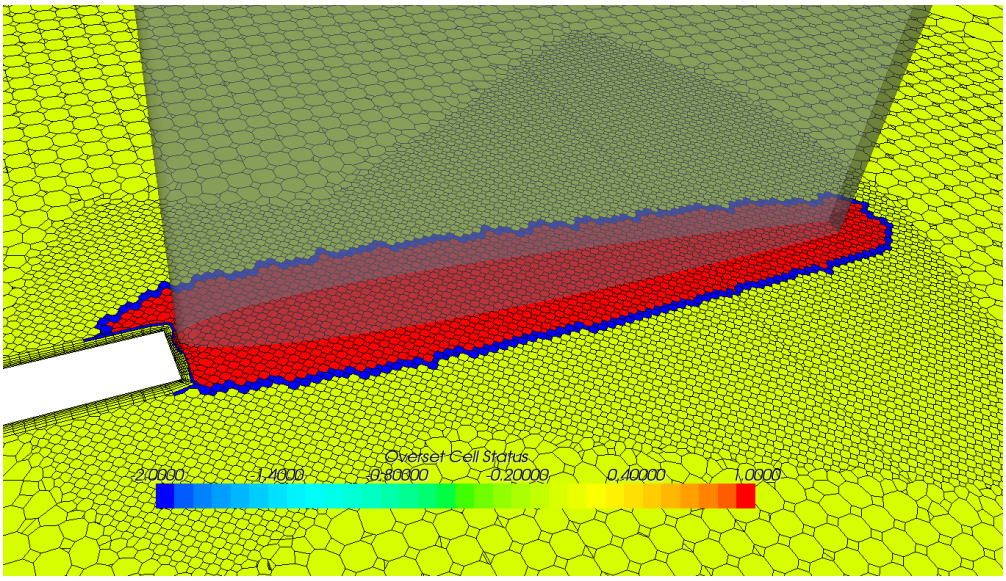
Figure 2.30: Control surface sweep spacial mesh refinement volumes (shown in orange).

the functionality of the overset mesh, as one cannot be sure that every possible deflection has been tested ahead of end-use in a maneuvering simulation.

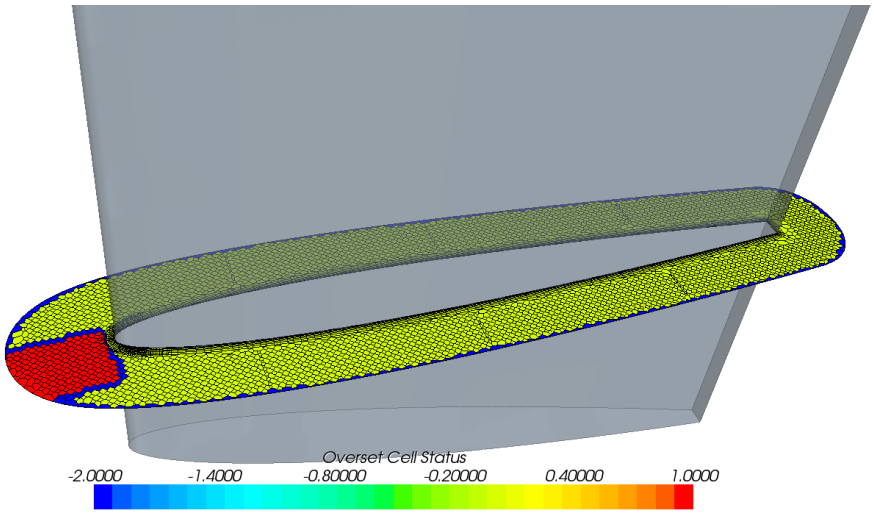
To provide a robust mesh configuration for end-use simulations, the positioning of control surfaces was limited to a smaller set of deflections withing their operating ranges. This was accomplished by rounding the control algorithm's control surface commands to the nearest 0.01 radians ($\sim 0.6°$). Likewise, the logic used to model the control surfaces' actuators includes restrictions to the rotational rates that can be employed to achieve these target deflections (in simulations with a time-step of $\Delta t = 0.05\,\mathrm{s}$, the rotational rate of flap deflections was rounded to the nearest 0.1 radians). The practice allows the functionality of the an overset mesh configuration (i. e., its ability to perform hole-cutting and solution interpolation tasks), to be tested rigorously *a-priori* of end-use simulations.

2.8.2.6 Visualization of Overset Mesh

Figure 2.31 shows the classification of cells in the background and overset regions near a control surface. In the background region (Figure 2.31a), cells within the control surfaces's overset region are inactive (red), while cells located away from control surface are active (yellow). These two types of cells are separated by blue acceptor cells. Figure 2.31b depicts the complementary image to Figure 2.31a, in which only cells from the overset region are shown. Comparing these two images, one can see that where one region's cells are active, the other's are inactive. Likewise, each region's acceptor cells are located in the same area (they



(a) Background region



(b) Overset region

Figure 2.31: Overset cell status in background and overset regions near control surface (inactive = red, active = yellow and acceptor = blue).

are actually partially overlapping). Note also in Figure 2.31b that there are inactive cells in the overset region near the leading edge of the control surface. These cells are currently located outside of the domain due to the presence of a fixed strake, and therefore it is not necessary to solve for the flow within them.

2.8.2.7 Comparison of Overset and Static Meshes

To validate the accuracy of the overset mesh configuration chosen for the GPAUV, a series of steady simulations were performed using both an overset and a traditional "static" mesh. To allow for a more direct comparison, the refinement settings of the static mesh were set to closely match those of the overset mesh. In each simulation, the GPAUV operates in head-on flow (with no side-slip) with its upper rudder deflected to a predetermined angle ($\delta = 0°, 8°, 16°, 24°, 32°$).[1]

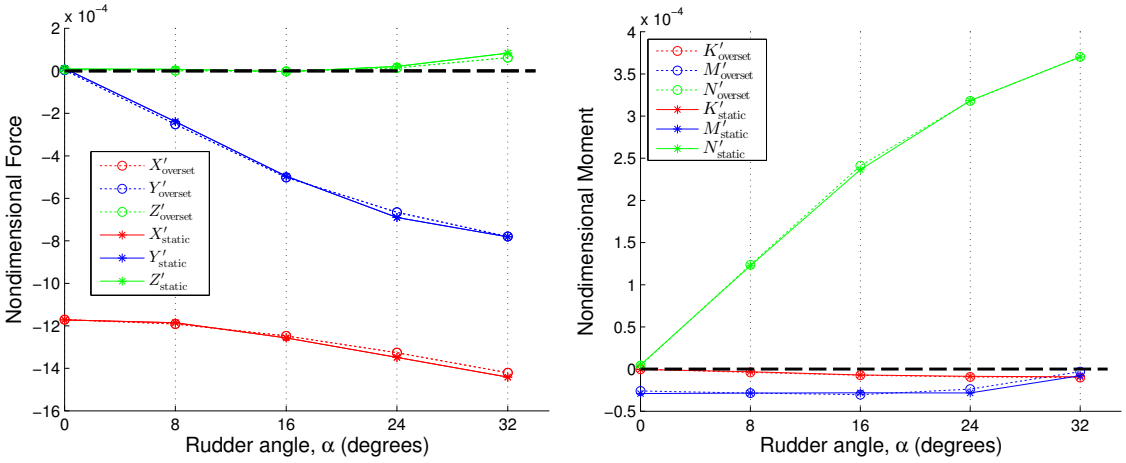
While the overset and static mesh simulations had fairly similar convergence behavior, a considerably larger number of iterations were required to converge the solution for control surface deflections experiencing flow separation ($\delta \leq 16°$). Simulations with lower control surface deflection angles generally converged within 500 iterations, while those with the higher deflection angles took as many 1250.

Results Figure 2.32 shows comparisons of the forces and moments acting on the GPAUV for the range of tested upper rudder deflections using both the overset and static meshes. The two sets of results show very good agreement. The largest discrepancies occur when the flow begins to separate from the deflected rudder ($\delta \cong 24°$), as the flow solution here is sensitive to small disparities between the two meshes. Subsequent consideration of this issue and discussion with other researchers has lead to the conclusion that, in addition the affect of interpolation at the interface between the background and overset regions, the discrepancies here are likely partially due the difference in directional alignment of the two meshes.

Figure 2.33 shows the results from the overset and static mesh simulations for $\delta = 16°$ and $\delta = 24°$. In each image, contours of the velocity magnitude are shown on a plane intersecting the half-span of the deflected rudder. The lower, pre-stall, angle of deflection simulations, shown in Figures 2.33a and b, are nearly identical, however the magnitude of differences in the higher rudder deflection simulations, Figures 2.33c and d, is substantial. The flow structure in this scenario is highly three-dimensional, and better visualized in Figure 2.34, which shows velocity magnitude on transverse planes in the overset and static mesh simulations for $\delta = 24°$.

Discussion The agreement between the overset and static mesh simulations presented in this section supports the usage of overset meshes to simulate the dynamic deflection of control surfaces in VFRM simulation. While some mesh dependence was evident in partially separated flow scenarios, the overall impact of these differences on the forces and moments experienced by the vehicle was relatively small. None-the-less, this event serves as a reminder of the potential for CFD solutions to be influenced by mesh resolution and configuration.

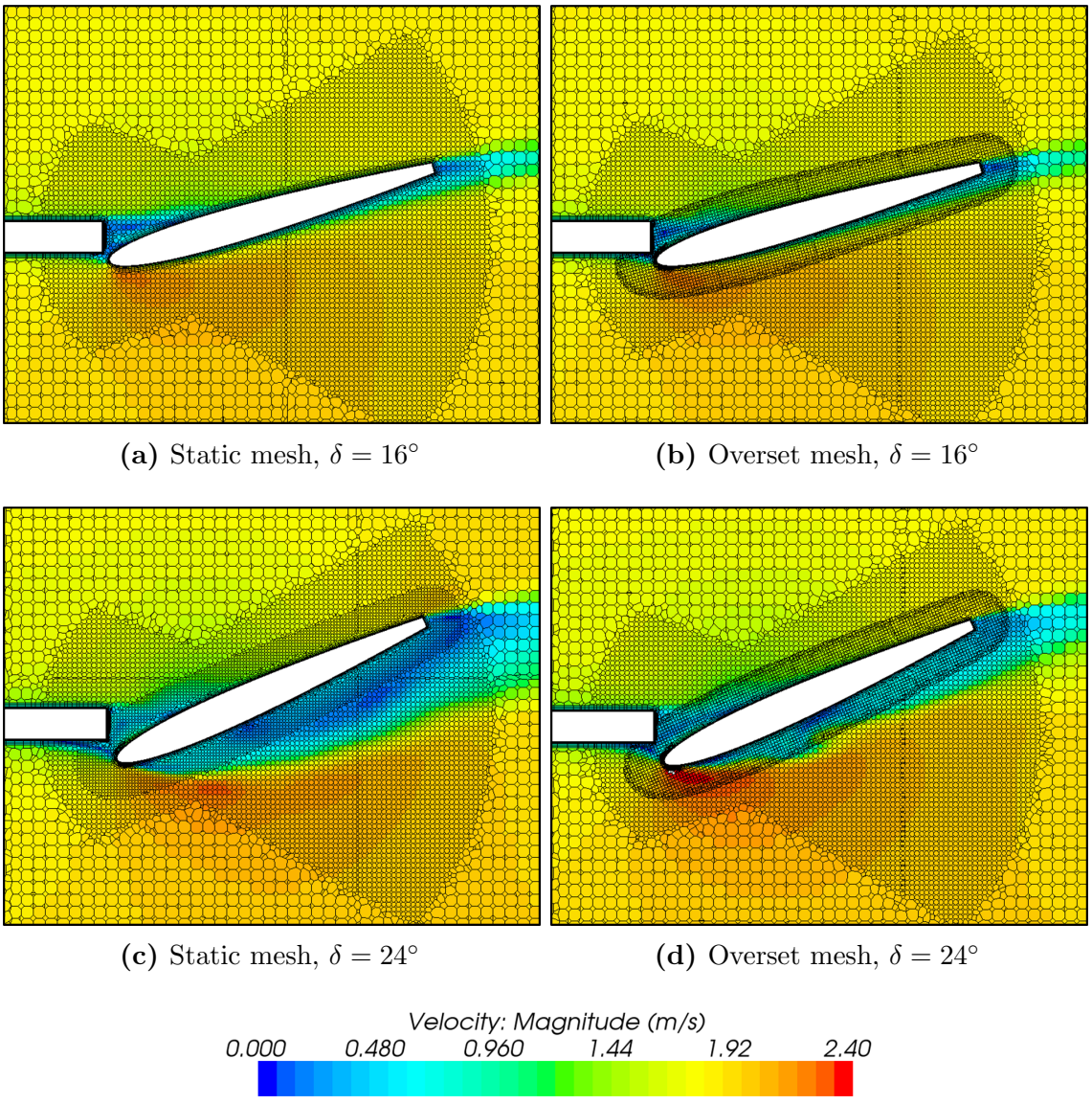
[1] This portion of the study was completed before later developments led the deflection limits of the control surfaces to be reduced from 32° to 18° due to physical limits of the GPAUV.



(a) Forces **(b)** Moments

Figure 2.32: Nondimensional reactions due to upper rudder deflection from overset (circle) and static (asterisk) simulations.

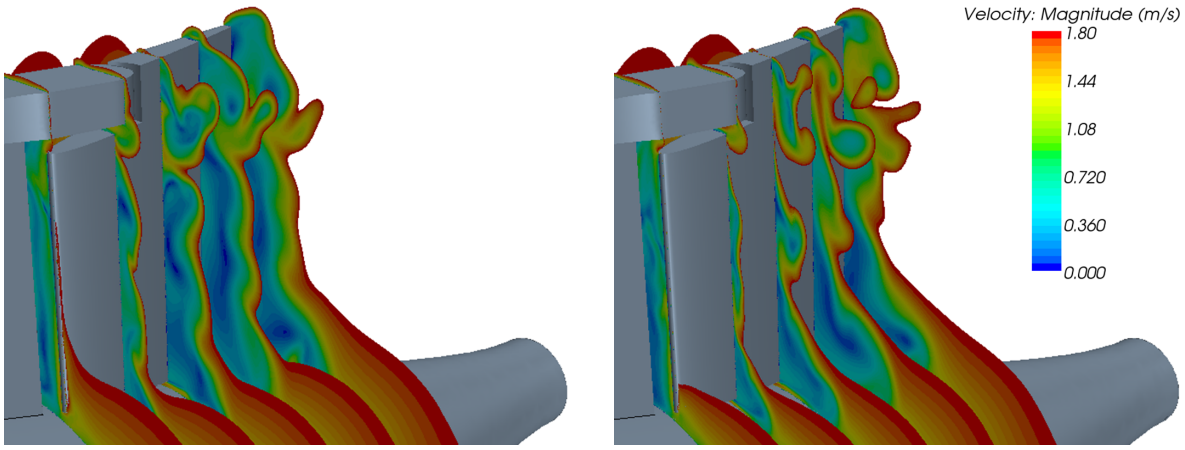
While it is not of particular importance when considering dynamic maneuvering simulations, some sets of static simulations, like those used to create the fin models described in Section 3.6.1, can reap a substantial time-savings from the use of overset meshes. Although five distinct meshes were required to run the five different control surface deflections with the static mesh approach, only one overset mesh is required. The different angles of deflection can be achieved by simply cutting a new hole in the background region based on the position of the overset region.



(a) Static mesh, $\delta = 16°$ **(b)** Overset mesh, $\delta = 16°$

(c) Static mesh, $\delta = 24°$ **(d)** Overset mesh, $\delta = 24°$

Figure 2.33: Velocity field at half-span of upper rudder in static and overset mesh simulations deflected to $\delta = 16°$ and $24°$.



(a) Static mesh **(b)** Overset mesh

Figure 2.34: Velocity magnitude field, shown on transverse planes, on suction side of rudder with $\delta = 24°$. (Legend shown in (b) applies to both sub-figures.)

2.9 Hydrostatic and Gravitational Effects

Hydrostatic effects can be applied within a CFD formulation to impart a buoyancy reaction on a rigid-body. This is accomplished by

$$p_{piezo} = -\rho g \left(z_0 - z \right), \tag{2.51}$$

where the convention of measuring z-positive downwards is maintained. Gravitational effects are generally applied directly to the rigid-body kinematic model.

A simplified simulation of the hydrostatic righting of a rectangular prism was used to verify the method used to apply a buoyancy-gravity wrench to a rigid-body vehicle. The body's center of gravity, CoG, is located at a vertical distance $0.1L$ below its center of buoyancy CoB (see Figure 2.35). The center of gravity is coincident with the body's center of volume, although the formulation used here allows for an arbitrary placement to accommodate partially flooded bodies.

Figure 2.36 shows the computational domain, with the rectangular prism located within a spherical boundary. The body was made free to rotate about its (horizontal) y-axis and given an initial velocity of 0.1 rad/s about that axis to avoid a condition of neutral stability.

Unsteady solutions were calculated on a simple tetrahedral mesh with $30{\times}10^3$ cells. The implicit solver used a time-step of 0.025 s.

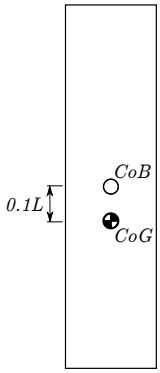


Figure 2.35: Hydrostatic stability test simulation centers of gravity and buoyancy.

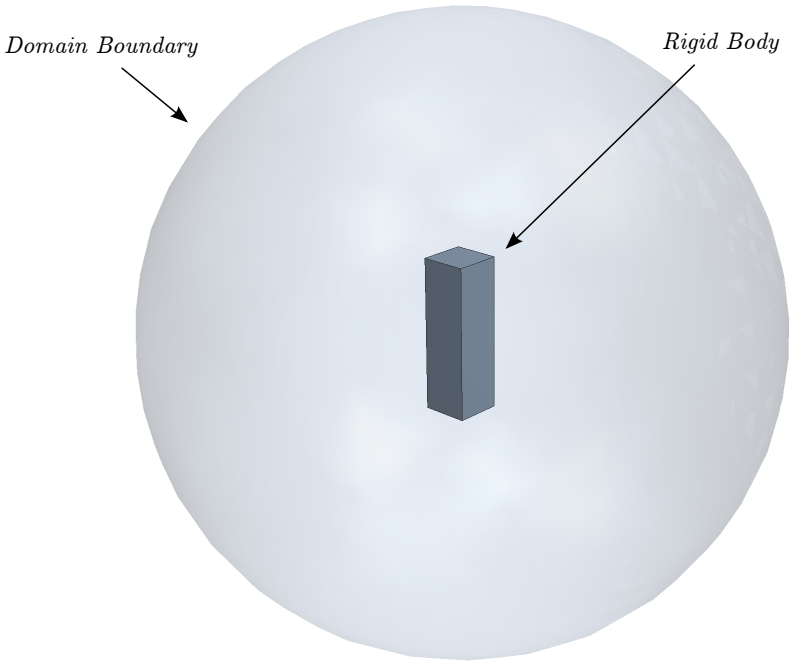


Figure 2.36: Domain used in simplified buoyancy-gravity wrench verification.

The rigid-body dynamics of the block are defined by

$$m = \forall \rho_{\text{block}} \tag{2.52}$$

$$I_y = \frac{m\left(a^2 + c^2\right)}{12}. \tag{2.53}$$

By allowing the body to be of a density equal to that of the water in which it is submerged, the system was made to be neutrally buoyant. As such, the forces acting on the body cancel out ($W = B = mg$), but as they act through different locations, their moments do not. Thus, the hydrostatic buoyancy-gravity balance reduces to[1]

$$\vec{g}(\eta) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_g \\ M_g \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (z_G W - z_B B)\cos(\theta)\sin(\phi) \\ (z_G W - z_B B)\sin(\theta) \\ 0 \end{bmatrix}. \tag{2.54}$$

Three simulations, using different approaches to model the effects buoyancy and gravity, were considered:

1. A simulation in which the gravity-buoyancy wrench is applied via STAR-CCM+'s internal methods. This approach is known from commercial testing to work well, but is restrictive in that the center of buoyancy must be located at a body's center of volume (not allowing for partially flooded spaces).

2. A simulation in which the gravity-buoyancy wrench is applied using an external macro. In this method, the 6-DoF wrench is computed at each time-step based on the body's orientation, then applied via an external force and moment at the center of gravity.

3. A simulation in which the gravity-buoyancy wrench is represented by two external forces: one for the body's gravitational force and one for its buoyancy force. These forces' locations are specified in a body-fixed frame, while their orientations are specified in an inertial frame.

The motion of the body is shown in Figure 2.37. As expected, the three histories are quite similar, although a small difference does exist between the external macro method and the other two. This is due to the explicit nature of the macro-based approach, in which the wrench is updated every time-step, whereas the other two implementations are implicit in that the wrench is updated at every iteration. For this reason, the body-fixed force approach was employed in this study.[2]

[1] The full hydrostatic buoyancy-gravity balance is presented in Chapter 3

[2] This approach to representing gravitational effects in a VFRM simulation lends itself to straightforward modeling of the inertial moving masses systems included in some AUVs [48].

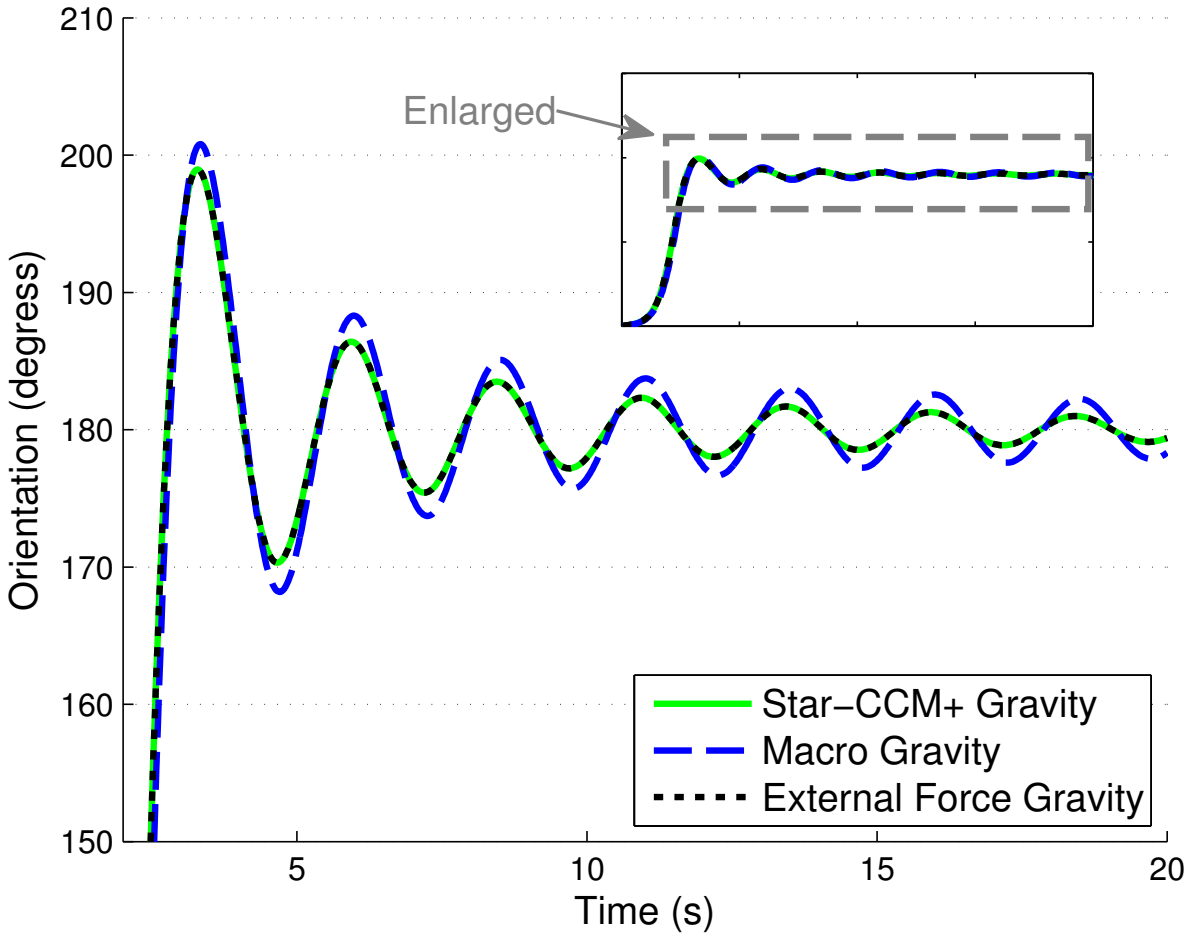


Figure 2.37: Body orientation during gravity-buoyancy wrench verification simulation using both STAR-CCM+ built-in method, external macro and body-fixed forces. Inset axes show entire motion history, with dashed gray box to indicate the portion displayed in on the larger axes.

2.10 CFD Methodology Validations

Three major validation analyses, using experimental data for comparison, have served to confirm the ability of CFD methods used within this study to reproduce real-world physics. While these simulations do not use the GPAUV geometry that is the main focus of this research project, they do focus on flows of a similar nature that have been the subject of many experimental and numerical studies. The discretization schemes, solution algorithms, mesh characteristics and turbulence closures used in these validation studies are the same or similar to those applied to the GPAUV.

2.10.1 DARPA SUBOFF

The SUBOFF geometry was developed by DARPA during the 1980s, primarily to serve as a validation case for numerical simulations of submarine hydrodynamics [49–51]. Tests were performed on a total of eight different configurations with varying appendage sets and locations. With this wealth of experimental data, along with various numerically-based studies, the SUBOFF geometry provides a well-studied reference point for validation of CFD methods in a flow regime similar to that of the GPAUV. The "bare" configuration of the SUBOFF, which has no appendages other than the sail is shown in Figure 2.38, was used during this study.

2.10.1.1 Mesh Independence Analysis

To evaluate the performance of mesh configurations similar to those applied to the GPAUV, a mesh independence analysis was completed using the SUBOFF geometry.

Methodology A sampling of the meshes employed to study mesh independence, designated h_1 through h_5 from finest to coarsest, are detailed in Table 2.2. These meshes are related by a

Figure 2.38: DARPA SUBOFF "bare" geometry.

Table 2.2: SUBOFF mesh independence analysis simulation details.

Property	h_1	h_3	h_5
Cell count	18.5×10^6	3.2×10^6	0.64×10^6
Prism layer thickness $\left(\frac{t}{L}\right)$		0.005	
Wall prism layer thickness $\left(\frac{t}{L}\right)$		3.4×10^{-6}	
Number of prism layers		24	

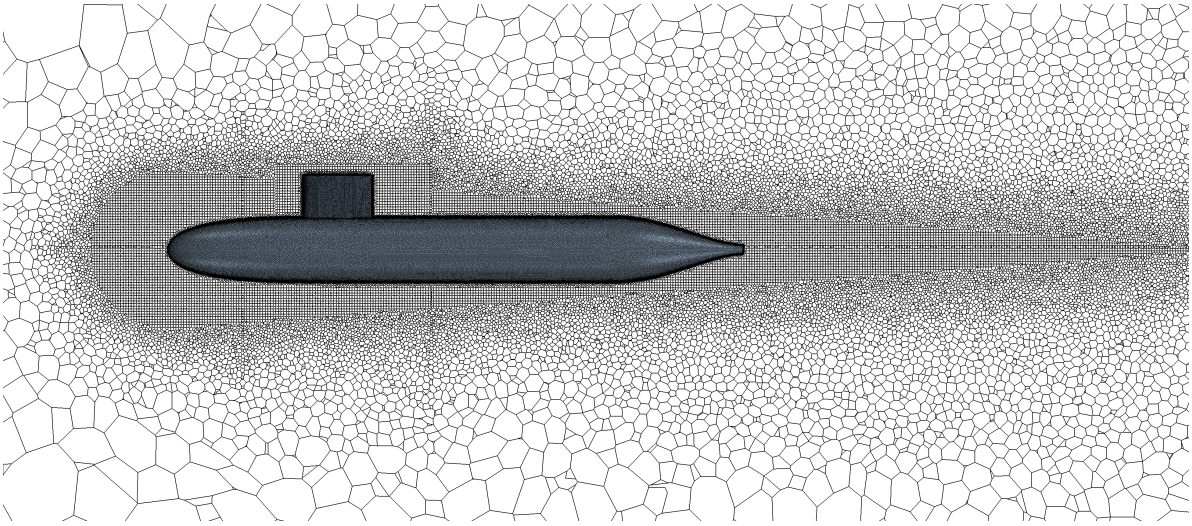


Figure 2.39: Cross section of SUBOFF h_3 mesh.

refinement ratio of $\sqrt{2}$, excluding the body-fitted prism layer mesh, which remains constant.[1] Figure 2.39 shows a cross section of the h_3 mesh.

Results The SUBOFF's axial drag coefficient, C_d, at a length-based Reynolds number of 1.76×10^7 was considered in this analysis.

$$C_d = \frac{-X}{\frac{1}{2}\rho A U^2} \tag{2.55}$$

Here, X, A and U are the axial force (positive forward), frontal area and forward speed respectively. Figure 2.40 shows the drag coefficient predictions from simulations using the

[1] Applying the same scaling to prism layer cells as the remainder of the domain tends to create overly large or small cells unsuited to capturing boundary layer flow.

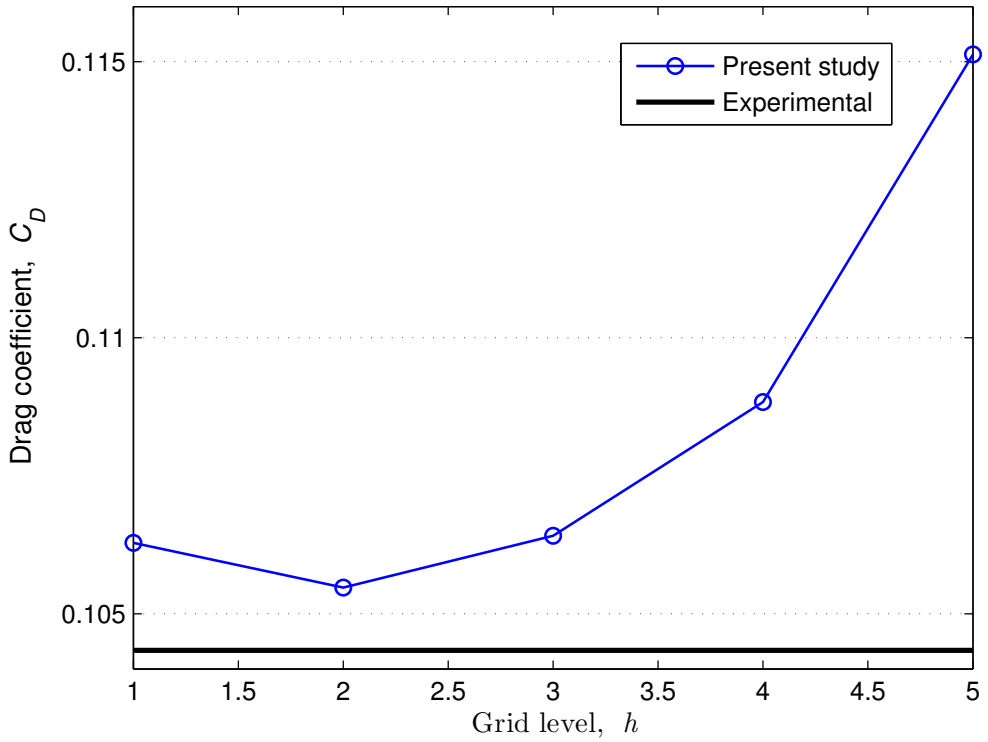


Figure 2.40: SUBOFF drag coefficient prediction with a range of grid spacings at $Re = 1.76 \times 10^7$ with experimental data from Crook [52].

range of grids shown in Table 2.2. The result from an experimental study is shown as well [52].

The results shown in Figure 2.40 can be used to assess the spatial convergence of the solutions following the Richardson extrapolation (RE) method summarized by Roache [53]. For a series of grids, $h_i = 1, 2, 3 \ldots$, related by refinement ratios, $r = \frac{h_i}{h_j}$,[1] with solutions, $\phi_i = 1, 2, 3 \ldots$, and fractional differences between those solutions, $\epsilon_{ji} = \frac{\phi_j - \phi_i}{\phi_i}$, the observed order of accuracy can be approximated by

$$p_{ijk} = \frac{\ln\left(\frac{\phi_k - \phi_j}{\phi_j - \phi_i}\right)}{\ln(r)} = \frac{\ln\left(\frac{\epsilon_{kj}}{\epsilon_{ji}}\right)}{\ln(r)}. \tag{2.56}$$

Applying (2.56) to the results shown in Figure 2.40 gives an order of accuracy of $p = 2.7$ for all grid combinations, except when considering the three finest grids (h_1, h_2, h_3), which appear to exhibit an oscillatory convergence behavior.

[1] A single refinement ratio was employed for this study, although, this is not a requirement of the RE method.

Roache presents grid convergence index (GCI) as a means of consistently reporting grid convergence in a way that is independent of refinement ratio. If a factor of safety, F_s, is selected, the grid convergence index on the finer grid of a set of solutions can be computed as

$$GCI_{ji}^{\text{fine}} = F_s \left| \epsilon_{ji} \right| . \tag{2.57}$$

Applying a factor of safety of $F_s = 3$ (as suggested by Roache) gives $GCI_{32}^{\text{fine}} = 0.007$ and $GCI_{43}^{\text{fine}} = 0.0178$. These are somewhat analogous to statistical error bounds (with $3\times$ the error based on $F_s = 3$).

The asymptotic convergence of the solutions can be verified by the following relation [54].

$$GCI_{43} = r^p GCI_{32} \tag{2.58}$$

Inserting the values from the SUBOFF analysis gives

$$\frac{r^p GCI_{32}^{\text{fine}}}{GCI_{43}^{\text{fine}}} = 1.0193, \tag{2.59}$$

where the value very near unity implies the solutions have asymptotic convergence.

2.10.1.2 Turbulence Closure Model Analysis

With a number of turbulence closure models (TCMs) available to complete the RANS system of equations,[1] a series of simulations were conducted to analyze their relative performance in the flow regime considered for this study.

Methodology A mesh similar to those considered in Section 2.10.1.1 (nearest to h_4) was employed to assess the effect of a range of TCMs on the simulation's solution. The details of this mesh are given in Table 2.3. Three TCMs were considered: k-ω SST, Spalart-Allmaras (SA), and k-ϵ [36]. While a large data set, with a wide array of experimental measurements and conditions exists for the SUBOFF geometry, this analysis was limited to straight-ahead axial drag. Iterative solution convergence of the steady solver was judged based on the asymptotic behavior of the simulation's drag prediction (iteration was discontinued when the drag maintained a value within 0.025 N for more than 10 iterations).

Results Figure 2.41 shows a series of drag coefficient estimates from simulations compared to experimental data from Crook [52]. A confidence interval of 1% was reported in the

Table 2.3: SUBOFF turbulence closure model analysis simulation details.

Parameter	Values
Cell count	1.25×10^{6}
Prism layer thickness $\left(\frac{t}{L}\right)$	0.005
Wall prism layer thickness $\left(\frac{t}{L}\right)$	3.4×10^{-6}
Number of prism layers	24
Average wall y^{+}	0.45

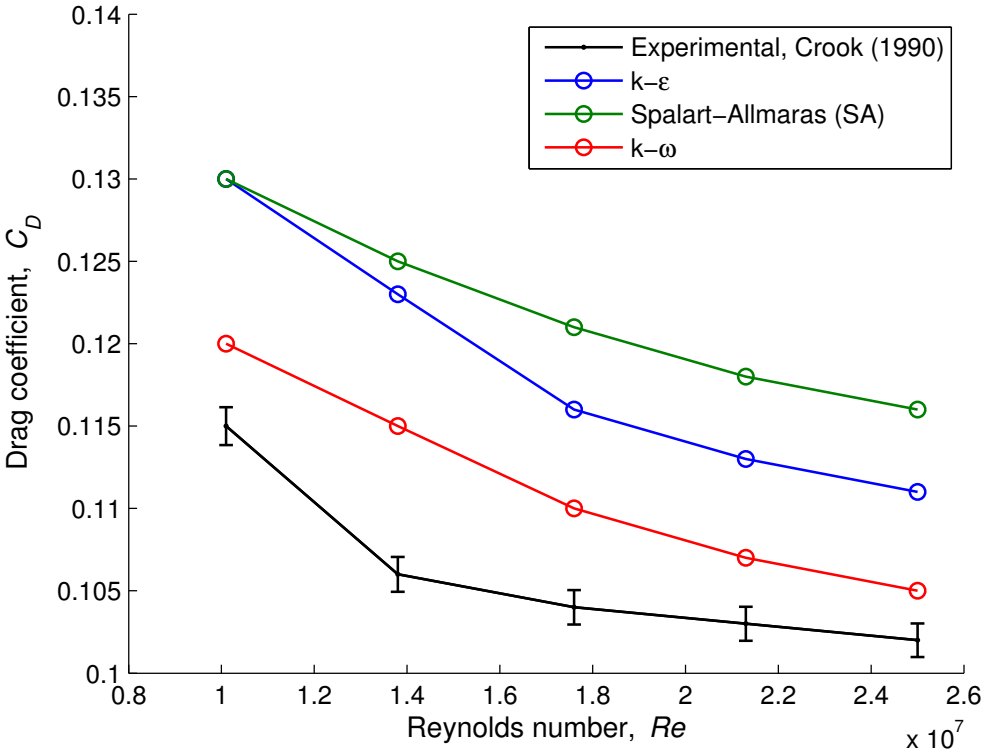


Figure 2.41: SUBOFF drag coefficient results from CFD with experimental data for comparison from Crook [52].

experimental testing and is plotted with the data. Each marker represents the results obtained from simulations using a specific turbulence model.

In a mostly attached flow, such as this, shear drag due to viscosity represents a large portion of the total drag. Accurate calculation of the shear drag component relies on the resolution of flow boundary layers and proper turbulence modeling. This was clearly shown by the poor

[1]See Section 2.2.2 for more on the closure of RANS formulations.

performance of a simulation not included in Figure 2.41, in which the k-ω model was used on a mesh without body-fitted prism layer cells. This simulation returned drag errors on the order of 20%.

With flow gradients properly resolved by appropriately sized prism layers, the k-ω model gave the best results for overall drag on this geometry. The SA and k-ϵ models returned slightly larger errors from the experimental measurements. The k-ϵ model performed significantly better than expected, as it is generally considered mostly suited to internal flows. Its proficient performance here is believed to be due to the highly streamlined shape of the SUBOFF geometry, which results in little flow separation.

2.10.2 Prolate Spheroid

A prolate spheroid is often used as an analytically-representative geometry for underwater vehicles (as well as missiles and airships). A large number of experimental fluid dynamic studies have examined prolate spheroids of various slenderness ratios, generally ranging from 5:1 to 8:1 [55–65]. These studies have analyzed both static and dynamic flow characteristics.

2.10.2.1 Cross-Flow Separation Flow Structure

Although a spheroid is a rather simple shape, the three-dimensional nature of the cross-flow separation that occurs on its leeward side has often been noted to be somewhat of an acid-test for numerical turbulence modeling. Unlike other canonical fluid flow geometries, such as a backward-facing step, an inclined prolate spheroid has no blatant flow separation point. The flow is also highly three-dimensional in nature (unlike a that over a cylinder). At nontrivial angles attack, flow separating off the leeward side of the spheroid forms a coherent vortex, creating a region of reattached flow. This flow structure is illustrated in Figure 2.42.

A number of researchers have shown fair-to-good agreement of numerical simulations of prolate spheroid flows with experimental data using laminar [66], RANS [67–71], DES [70, 72, 73] and LES [74–76] formulations. In general, these numerical simulations do a good job capturing the general nature of the flow field, but are often unable to accurately predict flow separation and reattachment locations.

2.10.2.2 Steady Flow Analysis

Based the amount of recent experimental and numerical studies available for comparison, $Re = 4.2 \times 10^6$ flow over a prolate spheroid with a 6:1 length-to-diameter ratio was selected for validation purposes.[1] Simulations with the spheroid at a 20° angle to the incident flow were considered for this study. This angle of flow has been shown experimentally to provide a fully developed cross-flow separation pattern as shown in Figure 2.42.

[1] If external appendages are neglected, the GPAUV has a length to diameter ratio of approximately 11:1.

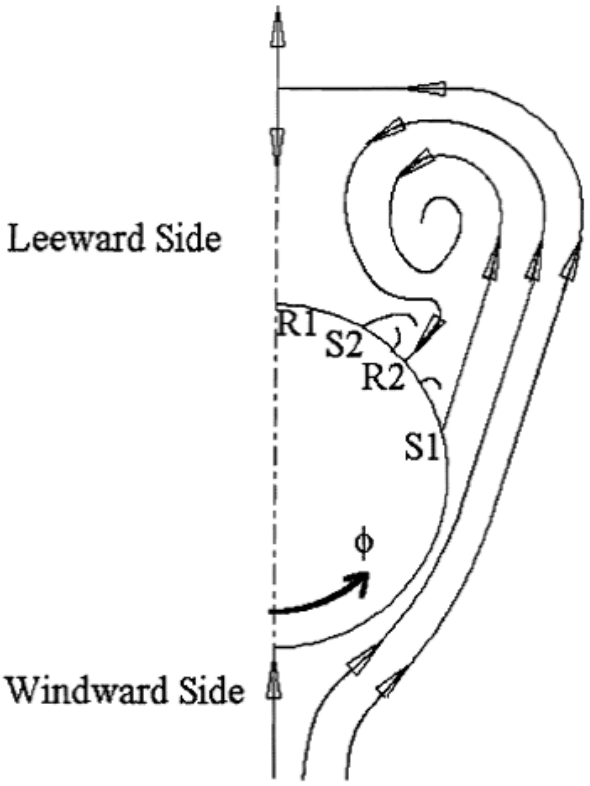


Figure 2.42: Flow diagram for a prolate spheroid at incidence showing: primary separation and reattachment (S1 and R1), secondary separation and reattachment (S2 R2) with projections of streamlines to a transverse plane [60].

Table 2.4: Prolate spheroid at incidence simulation details.

Property	Value
Number of finite volume cells in domain	2.80×10^6
Surface cell size ($\frac{\ell}{L}$)	0.0025
Prism layer thickness ($\frac{t}{L}$)	0.0125
Number of prism layers	36
Wall prism layer thickness ($\frac{t}{L}$)	5.00×10^5

Methodology A domain angled with the flow, as shown in Figure 2.43, was employed for these simulations. The details of the mesh used for solution iteration are given in Table 2.4. Based on the results of other studies [71, 77] as well as the results of the turbulence closure model study presented in Section 2.10.1.2, these simulations used a k-ω SST turbulence model. Iterative convergence of simulations was judged by monitoring lift and pitching moment coefficients on the spheroid.

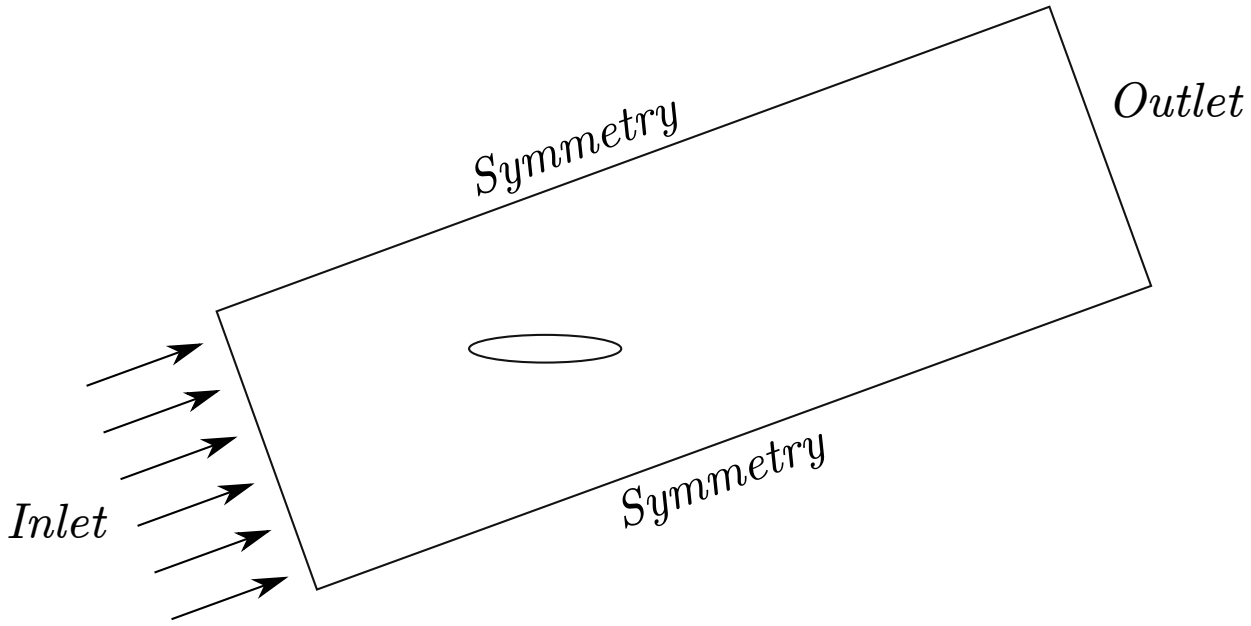


Figure 2.43: Computational domain for spheroid at incidence simulations ($\alpha = 20°$).

Local skin friction coefficient, C_f, on the surface of the spheroid serves as a good means of observing flow separation, and is the main metric used for comparison in this analysis.

$$C_f = \frac{\tau_w}{\frac{1}{2}\rho U_\infty^2} \tag{2.60}$$

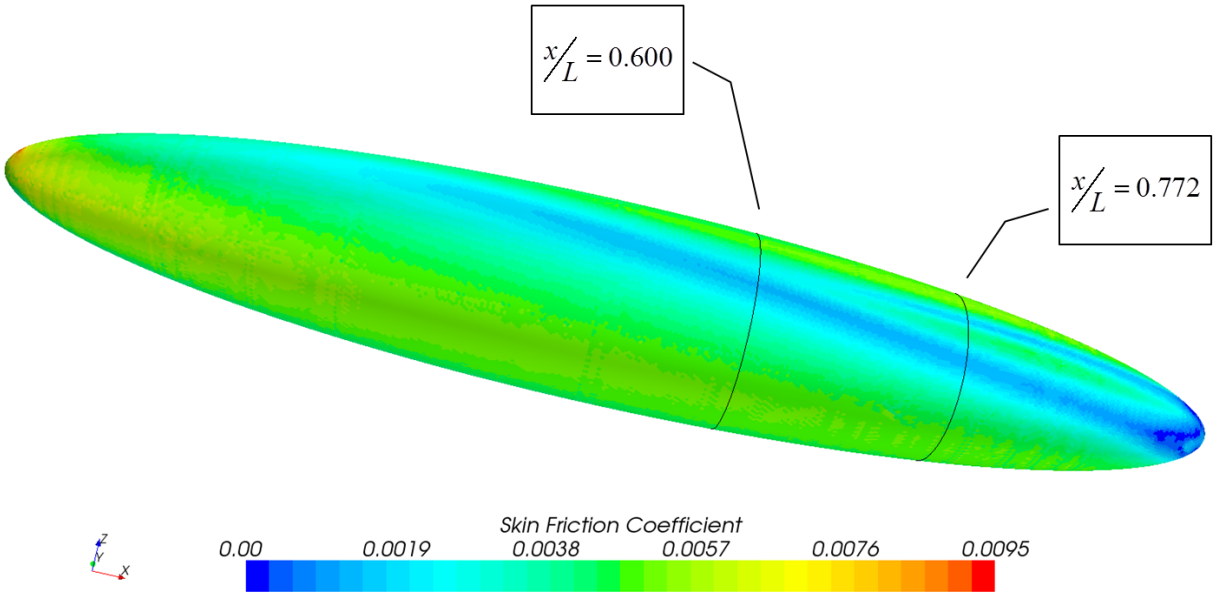
The skin friction, τ_w, is directly proportional to the gradient of the flow velocity near the surface.

$$\tau_w = \mu \left(\frac{\partial u}{\partial y} \right)_{y=0} \tag{2.61}$$

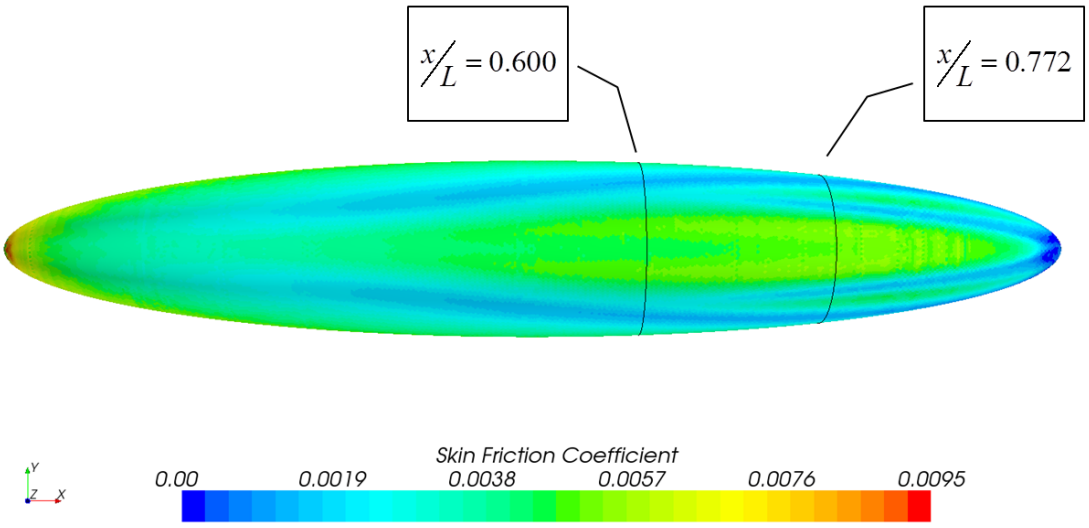
Flow separation is indicated by a local minimum in skin friction coefficient.

Results Figure 2.44 shows the skin friction coefficient scalar on the surface of the prolate spheroid at an angle of incidence of 20°. These images help to convey the complex nature of this flow, as the separation and reattachment features diagrammed in Figure 2.42 are visible.

Results from CFD simulations completed for this study are plotted in Figure 2.45 along with measurements from an experimental study [61] and a similar RANS study [78]. The azimuthal skin friction distribution on the spheroid is shown at two axial locations. The longitudinal locations of these data are shown in Figure 2.44. The coordinate system used to define azimuthal position, ϕ, is the same as that depicted in Figure 2.42, with $\phi = 0°$



(a) Side view



(b) Top view

Figure 2.44: 6:1 prolate spheroid at 20° incidence, $Re = 4.2 \times 10^6$, with skin friction coefficient scalar. Lines at $\frac{x}{L} = 0.600$ and $\frac{x}{L} = 0.772$ indicate locations of comparison to experimental measurements.

on the windward side of the spheroid. A 4% uncertainty was reported by Chesnakas and Simpson [61] for the experimental skin friction measurements, which were collected using a laser Doppler velocimeter (LDV) located within the spheroid (error bars shown in Figure 2.45 reflect this uncertainty).

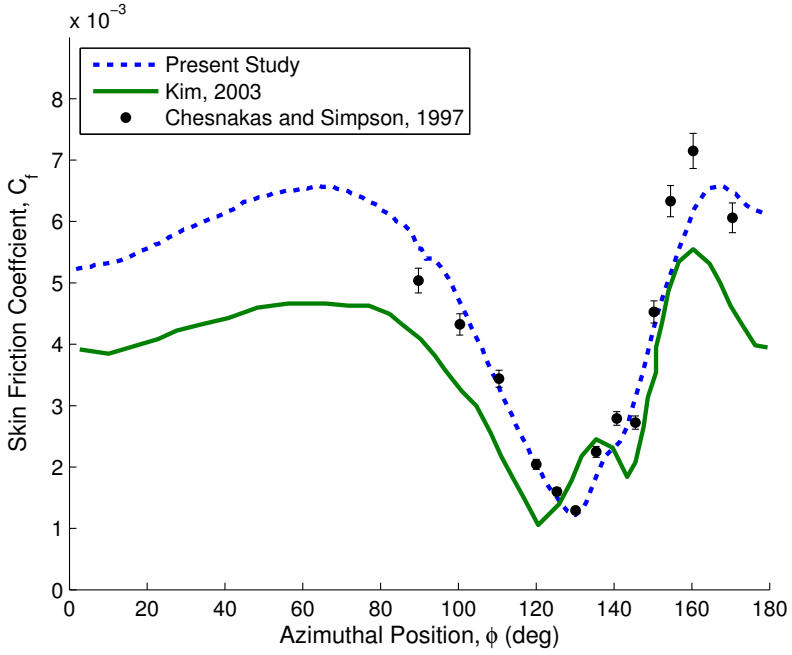
While the magnitudes of the skin friction values have noticeable errors, simulations from this study show very good agreement with the experimental data on the locations of flow separation. The more forward set of values ($\frac{x}{L} = 0.600$, shown in Figure 2.45a) shows slightly more error than those at $\frac{x}{L} = 0.772$ (Figure 2.45b). In particular, the simulation does not capture the inner flow separation at this forward location (labeled S2 in Figure 2.42). This may be due to an overly large dissipation of vortical momentum. As can be seen in Figure 2.46, the vortical structure is not as fully formed at the $\frac{x}{L} = 0.600$ location as the $\frac{x}{L} = 0.772$ location. This result is also evident in skin friction coefficient scalar shown in Figure 2.44.

Figure 2.47 shows a similar comparison as that shown in Figure 2.45, to a second set of experimental data collected by Wetzel [60]. These experimental measurements used constant current and constant temperature hot-film anemometers. Agreement between the CFD prediction and anemometer measurements is not as good as the with the LDV measurements shown in Figure 2.45. Wetzel reported a calibration uncertainty in their skin friction measurements (using both anemometer types) on the order of 20%. This implies the possibility of a 20% vertical shift in the experimental data presented in Figure 2.47 (this shift maintains the shape of the curve, but changes the average by 20%). The random uncertainty for these experiments was estimated at 5%, meaning errors in the shape of the experimental curves are relatively small. Based on this, the CFD simulation's ability to closely match the separation and reattachment locations of the experimental data while having a substantial error in magnitude is less concerning.

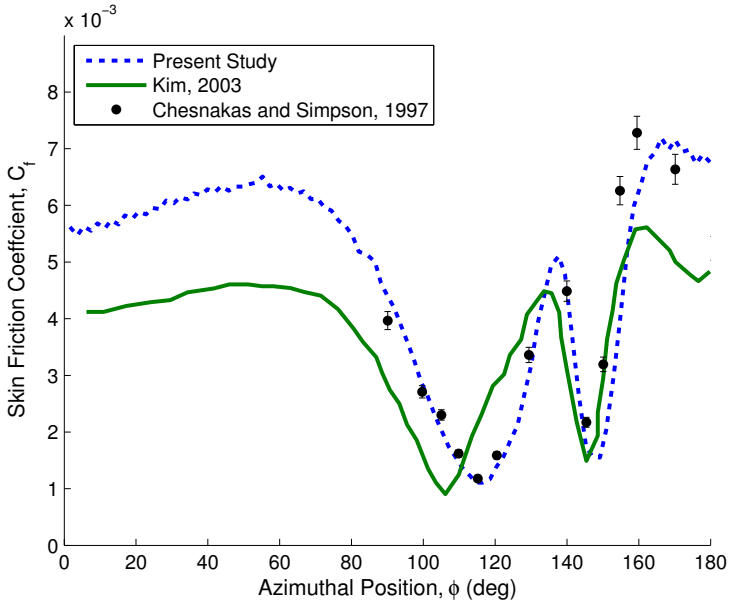
2.10.2.3 Unsteady Motion

Much of the impetus for this body of research is based on the difference between static and unsteady fluid dynamics. This issue was well demonstrated in a series of experiments analyzing the motion of a prolate spheroid [58–60, 62, 63]. Measurements taken during unsteady maneuvers of a prolate spheroid were compared to those taken at static angles of attack to show the importance of unsteady fluid dynamic phenomena. As the prediction of unsteady fluid dynamics is central to the success of a VFRM simulation, an assessment of the ability of the methods used within this study to predict unsteady flow was performed.

Spheroid Motion Wetzel and Simpson completed dynamic tests on the 6:1 prolate spheroid in a wind tunnel operating at a Reynolds number of 4.2×10^6 [60, 79]. These tests used a specially-designed sting referred to as the dynamic-plunge-pitch-roll (DyPPiR) to move the spheroid through a number of maneuvers. The time-dependent orientation of the spheroid



(a) $\frac{x}{L} = 0.600$



(b) $\frac{x}{L} = 0.772$

Figure 2.45: Skin friction (C_f) distribution on 6:1 prolate spheroid, with $Re = 4.2 \times 10^6$ flow at $\alpha = 20°$. CFD results shown with experimental data [61] and data from another RANS-based study [78].

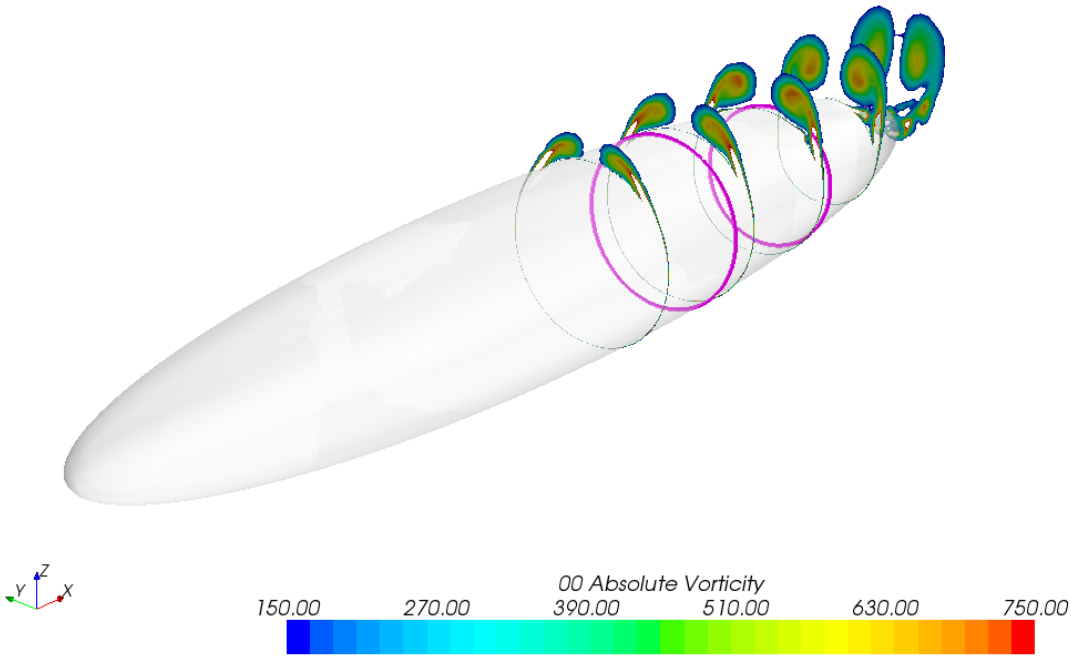


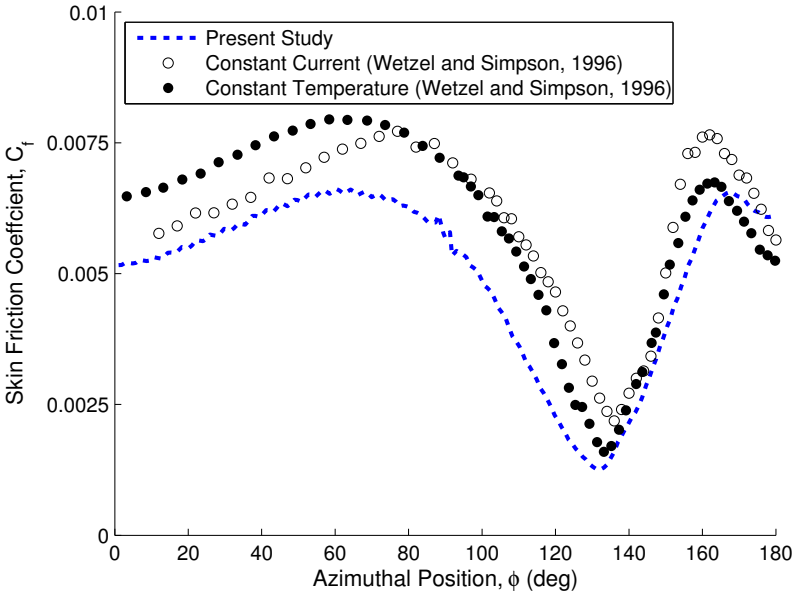
Figure 2.46: Isometric view of 6:1 prolate spheroid at 20° incidence with absolute vorticity (in the x-direction) scalar sections. Purple lines at $\frac{x}{L} = 0.600$ and $\frac{x}{L} = 0.772$ indicate locations of comparison to other studies.

in this experimental setup was found to have an uncertainty of ±2°. Measurements were collected using the anemometers described in Section 2.10.2.2.

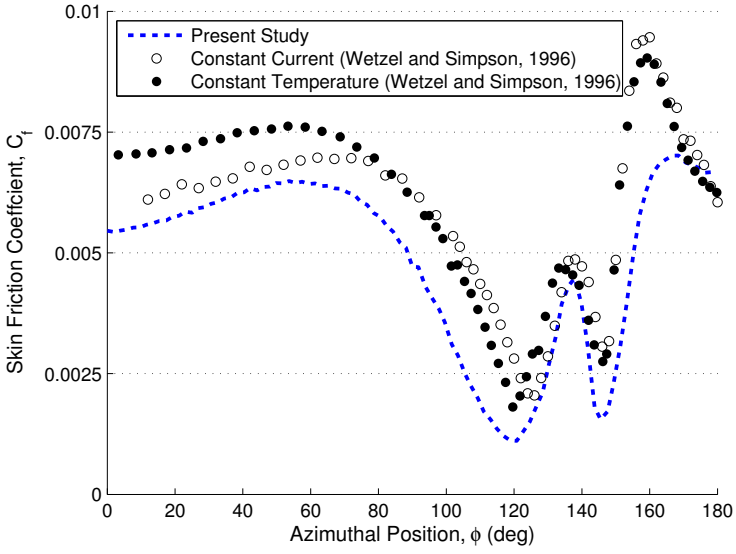
Wetzel's *pitch-up* maneuver, which is illustrated in Figure 2.48, was simulated in the present study. In this maneuver, the spheroid was pitched about an axis at its center of volume from $\alpha = 0°$ to $\alpha = 30°$ in 11 nondimensional time units ($t' = \frac{tL}{U_\infty}$). The accelerations at the beginning and end of maneuver, while finite, were as large as could be accomplished with the experimental setup.

Simulation and Validation Details Simulation motion was prescribed via a user-defined field-function, expressing angular pitch velocity as a piece-wise function of time.

$$q(t) = \begin{cases} 0, & \text{for } t' < t_0 \\ q_{\text{pitch}}, & \text{for } t_0 < t' < t_f \\ 0, & \text{for } t' > t_f \end{cases} \tag{2.62}$$



(a) $\frac{x}{L} = 0.576$



(b) $\frac{x}{L} = 0.729$

Figure 2.47: Skin friction (C_f) distribution on 6:1 prolate spheroid, with $Re = 4.2 \times 10^6$ flow at $\alpha = 20°$. CFD results shown with experimental data [60].

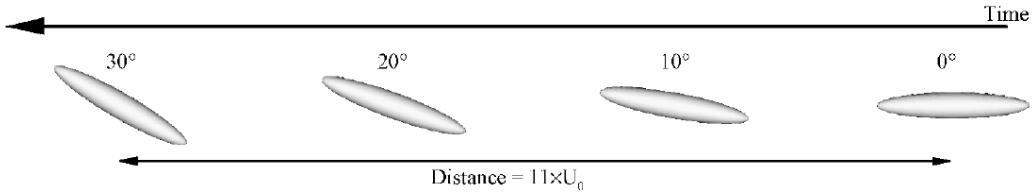


Figure 2.48: Diagram of spheroid pitch-up maneuvering [80].

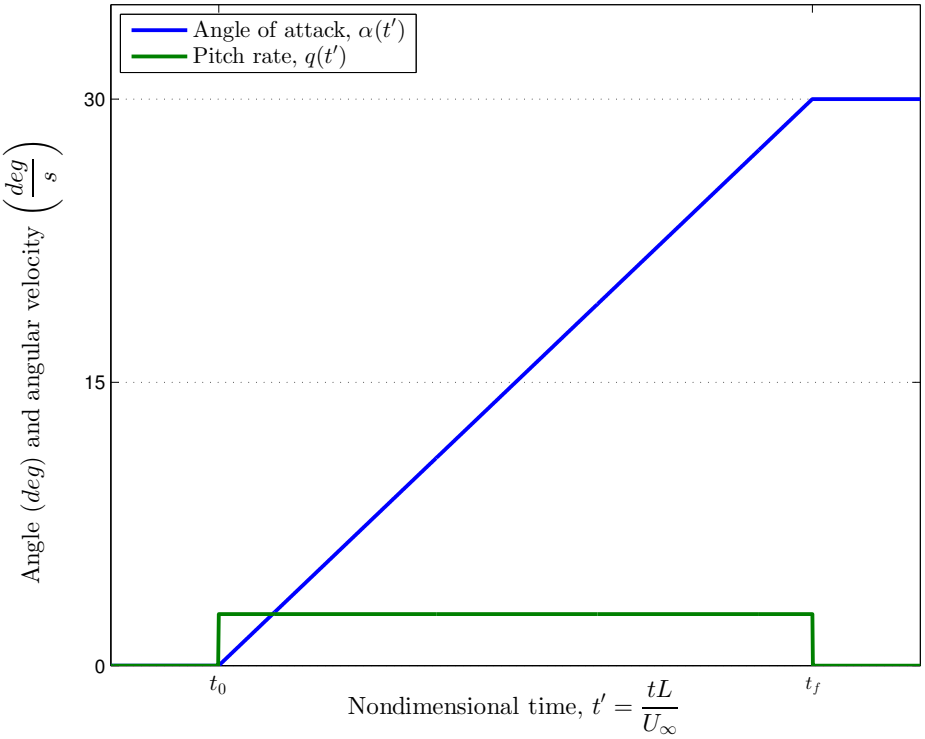


Figure 2.49: Prolate spheroid pitch-up maneuver motion.

This motion is depicted in Figure 2.49.

The motion described by (2.62) results in accelerations at the beginning and end of the maneuver that are inversely proportional to the simulation time-step, Δt.

$$|\dot{q}(t_0)| = |\dot{q}(t_f)| = \frac{\pi}{2\Delta t} \tag{2.63}$$

These high accelerations result in the introduction of transients to the solution.

The details of the simulation are the same as those given in Table 2.4. The simulation domain and the definition of boundary conditions are shown in Figure 2.50. A conic frustum

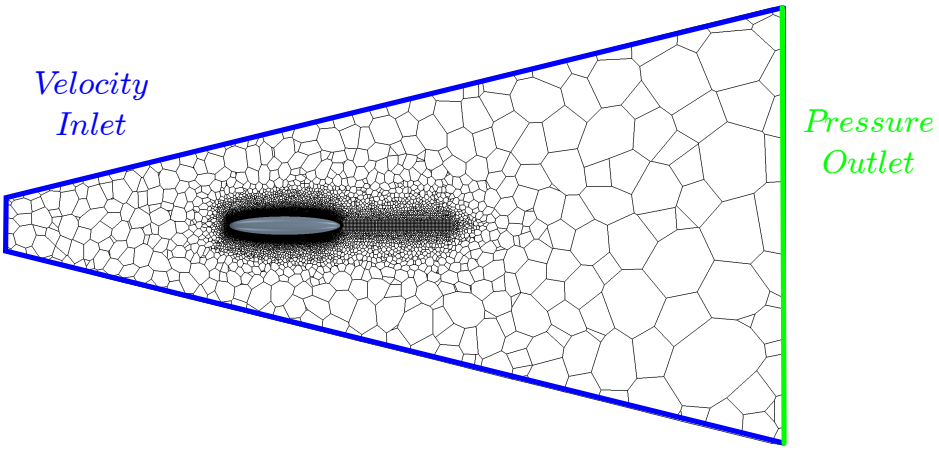


Figure 2.50: Computational domain used for spheroid pitch-up maneuver simulations.

shape was chosen to promote positive flow through the domain's velocity inlet boundaries during the pitch-up maneuver. A time-step size of $\Delta t = 9.3545 \times 10^{-3}$ s, corresponding to a nondimensional time-step size of $\Delta t' = 0.01$, was used in this simulation. This value matches that which was selected after a temporal refinement study conducted by Rhee and Hino [80]. Time-dependent boundary conditions were used to simulate the pitch-up motion of the spheroid (see Section 2.8).

Results The skin friction distribution on the spheroid at the end of the pitch-up maneuver ($\alpha = 30°$) is shown in Figure 2.51. Figure 2.52 shows a plot of the skin friction distribution over the course of the maneuver. This three-dimensional plot shows the skin friction distribution around the spheroid's circumference at an axial location of $\frac{x}{L} = 0.729$ (the position of this measurement is shown in Figure 2.51). Experimental data at three time locations from [60] is shown as well.

A more quantitative comparison can be made with two-dimensional plots. Figure 2.53 shows the skin friction distribution at three time locations during the pitch-up maneuver ($\alpha = 12.2°$, 20.2° and 29.9°). As with Figure 2.52, the skin friction is reported at an axial location of $\frac{x}{L} = 0.729$. The gray colored lines show an area of confidence based on the uncertainties reported by Wetzel (see Section 2.10.2.2). An additional data set from a similar RANS study conducted by Rhee and Hino [80] is also included. While errors in the prediction of skin friction magnitudes are larger than in the steady flows considered in Section 2.10.2.2, the CFD simulations do fairly well at predicting flow separation and reattachment locations. The CFD predictions from this study slightly under-perform those presented by Rhee and Hino in their ability to match skin friction magnitude. The differences in results from Rhee and Hino and simulations of this study are likely, at least partially, a result of differences in the computational meshes. While this study employed an unstructured polyhedral mesh, Rhee

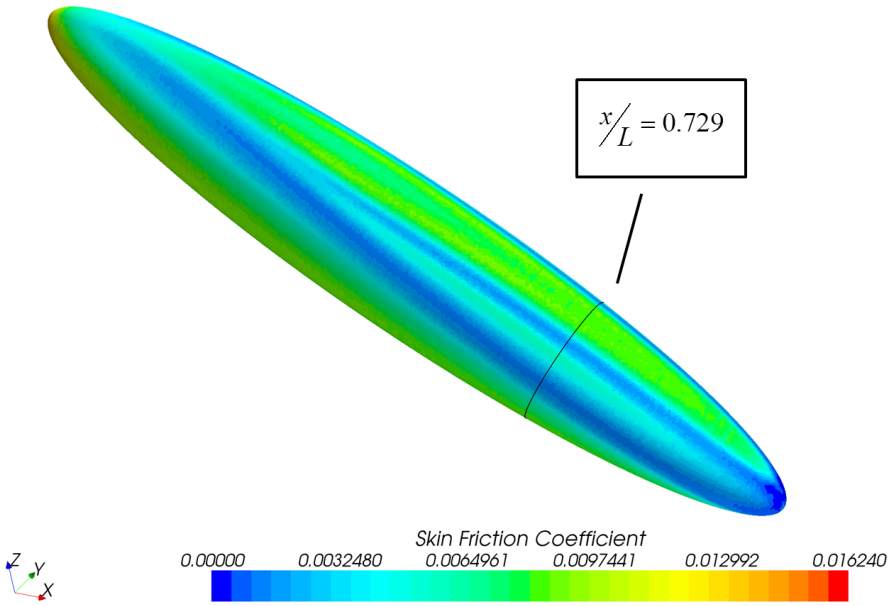


Figure 2.51: Spheroid skin friction distribution at end of the pitch-up maneuver ($\alpha = 30°$, $Re = 4.2 \times 10^6$). Line at $\frac{x}{L} = 0.729$ indicates location of comparison to experimental measurements.

and Hino used a structured hexahedral mesh. Also, Rhee and Hino used a Spalart-Allmaras turbulence closure model while a k-ω SST model was employed in this study.

Figure 2.54 shows the unsteady normal force and pitching moment predicted by the CFD simulation along side experimental data reported by Wetzel and Simpson and RANS results from Rhee and Hino.

$$C_Z = \frac{Z}{\frac{1}{2}\rho U_\infty^2 \pi r^2} \tag{2.64}$$

$$C_M = \frac{M}{\frac{1}{2}\rho U_\infty^2 \pi r^2 L} \tag{2.65}$$

Although both series of data are composed of discrete measurements, the high frequency of sampling makes showing each data point impractical. Thus, lines are plotted instead. Uncertainty bounds have been omitted in Figure 2.54, but are reported to be roughly 1.5%. The CFD simulation captures the general trend of the normal force and pitching moment reactions but over-predicts the magnitude of both. The numerical simulations also fail to reproduce the force and moment fluctuations seen in the experimental data. The results

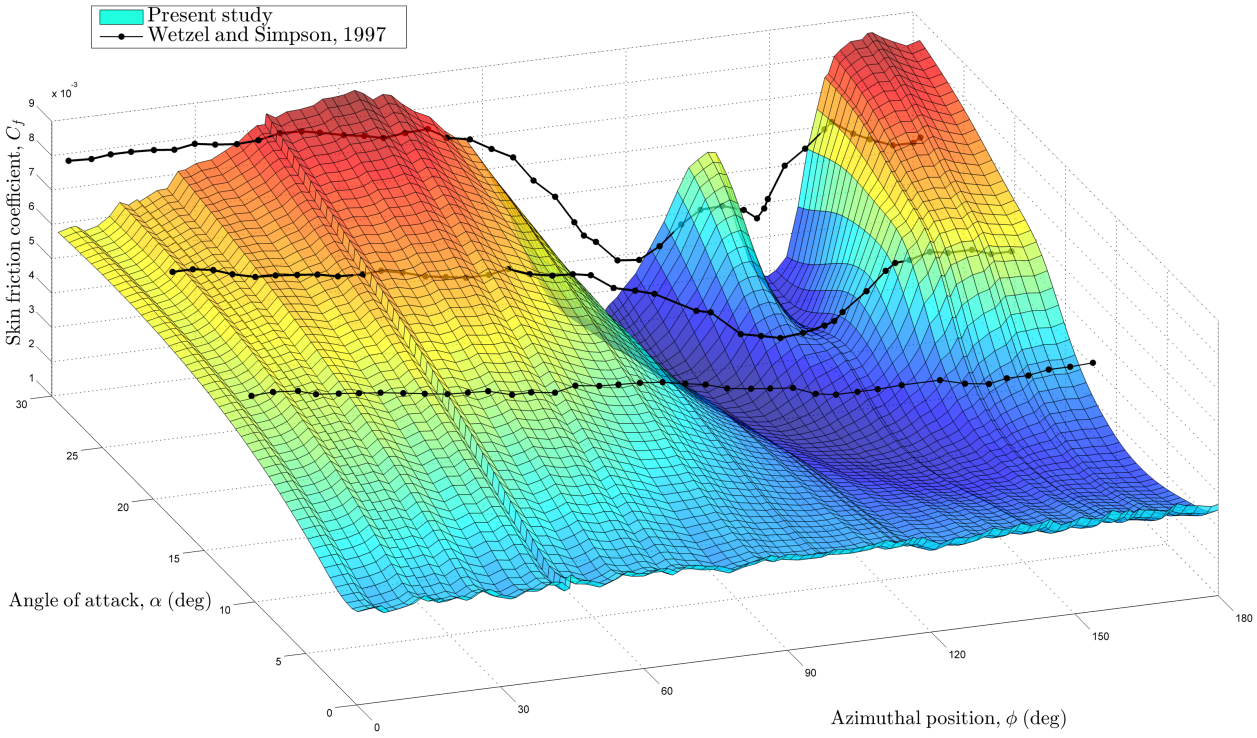
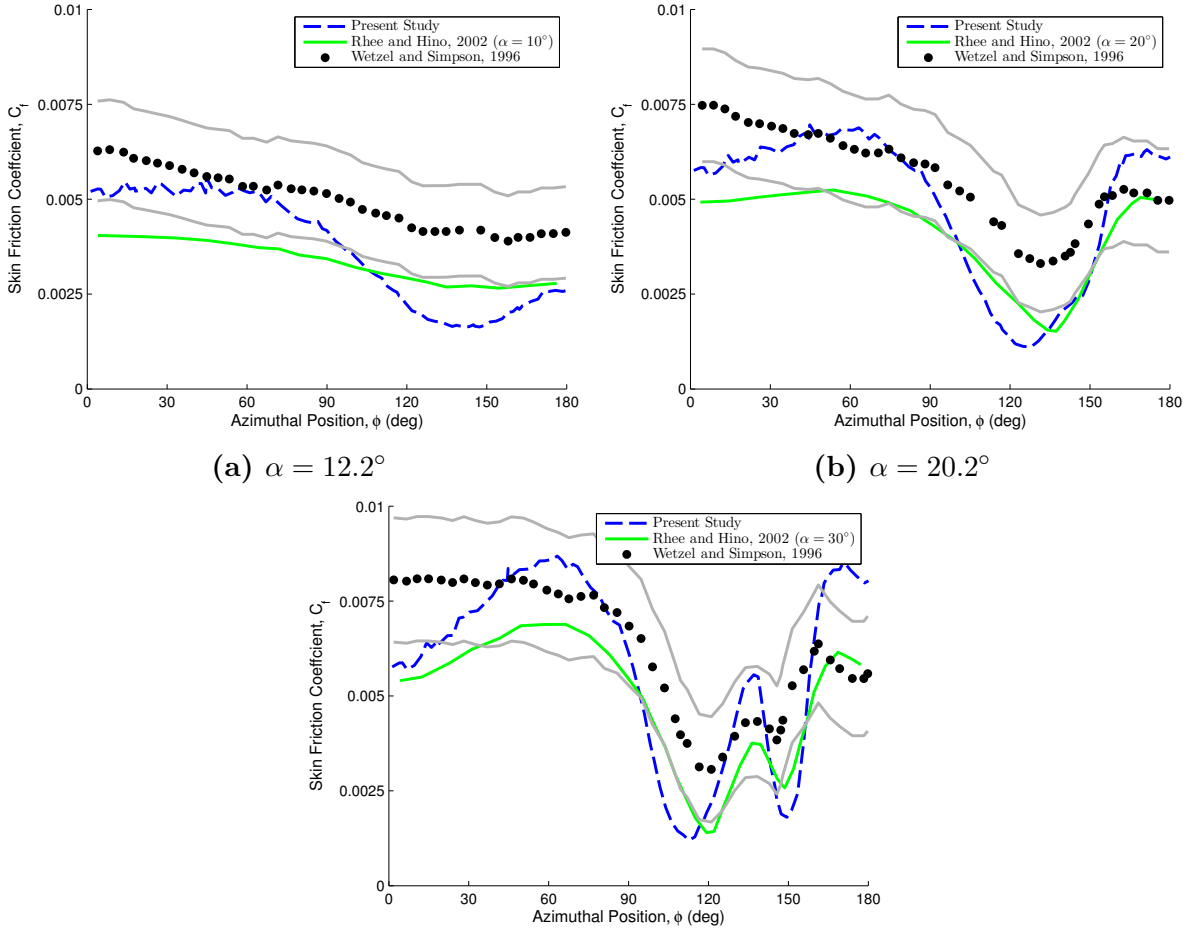


Figure 2.52: Skin friction coefficient, C_f, distribution on 6:1 prolate spheroid during *pitch-up* maneuver ($\frac{x}{L} = 0.729$, $Re = 4.2 \times 10^6$). CFD results shown in comparison to experimental measurements [60].

from simulations completed for this study show errors of a comparable magnitude with the experimental data as the RANS results presented by Rhee and Hino. The diminishing slope of the pitching moment coefficient (plotted in Figure 2.54b) is better captured in the results from the present study. One possible source of the error between the numerical and experimental results in Figure 2.54 is the process by which the experimental force and moment on the spheroid were found. Load cells in the sting were used to measure the force and moment on the spheroid. The aerodynamic forces and moments were obtained by subtracting those measured with tunnel flow shut off from those measured with the flow on. This subtraction process was used to remove inertial effects (which include rigid body and added mass) from the measured quantities.

Discussion The CFD simulations of an unsteady pitch-up maneuver of a 6:1 prolate spheroid show mixed agreement with experimental results, but performed with an accuracy similar to that of another RANS-based study. These findings indicate that while the applied CFD methods do a fair job of capturing unsteady fluid dynamics, there remains room for improvement. Results from a study using a higher-fidelity flow formulation (LES) show improved performance in this area [72].

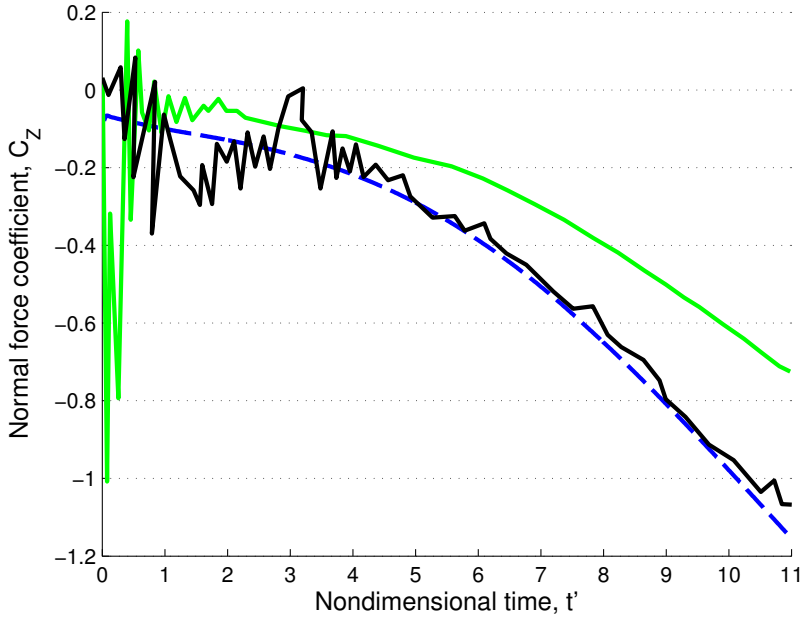


(a) $\alpha = 12.2°$

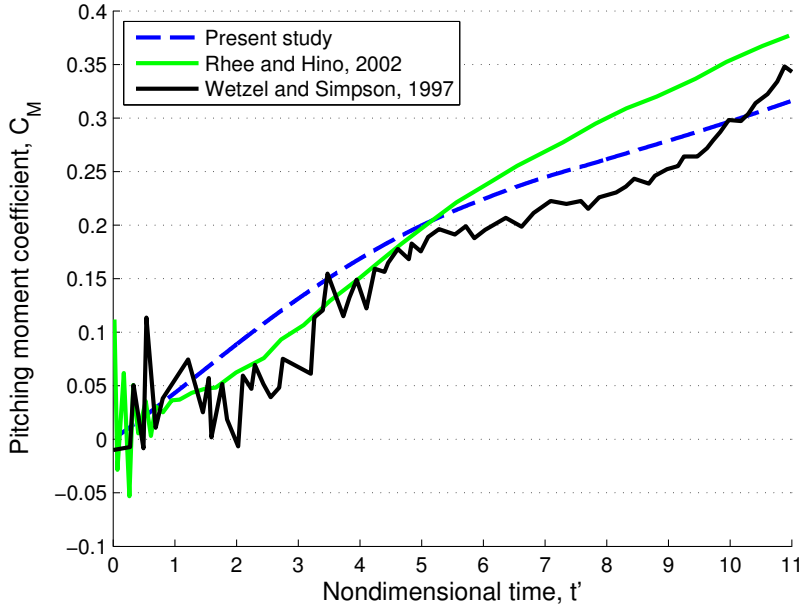
(b) $\alpha = 20.2°$

(c) $\alpha = 29.9°$

Figure 2.53: Skin friction coefficient, C_f, distribution on 6:1 prolate spheroid at various stages of *pitch-up* maneuver ($\frac{x}{L} = 0.729$, $Re = 4.2 \times 10^6$). CFD results shown in comparison to experimental measurements with error bars shown in gray [60] and another RANS study [80].



(a) Normal force, C_Z



(b) Pitching moment, C_M

Figure 2.54: Unsteady normal force and pitching moment during pitch-up maneuver. Predictions from CFD simulation are shown with experimental data [79] and results from a similar RANS study [80]. (Legend shown in (b) applies to both sub-figures.)

Chapter 3

Quasi-Steady State-Space Models

In addition to offering a high-fidelity alternative in the form of VFRM simulations, this study has also focused on the examination of quasi-steady state-space models. These efforts have yielded an increased understanding of the formulation, construction and usage of these models. Based on this review, three different state-space models have been developed for the GPAUV to be analyzed in comparison with VFRM simulations and experimental tests.

3.1 Vehicle Dynamics Nomenclature

In order to effectively discuss the state-space models used in underwater vehicle maneuvering analysis, it is necessary to more fully define a formal means of characterizing the rigid-body motion of a vehicle. The general convention in submarine analysis, and that which is adopted in this dissertation, is to use an aircraft-style coordinate system, like that shown in Figure 3.1. The location of the vehicle's centers of gravity and buoyancy can be defined within this frame, in relation to some arbitrary origin, as

$$\vec{r}_{CG} = \begin{bmatrix} x_G & y_G & z_G \end{bmatrix}^{\mathsf{T}} \tag{3.1}$$

$$\vec{r}_{CB} = \begin{bmatrix} x_B & y_B & z_B \end{bmatrix}^{\mathsf{T}}. \tag{3.2}$$

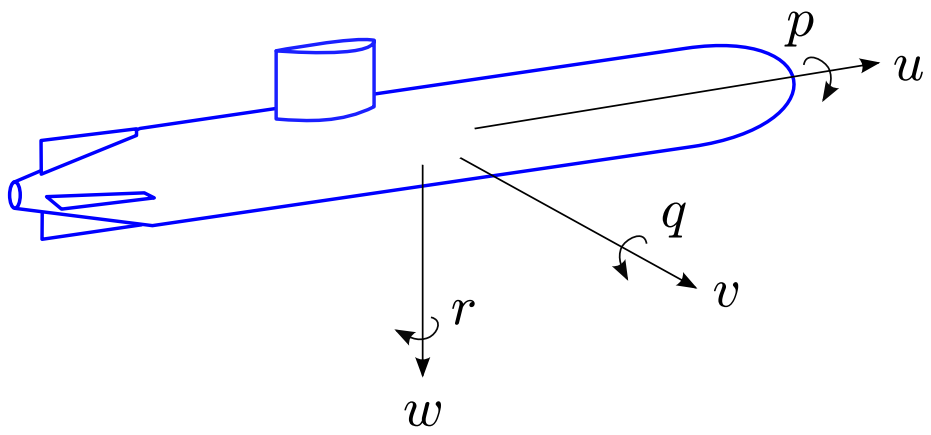


Figure 3.1: Coordinate system for submarines and AUVs showing velocity (u v, w, p, q, r).

It is useful to adopt a vectorized representation of the vehicle dynamics. The vehicle's velocity and acceleration can be defined as follows.

$$\vec{\nu} = \begin{bmatrix} \vec{\nu}_1 \\ -- \\ \vec{\nu}_2 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ -- \\ p \\ q \\ r \end{bmatrix} \tag{3.3}$$

$$\dot{\vec{\nu}} = \begin{bmatrix} \dot{\vec{\nu}}_1 \\ -- \\ \dot{\vec{\nu}}_2 \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ -- \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{3.4}$$

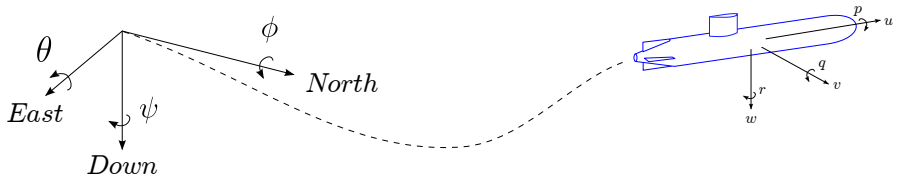


Figure 3.2: Euler angles in North-East-down coordinate system

The position of the vehicles center of gravity and its orientation can be defined with the addition of a similarly oriented ("North-East-down") inertial reference frame.

$$\vec{\eta} = \begin{bmatrix} \vec{\eta}_1 \\ -- \\ \vec{\eta}_2 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -- \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{3.5}$$

Figure 3.2 shows the definition of the Euler angles ϕ, θ and ψ in a North-East-down coordinate system.

It is also useful to define a vector that contains the vehicle's control inputs. For the purposes of this discussion, the control input can be defined by

$$\vec{u} = \begin{bmatrix} \vec{\delta} \\ -- \\ RPM \end{bmatrix} = \begin{bmatrix} \delta_{dp} \\ \delta_r \\ -- \\ RPM \end{bmatrix}. \tag{3.6}$$

Here, δ_{dp} and δ_r represent the dive plane and rudder deflection command. RPM is the propellers rotational rate.

Together, these parameter's define the vehicle's *state*.[1]

$$\vec{\chi} = \begin{bmatrix} \vec{\nu} \\ \dot{\vec{\nu}} \\ \vec{\eta}_2 \\ \vec{u} \end{bmatrix} \tag{3.7}$$

[1] This is the vehicle state used by a state-space model. The state used by a vehicle control algorithm may be different. For the GPAUV, $\vec{\chi}_{control} = [\vec{\eta}, \vec{\nu}]^\mathsf{T}$

3.2 Fundamental Concept

The basic approach of state-space modeling can be summarized by the concept of using a vehicle's current state to determine its state at some time in the near future.

$$\vec{\chi}\left(t + \Delta t\right) = f\left(\vec{\chi}\left(t\right), t\right) \tag{3.8}$$

This is accomplished by using various modeling approaches to predict the forces and moments (and therefore accelerations) on the vehicle as a function of its state. Accelerations can be determined from these forces and moments, allowing for the state to be advanced over time using a numerical integration scheme.

3.3 Model Components

A wide variety of quasi-steady state-space models have been formulated to predict the dynamics of underwater vehicles. Although a brief survey of literature within the field reveals an overwhelming variety of formulations, most models are constructed within the framework illustrated in Figure 3.3. Models are composed of a number of components (i. e., sub-models) which function mostly as separate entities.

3.3.1 Rigid-Body Kinematics

The Newton-Euler equations are generally used for the rigid-body kinematics sub-model shown in Figure 3.3. The equations relate the accelerations (linear and angular) of a body-fixed coordinate system to external forces and moments. The Newton-Euler equations are presented in Appendix A.1.1.

These motions can be related back to the inertial reference frame using Euler angles or quaternions. Fossen provides descriptions covering both of these methods [15, 17, 81].

3.3.2 Hydrostatics

The hydrostatic sub-model in Figure 3.3 includes effects due to gravity and buoyancy. For a fully submerged vehicle such as an AUV, the forces and moments due to this balance can be

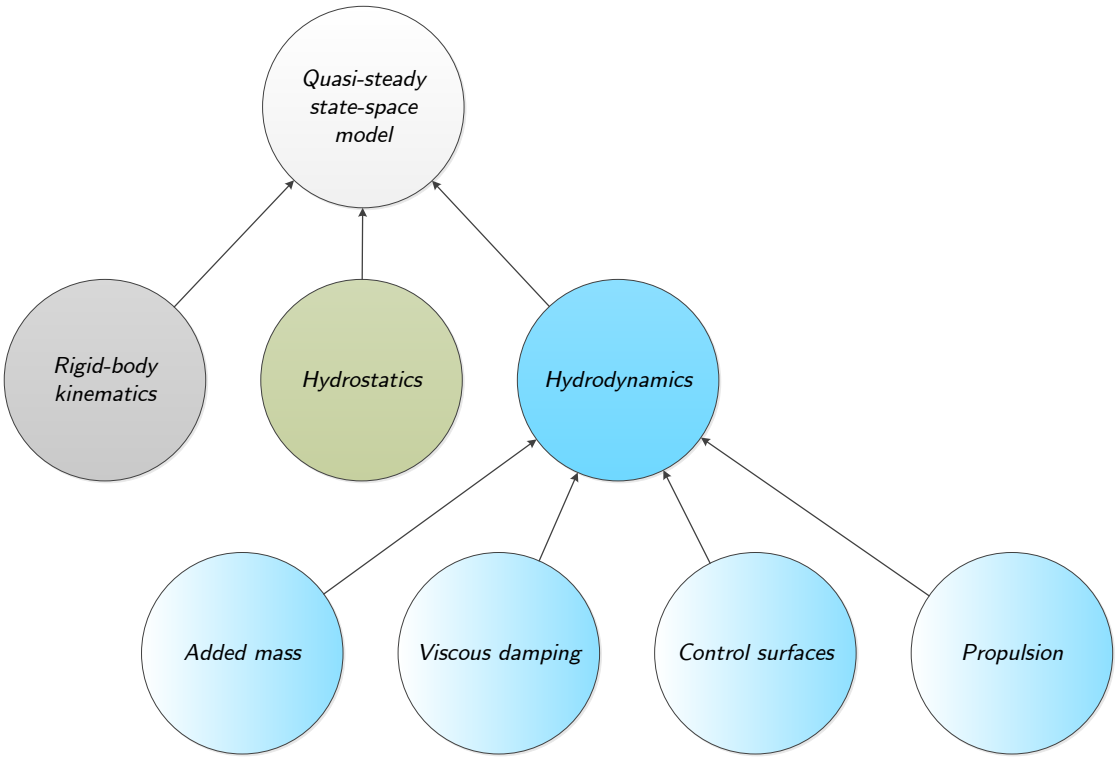


Figure 3.3: Common state-space model components.

written as a function of the vehicle's orientation

$$\vec{g}\left(\vec{\eta}\right) = \begin{bmatrix} (W-B)\sin(\theta) \\ -(W-B)\cos(\theta)\sin(\phi) \\ -(W-B)\cos(\theta)\cos(\phi) \\ -(y_G W - y_B B)\cos(\theta)\cos(\phi) + (z_G W - z_B B)\cos(\theta)\sin(\phi) \\ (z_G W - z_B B)\sin(\theta) + (x_G W - x_B B)\cos(\theta)\sin(\phi) \\ -(x_G W - x_B B)\cos(\theta)\sin(\phi) + (y_G W - y_B B)\sin(\theta) \\ . \end{bmatrix} \tag{3.9}$$

Here, W and B are the vehicle's weight and buoyancy respectively. Note that (3.9) gives the hydrostatic forces and moments in the body-fixed reference frame.

3.3.3 Unsteady Ideal Fluid Dynamics and Added Mass

Added mass effects, which provide additional inertia to a submerged body, can be derived from ideal flow theory. The following derivation is presented to provide a theoretically-based method for determining the added mass of a body. In addition to this, the derivation also serves as the basis for a discussion of unsteady fluid dynamics.

Consider a rigid-body moving at an arbitrary linear velocity, $\vec{\nu}$, within some infinite ideal fluid with a density ρ.[1] If the velocity of the fluid is represented by the gradient of a potential scalar, ϕ, the fluid force on the rigid-body can be obtained from

$$\vec{F} = \frac{\partial}{\partial t} \iint \rho \phi \vec{n} \, dS, \tag{3.10}$$

where $\vec{n}$ is the vector outward normal from the body's surface, S.

The scalar velocity potential, ϕ, can be defined by the body geometry, $\vec{r}$, and its velocity.

$$\begin{aligned} \phi &= f\left(\vec{r}, \vec{\nu}\left(t\right)\right) \\ &= f\left(\vec{r}, t\right) \end{aligned} \tag{3.11}$$

Here, the vector $\vec{r}$ gives the position with respect to some body-fixed origin. As the time dependence of ϕ is due only to velocity of the body, it can be deduced that when moving at a constant velocity, the force experienced by the body will be zero (*d'Alembert's paradox*).

If $\vec{F_e}$ is used to represent an external force used to accelerate the body and $\vec{F}$ as the counter-acting force applied to the body by the fluid, Newton's second law can be written as

$$\frac{d}{dt}\left(m\vec{\nu}\right) = \vec{F_e} + \vec{F}. \tag{3.12}$$

Using (3.10) for the fluid force, this can be rewritten as

$$\vec{F_e} = \frac{d}{dt}\left[\left(m\vec{\nu}\right) - \iint \rho \phi \vec{n} \, dS\right] \tag{3.13}$$

The last term in (3.13) is the negative impulse on the fluid field.

$$\vec{I} = -\iint \rho \phi \vec{n} \, dS \tag{3.14}$$

[1] The following derivation and explanation follows those presented by Lamb, Karamcheti and Newman [12, 82, 83].

Thus (3.13) can be rewritten as

$$\vec{F_e} = \frac{d}{dt}\left(m\vec{\nu} + \vec{I}\right). \tag{3.15}$$

Equation (3.15) shows that an external force applied to a rigid body is balanced by a change in the body's momentum, $\frac{d}{dt}(m\vec{\nu})$, as well as *a change in the impulse applied to the fluid by the body*, $\frac{d}{dt}\vec{I}$. The change in impulse on the fluid is thus related to the inertia of the rigid-body.

For an ideal incompressible fluid, Laplace's equation is enforced throughout the domain.

$$\vec{\nabla}^2\phi = 0 \tag{3.16}$$

Additionally, a boundary condition on the surface of the rigid-body is needed to insure that normal velocity of the near surface flow be equal to that of the surface itself.

$$\vec{\nabla}\phi \cdot \vec{n} = \frac{\partial\phi}{\partial n} = \vec{\nu}(t) \cdot \vec{n} \text{ on S} \tag{3.17}$$

To limit the energy of the system to a finite magnitude, the gradient of the velocity potential, $\vec{\nabla}\phi$, must also vanish as the distance from the body extends towards infinity. As these conditions are all linear equations, the velocity potential can be assumed to be of a form

$$\phi = \phi_1 + \phi_2 + \phi_3. \tag{3.18}$$

Each of these components of the velocity potential can be defined as a linear function of the corresponding component of body velocity.

$$\phi_i = \nu_i\varphi_i \quad \text{for } i = 1, 2, 3 \tag{3.19}$$

Here, the body velocity is denoted by

$$\vec{\nu}(t) = \nu_1(t)\,\hat{i} + \nu_2(t)\,\hat{j} + \nu_3(t)\,\hat{k}. \tag{3.20}$$

The terms φ_i are solely functions of position, while the terms ν_i are functions of time This This distinction is significant, as the added mass coefficients subsequently derived using the terms φ_i are thus independent of velocity and time. This independence contributes the quasi-steady nature of most state-space models.

Substituting (3.18) into (3.14), and recalling that the velocity $\vec{\nu}$ is constant on the body's surface, S, gives

$$\begin{aligned} -\vec{I} &= \iint \rho\phi\vec{n}\, dS \\ &= \iint \rho\left(\vec{\nu}\cdot\vec{\varphi}\right)\vec{n}\, dS \\ &= \iint \rho\left(\sum_{k}^{3}\nu_k\varphi_k\right)\vec{n}dS \\ &= \sum_{k}^{3}\left(\iint \rho\varphi_k\vec{n}dS\right)\nu_k. \end{aligned} \tag{3.21}$$

If the components of the vector, $\vec{I}$, are written separately as

$$\vec{I} = I_1\hat{i} + I_2\hat{j} + I_3\hat{k}, \tag{3.22}$$

then (3.21) can be rewritten as

$$I_i = \sum_{k}^{3}\left(-\iint \rho\varphi_k n_i dS\right)\nu_k. \tag{3.23}$$

The integral term within the parenthesis of (3.23) is generally referred to as the added mass.

$$\begin{aligned} m_{ki} &= -\rho\iint \varphi_k n_i\, dS \\ &= -\rho\iint \varphi_k\left(\frac{\partial\varphi_i}{\partial n}\right) dS \end{aligned} \tag{3.24}$$

Physically, the term m_{ki} represents the fluid inertial effect in the i^{th} direction due to a body motion in the k^{th} direction. The result shown in (3.24) can be used to find the added mass coefficients for any body as a function of only its geometry (and density of the fluid within which it is immersed).

The concept of added mass is often explained as an additional inertia caused by the need to accelerate a volume of fluid that surrounds a body. This is a fairly good description, although as Newman notes, there is not, in reality, some finite volume of fluid accelerating with the body. Instead, there exists a continuum of fluid particles accelerating at varying rates, depending on relative position and distance to the body [12].

If the moments and forces on a body moving with arbitrary linear and angular velocity are considered, the added mass of a body can be defined by a six-by-six matrix. Following the

nomenclature convention from (3.24), the full added mass matrix for a body can be written as

$$\mathbf{M}_A = -\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & m_{16} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} & m_{26} \\ m_{31} & m_{32} & m_{33} & m_{34} & m_{35} & m_{36} \\ m_{41} & m_{42} & m_{43} & m_{44} & m_{45} & m_{46} \\ m_{51} & m_{52} & m_{53} & m_{54} & m_{55} & m_{56} \\ m_{61} & m_{62} & m_{63} & m_{64} & m_{65} & m_{66} \end{bmatrix}. \tag{3.25}$$

While the previous derivation considered only linear motion and forces, the formulation defined in (3.24) holds true for rotational motion and moments, and can be used to obtain each term in (3.25).

3.3.3.1 Properties of the Added Mass Matrix

Symmetry of the Added Mass Matrix The six-by-six added mass matrix can be shown to be symmetric, and thus contain twenty-one independent elements. Rewriting (3.24) as

$$m_{ki} = \rho \iint \varphi_i \left(\vec{\nabla} \varphi_k \cdot \hat{n} \right) \, dS, \tag{3.26}$$

Green's theorem can be applied to give

$$\begin{aligned} m_{ki} &= \rho \iiint \vec{\nabla} \cdot \left(\varphi_i \vec{\nabla} \varphi_k \right) \, d\forall \\ &= \rho \iiint \left(\vec{\nabla} \varphi_i \cdot \vec{\nabla} \varphi_k + \varphi_i \vec{\nabla}^2 \phi_k \right) \, d\forall. \end{aligned} \tag{3.27}$$

Recalling (3.16), (3.27) can be reduced to

$$m_{ki} = \rho \iiint \left(\vec{\nabla} \varphi_i \cdot \vec{\nabla} \varphi_k \right) \, d\forall \tag{3.28}$$

Thus, the added mass matrix is symmetric, as the formulas to find m_{ki} and m_{ik} are the same.[1]

[1] Although the property of a symmetric added mass matrix is dependent on the assumption of an ideal fluid, added mass matrices obtained experimentally in real fluids have been shown in to be nearly symmetric [84].

Effect of Body Symmetry The added mass matrices of bodies with planes and/or axes of symmetry are sparse. This arises directly from (3.24). For a vehicle with port-starboard symmetry, $\mathbf{M}_A$ has twelve unique non-zero elements.

$$\mathbf{M}_A = \begin{bmatrix} m_{11} & 0 & m_{13} & 0 & m_{15} & 0 \\ & m_{22} & 0 & m_{24} & 0 & m_{26} \\ & & m_{33} & 0 & m_{35} & 0 \\ & & & m_{44} & 0 & m_{46} \\ & & & & m_{55} & 0 \\ & sym. & & & & m_{66} \end{bmatrix} \tag{3.29}$$

If the vehicle has both port-starboard and top-bottom symmetry, $\mathbf{M}_A$ is reduced to only 8 unique non-zero elements.

$$\mathbf{M}_A = \begin{bmatrix} m_{11} & 0 & 0 & 0 & 0 & 0 \\ & m_{22} & 0 & 0 & 0 & m_{26} \\ & & m_{33} & 0 & m_{35} & 0 \\ & & & m_{44} & 0 & 0 \\ & & & & m_{55} & 0 \\ & sym. & & & & m_{66} \end{bmatrix} \tag{3.30}$$

The added mass matrix of a body such as a prolate spheroid, with three planes of symmetry, oriented with the x-axis along its major diameter, contains only three unique non-zero elements.

$$\mathbf{M}_A = \begin{bmatrix} m_{11} & 0 & 0 & 0 & 0 & 0 \\ & m_{22} = m_{33} & 0 & 0 & 0 & 0 \\ & & m_{33} = m_{22} & 0 & 0 & 0 \\ & & & 0 & 0 & 0 \\ & & & & m_{55} = m_{66} & 0 \\ & sym. & & & & m_{66} = m_{55} \end{bmatrix} \tag{3.31}$$

3.3.3.2 Relation to Kinetic Energy

To understand the full implications of added mass effects on rigid-body dynamics, it is useful to consider the kinetic energy of the fluid surrounding the vehicle, T_{KE}.

$$\begin{aligned} T_{KE} &= -\frac{1}{2} \iint \rho \phi \frac{\partial \phi}{\partial n} \, dS \\ &= -\frac{1}{2} \iint \rho \phi \vec{\nabla} \phi \cdot \vec{n} \, dS \end{aligned} \tag{3.32}$$

See Karamcheti for the derivation that leads to (3.32) [83]. Recalling that (3.17) gives the boundary condition for the fluid at the surface of the body, (3.32) can be rewritten as

$$\begin{aligned} T_{KE} &= -\frac{1}{2}\iint \rho\phi\vec{\nu}\cdot\vec{n}\,dS \\ &= -\frac{1}{2}\vec{\nu}\cdot\iint \rho\phi\vec{n}\,dS. \end{aligned} \tag{3.33}$$

The integral in (3.33) can be recognized as the impulse on the fluid, previously defined in (3.14).

$$\begin{aligned} T_{KE} &= -\frac{1}{2}\vec{\nu}\cdot\vec{I} \\ &= \frac{1}{2}\sum_i \nu_i I_i \end{aligned} \tag{3.34}$$

Referencing back to (3.23) and (3.24), this result can be further expanded to

$$T_{KE} = \frac{1}{2}\sum_i \left(\sum_k m_{ik}\nu_k\right)\nu_i. \tag{3.35}$$

The expression for kinetic energy shown in (3.35) can be used to derive the rigid-body dynamic expression for an arbitrary body submerged in an ideal fluid.

At this point, it is beneficial to switch from the notation shown in (3.24) and (3.35) to that established by the Society of Naval Architects and Marine Engineers (SNAME) in 1950 [85].[1]

$$\mathbf{M}_A = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{3.36}$$

[1] This same notation style is also used for viscous damping coefficients, which will be discussed later, with the subscript indicating the excitation(s) which the coefficient are a function of (e. g., $M_{vr} = \frac{\partial^2 M}{\partial v \partial r}$).

Here, the velocities follow the convention discussed in Section 3.1.

$$\vec{\nu} = \begin{bmatrix} \vec{v} \\ \vec{\omega} \end{bmatrix} = \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \\ \nu_6 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \tag{3.37}$$

This notation is somewhat more physically intuitive than the index-style notation. For instance, $X_{\dot{v}}$ denotes the force in the x-direction per a unit of acceleration in the y-direction.

$$X_{\dot{v}} = \frac{\partial X}{\partial \dot{v}} = -m_{12} \tag{3.38}$$

3.3.3.3 Kirchoff's Equations

The derivation presented earlier in this section, which considered only forces (not moments) due to linear (not angular) motion, does not provide a representation for the full effect of added mass on a rigid-body kinematic system. A derivation using the kinetic energy of fluid and Kirchoff's equations for the motion of a rigid-body in an ideal fluid, shows that each added mass element contributes to motion in multiple axes of the rigid-body system. Rewriting (3.35) in matrix form

$$T_{KE} = -\frac{1}{2}\vec{\nu}^T \mathbf{M}_A \vec{\nu} \tag{3.39}$$

and expanding gives[1]

$$\begin{aligned} 2T_{KE} = &- X_{\dot{u}}u^2 - Y_{\dot{v}}v^2 - Z_{\dot{w}}w^2 - 2Y_{\dot{w}}vw - 2X_{\dot{w}}wu - 2X_{\dot{v}}uv \\ &- K_{\dot{p}}p^2 - M_{\dot{q}}q^2 - N_{\dot{r}}r^2 - 2M_{\dot{r}}qr - 2K_{\dot{r}}rp - 2K_{\dot{q}}pq \\ &- 2p\left(X_{\dot{p}}u + Y_{\dot{p}}v + Z_{\dot{p}}w\right) \\ &- 2q\left(X_{\dot{q}}u + Y_{\dot{q}}v + Z_{\dot{q}}w\right) \\ &- 2r\left(X_{\dot{r}}u + Y_{\dot{r}}v + Z_{\dot{r}}w\right) \end{aligned} \tag{3.40}$$

With this expression, Kirchhoff's equations[2] can be applied to find the forces (X, Y, Z) and moments (K, M, N) acting on the rigid-body in an ideal fluid in relation to the kinetic energy.

[1] Here, symmetry of $\mathbf{M}_A$ has been assumed (i.e., $X_{\dot{v}} = Y_{\dot{u}} \dots$).

[2] See Milne-Thomson [86] and Miloh and Landweber [87] for derivations of Kirchoff's equations

$$\begin{aligned}\text{Forces} &= -\frac{\partial}{\partial t}\left(\frac{\partial T_{KE}}{\partial \vec{v}}\right) - \vec{\omega} \times \left(\frac{\partial T_{KE}}{\partial \vec{v}}\right) \\ \text{Moments} &= -\frac{\partial}{\partial t}\left(\frac{\partial T_{KE}}{\partial \vec{\omega}}\right) - \vec{\omega} \times \left(\frac{\partial T_{KE}}{\partial \vec{\omega}}\right) - \vec{v} \times \left(\frac{\partial T_{KE}}{\partial \vec{v}}\right)\end{aligned} \tag{3.41}$$

These relations can be written for each force and moment component as

$$\begin{aligned}X &= -\frac{\partial}{\partial t}\left(\frac{\partial T_{KE}}{\partial u}\right) - q\frac{\partial T_{KE}}{\partial w} + r\frac{\partial T_{KE}}{\partial v} \\ Y &= -\frac{\partial}{\partial t}\left(\frac{\partial T_{KE}}{\partial v}\right) - r\frac{\partial T_{KE}}{\partial u} + p\frac{\partial T_{KE}}{\partial w} \\ Z &= -\frac{\partial}{\partial t}\left(\frac{\partial T_{KE}}{\partial w}\right) - p\frac{\partial T_{KE}}{\partial v} + q\frac{\partial T_{KE}}{\partial u} \\ K &= -\frac{\partial}{\partial t}\left(\frac{\partial T_{KE}}{\partial p}\right) - q\frac{\partial T_{KE}}{\partial r} + r\frac{\partial T_{KE}}{\partial q} - v\frac{\partial T_{KE}}{\partial w} + w\frac{\partial T_{KE}}{\partial v} \\ M &= -\frac{\partial}{\partial t}\left(\frac{\partial T_{KE}}{\partial q}\right) - r\frac{\partial T_{KE}}{\partial p} + p\frac{\partial T_{KE}}{\partial r} - w\frac{\partial T_{KE}}{\partial u} + u\frac{\partial T_{KE}}{\partial w} \\ N &= -\frac{\partial}{\partial t}\left(\frac{\partial T_{KE}}{\partial r}\right) - p\frac{\partial T_{KE}}{\partial q} + q\frac{\partial T_{KE}}{\partial p} - u\frac{\partial T_{KE}}{\partial v} + v\frac{\partial T_{KE}}{\partial u}.\end{aligned} \tag{3.42}$$

The six partial derivatives in (3.42) are

$$\begin{aligned}\frac{\partial T_{KE}}{\partial u} &= -X_{\dot{u}}u - X_{\dot{w}}w - X_{\dot{v}}v - X_{\dot{p}}p - X_{\dot{q}}q - X_{\dot{r}}r \\ \frac{\partial T_{KE}}{\partial v} &= -Y_{\dot{u}}u - Y_{\dot{w}}w - Y_{\dot{v}}v - Y_{\dot{p}}p - Y_{\dot{q}}q - Y_{\dot{r}}r \\ \frac{\partial T_{KE}}{\partial w} &= -Z_{\dot{u}}u - Z_{\dot{w}}w - Z_{\dot{v}}v - Z_{\dot{p}}p - Z_{\dot{q}}q - Z_{\dot{r}}r \\ \frac{\partial T_{KE}}{\partial p} &= -K_{\dot{p}}p - K_{\dot{q}}q - K_{\dot{r}}r - K_{\dot{u}}u - K_{\dot{w}}w - K_{\dot{v}}v \\ \frac{\partial T_{KE}}{\partial q} &= -M_{\dot{p}}p - M_{\dot{q}}q - M_{\dot{r}}r - M_{\dot{u}}u - M_{\dot{w}}w - M_{\dot{v}}v \\ \frac{\partial T_{KE}}{\partial r} &= -N_{\dot{p}}p - N_{\dot{q}}q - N_{\dot{r}}r - N_{\dot{u}}u - N_{\dot{w}}w - N_{\dot{v}}v.\end{aligned} \tag{3.43}$$

These can be substituted into (3.42) to give expressions for the forces and moments on the body due to ideal fluid effects.

$$\begin{aligned} X =& X_{\dot{u}}\dot{u} + X_{\dot{w}}\left(\dot{w} + uq\right) + X_{\dot{q}}\dot{q} + Z_{\dot{w}}wq + Z_{\dot{q}}q^2 \\ &+ X_{\dot{v}}\dot{v} + X_{\dot{p}}\dot{p} + X_{\dot{r}}\dot{r} - Y_{\dot{v}}vr - Y_{\dot{p}}rp - Y_{\dot{r}}r^2 \\ &+ X_{\dot{v}}ur - Y_{\dot{w}}wr \\ &+ Y_{\dot{w}}vq + Z_{\dot{p}}pq - \left(Y_{\dot{q}} - Z_{\dot{r}}\right)qr \end{aligned} \tag{3.44}$$

$$\begin{aligned} Y =& X_{\dot{v}}\dot{u} + Y_{\dot{w}}\dot{w} + Y_{\dot{q}}\dot{q} \\ &+ Y_{\dot{v}}\dot{v} + Y_{\dot{p}}\dot{p} + Y_{\dot{r}}\dot{r} + X_{\dot{v}}vr + Y_{\dot{w}}vp + X_{\dot{r}}r^2 + \left(X_{\dot{q}} - Z_{\dot{r}}\right)rp - Z_{\dot{p}}p^2 \\ &- X_{\dot{w}}\left(up - wr\right) + X_{\dot{u}}ur + Z_{\dot{w}}wp \\ &- Z_{\dot{q}}pq + X_{\dot{q}}qr \end{aligned} \tag{3.45}$$

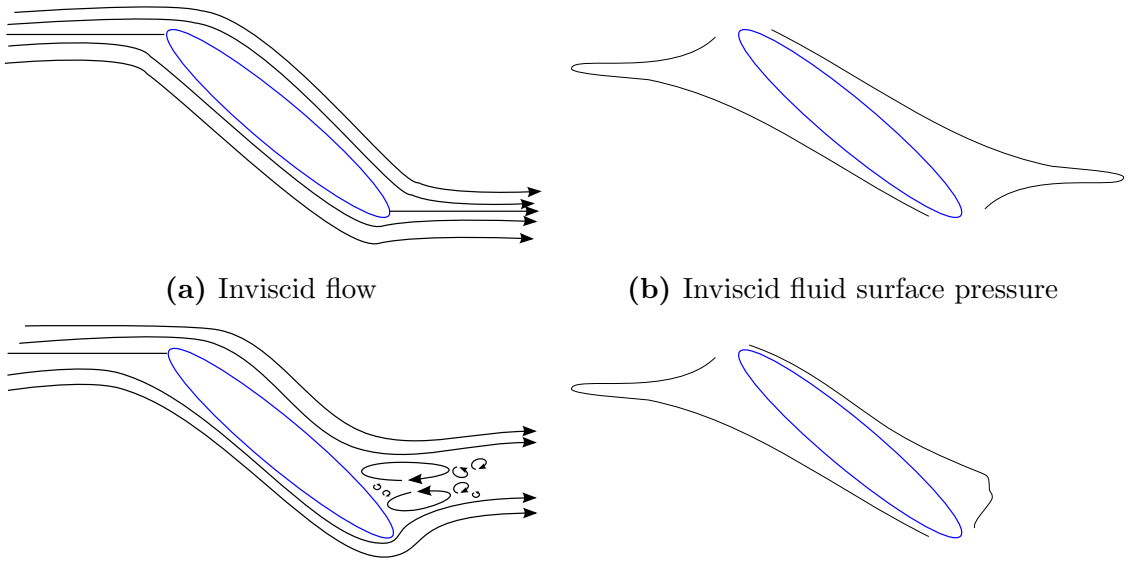
$$\begin{aligned} Z =& X_{\dot{w}}\left(\dot{u} - wq\right) + Z_{\dot{w}}\dot{w} + Z_{\dot{q}}\dot{q} - X_{\dot{u}}uq - X_{\dot{q}}q^2 \\ &+ Y_{\dot{w}}\dot{v} + Z_{\dot{p}}\dot{p} + Z_{\dot{r}}\dot{r} + Y_{\dot{v}}vp + Y_{\dot{r}}rp + Y_{\dot{p}}p^2 \\ &+ X_{\dot{v}}uq + Y_{\dot{w}}wp \\ &- X_{\dot{v}}vq - \left(X_{\dot{q}} - Y_{\dot{q}}\right)pq - X_{\dot{r}}qr \end{aligned} \tag{3.46}$$

$$\begin{aligned} K =& X_{\dot{p}}\dot{u} + Z_{\dot{p}}\dot{w} + K_{\dot{q}}\dot{q} - X_{\dot{v}}wu + X_{\dot{r}}uq - Y_{\dot{w}}w^2 - \left(Y_{\dot{q}} - Z_{\dot{r}}\right)wq + M_{\dot{r}}q^2 \\ &+ Y_{\dot{p}}\dot{v} + K_{\dot{p}}\dot{p} + K_{\dot{r}}\dot{r} + Y_{\dot{w}}v^2 - \left(Y_{\dot{q}} - Z_{\dot{r}}\right)vr + Z_{\dot{p}}vp - M_{\dot{r}}r^2 - K_{\dot{q}}rp \\ &+ X_{\dot{w}}uv - \left(Y_{\dot{v}} - Z_{\dot{w}}\right)vw - \left(Y_{\dot{r}} - Z_{\dot{q}}\right)wr - Y_{\dot{p}}wp - X_{\dot{q}}ur \\ &+ \left(Y_{\dot{r}} + Z_{\dot{q}}\right)vq + K_{\dot{r}}pq - \left(M_{\dot{q}} - N_{\dot{r}}\right)qr \end{aligned} \tag{3.47}$$

$$\begin{aligned} M =& X_{\dot{q}}\left(\dot{u} + wq\right) + Z_{\dot{q}}\left(\dot{w} - uq\right) + M_{\dot{q}}\dot{q} - X_{\dot{w}}\left(u^2 - w^2\right) - \left(Z_{\dot{w}} - X_{\dot{u}}\right)wu \\ &+ Y_{\dot{q}}\dot{v} + K_{\dot{q}}\dot{p} + M_{\dot{r}}\dot{r} + Y_{\dot{p}}vr - Y_{\dot{r}}vp - K_{\dot{r}}\left(p^2 - r^2\right) + \left(K_{\dot{p}} - N_{\dot{r}}\right)rp \\ &- Y_{\dot{w}}uv + X_{\dot{v}}vw - \left(X_{\dot{r}} + Z_{\dot{p}}\right)\left(up - wr\right) + \left(X_{\dot{p}} - Z_{\dot{r}}\right)\left(wp + ur\right) \\ &- M_{\dot{r}}pq + K_{\dot{q}}qr \end{aligned} \tag{3.48}$$

$$\begin{aligned} N =& X_{\dot{r}}\dot{u} + Z_{\dot{r}}\dot{w} + M_{\dot{r}}\dot{q} + X_{\dot{v}}u^2 + Y_{\dot{w}}wu - \left(X_{\dot{p}} - Y_{\dot{q}}\right)uq - Z_{\dot{p}}wq - K_{\dot{q}}q^2 \\ &+ Y_{\dot{r}}\dot{v} + K_{\dot{r}}\dot{p} + N_{\dot{r}}\dot{r} - X_{\dot{v}}v^2 - X_{\dot{r}}vr - \left(X_{\dot{p}} - Y_{\dot{q}}\right)vp + M_{\dot{r}}rp + K_{\dot{p}}p^2 \\ &- \left(X_{\dot{u}} - Y_{\dot{v}}\right)uv - X_{\dot{w}}vw + \left(X_{\dot{q}} + Y_{\dot{p}}\right)up + Y_{\dot{r}}ur + Z_{\dot{q}}wp \\ &- \left(X_{\dot{q}} + Y_{\dot{p}}\right)vq - \left(K_{\dot{p}} - M_{\dot{q}}\right)pq - K_{\dot{r}}qr \end{aligned} \tag{3.49}$$

Equations (3.44-3.49) are presented in the style introduced by Imlay, which has subsequently become commonplace, where each expression is organized into four lines [88]. The first line contains factors due to longitudinal components of motion (within the x-z plane). The second line contains components of lateral motion. The third line contains mixed quadratic terms



(a) Inviscid flow **(b)** Inviscid fluid surface pressure

(c) Viscous flow **(d)** Viscous fluid surface pressure

Figure 3.4: Spheroid at constant angle of attack.

involving either u or w. Components on the fourth line are second order terms which tend to be small and are often neglected.

3.3.3.4 Issues for Consideration

Munk Moments A well-known phenomenon described by (3.44-3.49) are the so-called *Munk moments*. These are moments experienced by a body due to added-mass effects. While a body moving with a constant linear velocity will experience no force within an ideal fluid (*d'Alembert's paradox*), it can in fact experience a moment reaction. This moment has a destabilizing effect on the body.

A classic example of a Munk moment is the $(Z_{\dot{W}} - X_{\dot{u}})\, wu$ term on the first line of (3.48). This represents the pitching moment experienced by a body when traveling at a steady angle of attack ($u = C_1, w = C_2, v = p = q = r = 0$), as shown in Figure 3.4a and 3.4b. Although the force in x-direction is zero, the centers of pressure on the windward and leeward surfaces are not coincident (see Figure 3.4b). Thus the equal pressure forces act through different points, creating a destabilizing moment on the body.

Viscous Added Mass Effects While much can be gained from an ideal flow analysis, it is important to note that there exists a discrepancy between this idealized field and that which would occur in a real, viscous fluid. In the viscous fluids that must be considered for AUV design, the flow field and pressure distribution on the inclined spheroid will tend to

look more like those depicted in Figure 3.4c and 3.4d. Adverse pressure gradients lead to flow separation, and a significantly different flow field. The surface pressures induced by this flow are, in turn, different than those in the ideal flow. In this situation, which is one of many in which added mass effects are influenced by viscosity, the destabilizing moment in an viscous fluid will be less than that in an inviscid fluid.

Aside from their overlap with viscous phenomena, added mass effects can be completely accounted for in the general form presented in (3.44-3.49). State-space models generally use either: (a) the complete nonlinear added mass formulation, or (b) only the linear added mass terms (those that are taken as a product with acceleration) in (3.44-3.49).

An additional reduction in added mass terms is often obtained from vehicle symmetry. As noted in Section 3.3.3.1, bodies with planes and/or axes of symmetry present a sparse added mass matrix. This allows for a substantial simplification in (3.44-3.49). Nearly all models for underwater vehicles are formulated with the assumption of port-starboard symmetry, reducing the added mass matrix to (3.29).

It is also not uncommon for added mass terms to be neglected based on an *a-priori* knowledge of their relative magnitude for similar vehicle geometries (e.g., cigar-shaped submarines). In-fact, many models neglect all but the diagonal terms of the added mass matrix, as they are often one to two orders of magnitude larger than the off-diagonal terms [89].

3.3.4 Viscous Damping

With no analytic solution to represent the viscous fluid flow that influences the motion of a vehicle, the approaches to modeling the viscous damping components of the system have the widest variation of those depicted in Figure 3.3. Taylor series expansions and so-called *cross-flow* formulations are the two most common approaches. Most viscous damping formulations use a combination of these approaches with a substantial number of terms omitted based on the specific application. An additional approach, referred to here as parametric damping, is also presented.

3.3.4.1 Taylor Series Expansion Damping

The use of Taylor series expansions in the modeling of viscous damping phenomena can appear at first to be a fairly heuristic approach, but proper consideration can lead to models that are, if not "first-principles-based," at least "first-principles-informed." This approach is generally credited to Abkowitz [11] and Strom-Tejsen [13], although the Gertler-Hagen equations also include Taylor series terms (see Appendix A.1.2).

The hydrodynamic forces on a vehicle operating at an equilibrium can be approximately defined as a function of a finite set of parameters.

$$\vec{\tau}_{\text{hydrodynamic}} = f\left(\vec{\nu}, \dot{\vec{\nu}}, \vec{\delta}\right) \tag{3.50}$$

Expanding (3.50) out to the third-order about a constant forward speed equilibrium for the X-force equation gives

$$\begin{aligned} X = X_0 + \left[X_u \Delta u + X_v v + X_w w + X_r r + \cdots + X_{\dot{u}} \dot{u} + \cdots + X_{\delta_1} \delta_1 + \ldots\right] \\ + \frac{1}{2!}\left[X_{uu} \Delta u^2 + \cdots + X_{\dot{u}\dot{u}} \dot{u}^2 + \cdots + X_{\delta_1 \delta_1} \delta_1^2 + \cdots + 2X_{uv} \Delta u v + \cdots + 2X_{\dot{r}\delta_1} \dot{r} \delta_1\right] \\ + \frac{1}{3!}\left[X_{uuu} \Delta u^3 + \cdots + 3X_{uuv} \Delta u^2 v + \cdots + 6X_{uvw} \Delta u v w + \ldots\right], \end{aligned} \tag{3.51}$$

where X_0 is the drag at the equilibrium point. The convention of truncating the series at the third-order is based on limitations in measurement capabilities and experience that an expansion of this order is sufficient for accurate prediction of most maneuvers [90]. Abkowitz also made the assumption that interactions between viscous and inertial phenomena are minimal. This leads to cancellation of terms such as $X_{\dot{u}\dot{u}}$, $X_{\dot{v}\dot{r}}$ and $X_{\dot{u}\dot{v}w}$.

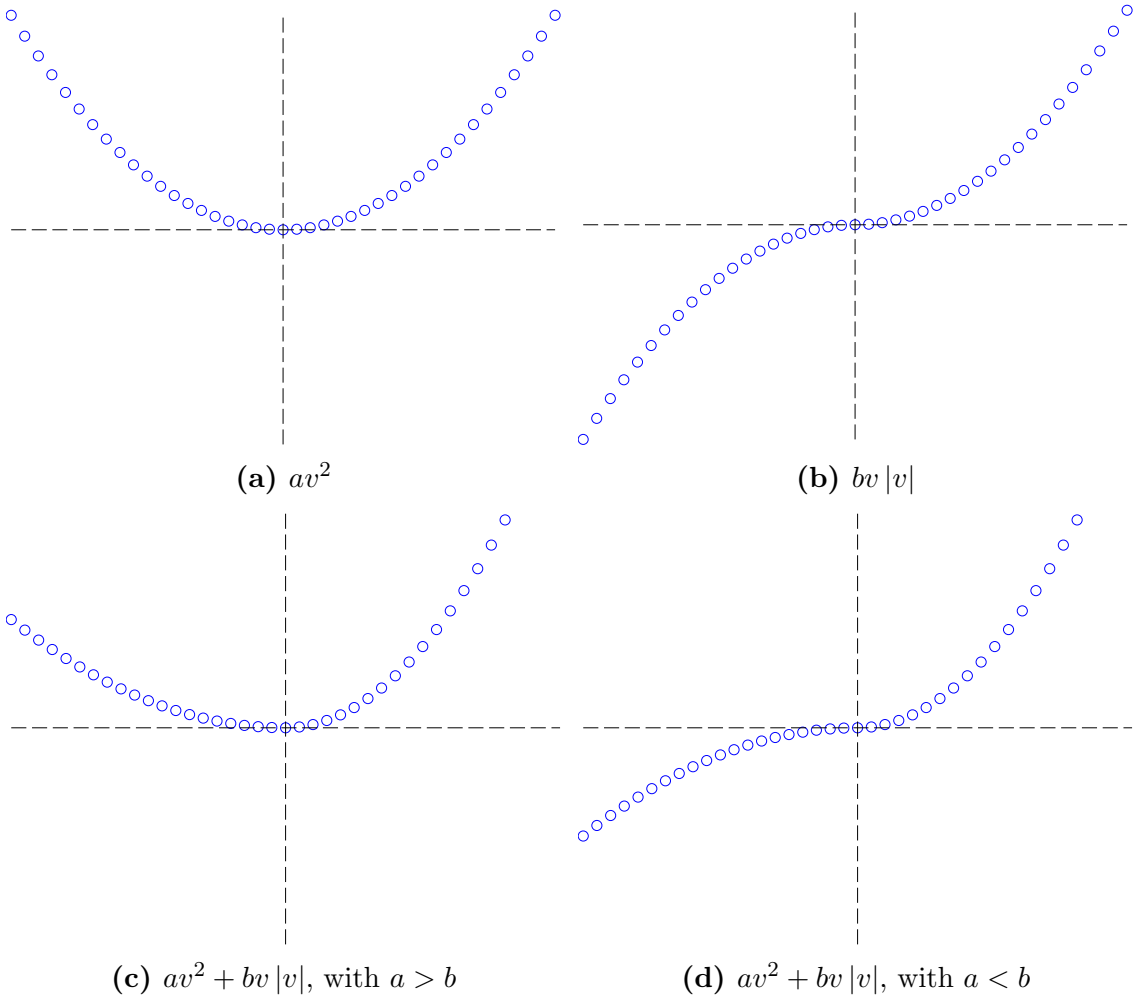
Generally, the number of terms is reduced substantially from those shown in (3.51). This is accomplished by characterizing the nature of each reaction. Considering the X-force due to a v-velocity as an example, the reaction to an excitation can generally be characterized by one of three trends.

$$X(v) = \begin{cases} \text{even:} & av^2 \\ \text{odd:} & bv\,|v| \\ \text{mixed:} & av^2 + bv\,|v| \end{cases} \qquad \text{where } a \text{ and } b \text{ are constants} \tag{3.52}$$

The terms *even*, *odd* and *mixed* refer the relationships illustrated by Figure 3.5a, b and c or d respectively. If X is a purely even function of v ($X(v) = av^2$), the reaction will follow the shape shown in Figure 3.5a. Alternatively, if X is a purely odd function of v ($X(v) = bv\,|v|$), the reaction will follow the shape shown in Figure 3.5b.[1] Figures 3.5c and d shown examples of cases where both even and odd terms of v appear ($X(v) = av^2 + bv\,|v|$).

An assumption of port-starboard symmetry (about the xz-plane) leads to a significant simplification of (3.51). Following the logic used by Abkowitz [11] to describe the expansion for a ship (which only considers motions in the x, y and r degrees-of-freedom), the relationship

[1] The odd quadratic function $X = bv\,|v|$, instead of the first order odd term $X = bv$, has been used here to reflect the generally quadratic behavior of viscous drag.



(a) av^2 **(b)** $bv\,|v|$

(c) $av^2 + bv\,|v|$, with $a > b$ **(d)** $av^2 + bv\,|v|$, with $a < b$

Figure 3.5: Reaction types for a second-order Taylor series, see (3.52).

between X and v can be illustrated by one of the three even function curves shown in Figure 3.6. A vehicle with port-starboard symmetry will experience the same reaction in X when swaying to the port ($v < 0$) or starboard ($v > 0$). With the knowledge that the relationship between X and v is purely even, odd functions of v ($X_v, X_{vvv}, X_{uv} \dots$) can be omitted from (3.51). The same logic can be used to show that relationships between X and the $\dot{v}$, r, $\dot{r}$, δ_1 and δ_3 parameters are purely even as well, allowing a similar cancellation of odd terms.

A corresponding example can be observed by considering the relationship between the Y-force and v excitation of a port-starboard symmetric vehicle. If the vehicle sways to port ($v < 0$), it will induce some counteracting force to starboard ($Y > 0$). Conversely, if the vehicle sways with the same velocity to starboard ($v > 0$), it will experience a force to port ($Y < 0$). The relationship here is purely odd, like the function depicted in Figure 3.5b.

The relationship between X and the parameters w and q is more complex. These will in

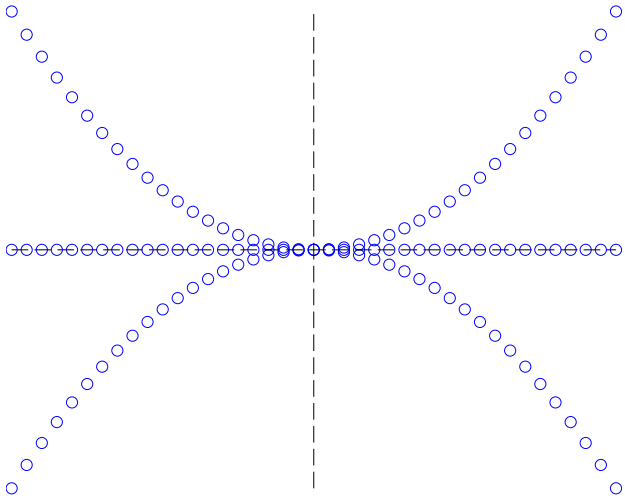


Figure 3.6: Possible relationships between X and v.

	u	v	w	p	q	r
X	mixed	even	mixed	even	mixed	even
Y	zero	odd	zero	odd	zero	odd
Z	mixed	even	mixed	even	mixed	even
K	zero	odd	zero	odd	zero	odd
M	mixed	even	mixed	even	mixed	even
N	zero	odd	zero	odd	zero	odd

Figure 3.7: Relationships between reactions (force and moments) and excitations (velocities and accelerations) for a vehicle such as the GPAUV with only port-starboard symmetry.

general return reactions like those characterized by Figure 3.5c and d. This can be deduced intuitively by realizing that flow towards the top of a vehicle will "see" a different surface than flow towards the bottom of a vehicle (assuming no top-bottom symmetry). These relationships are neither purely even or odd, but mixed.

By considering a typical AUV, such as the GPAUV, with only port-starboard symmetry, the relationship matrix shown in Figure 3.7 was developed. This matrix can be obtained by means of thought experiments, like those discussed above.

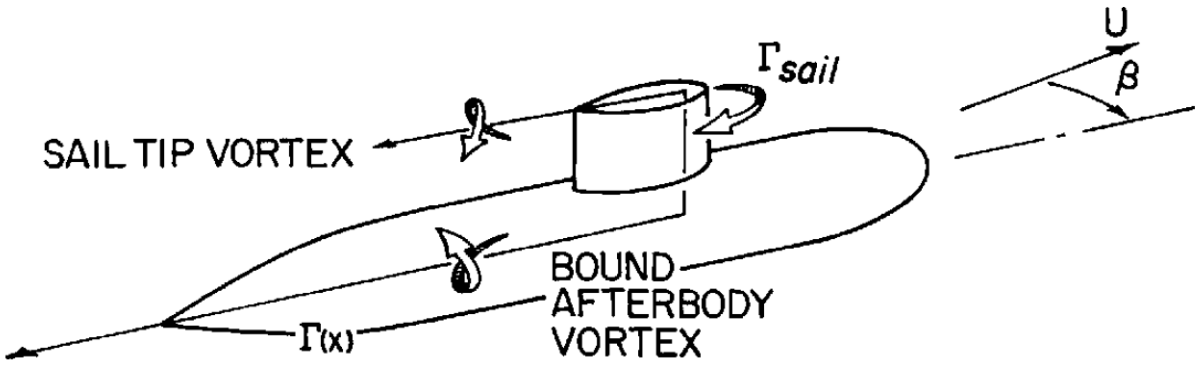


Figure 3.8: Diagram showing circulation, Γ, on submarine experiencing cross-flow [91].

3.3.4.2 Cross-Flow Damping

Experience in the modeling of submarine dynamics has shown that bound vortices emanating from the sail can create significant *out-of-plane* reactions. These reactions are out-of-plane in the sense that heave forces and pitching moments are created from sway and yaw excitations. Figure 3.8 depicts the major vortices that contribute to a nonzero circulation, Γ, for a typical submarine.

Following the explanation given by Mackay [91], the hull bound vortex, shown as $\Gamma(x)$ in Figure 3.8, has the largest contribution to out-of-plane reactions. Note that the variable x is used in this section to describe axial location. The Kutta-Joukowski theorem can be used to give a distribution of vertical force resulting from this vortex, $\hat{Z}(x)$.

$$\hat{Z}(x) = -\rho v(x)\,\Gamma(x) \tag{3.53}$$

The local sway velocity, $v(x)$, can be obtained by combining the body sway velocity, v, and the local sway velocity due to yaw rotation, rx.

$$v(x) = v + rx \tag{3.54}$$

Mackay links the local circulation, $\Gamma(x)$, to the circulation on the sail, Γ_{sail} by defining it as a function of downstream location, $f(x)$.

$$\Gamma(x) = f(x)\,\Gamma_{\text{sail}} \tag{3.55}$$

Likewise, the circulation on the sail can be defined by its lift, Y_{sail}.

$$Y_{\text{sail}} = \rho U \Gamma_{\text{sail}} b_{\text{sail}} \tag{3.56}$$

Here, b_{sail} and U are the span of the sail and body velocity magnitude respectively. These expressions allow for the total vertical force on the hull due to the sail-bound vortex to be written as

$$Z = \int \frac{-v\left(x\right) f\left(x\right) Y_{\text{sail}}}{b_{\text{sail}}} dx, \tag{3.57}$$

where the integral is taken from the trailing edge of the sail to the hull's aft perpendicular. A similar expression appears in Feldman's Revised Submarine Equations of Motion (Appendix A.1.3).

3.3.5 Propulsion

State-space models use a variety of propeller modeling approaches, but they are generally based on the propeller performance concepts discussed in Section 2.7.2.3. These sub-models approximate the thrust and torque output by the propeller based on shaft RPM and inflow velocity. As discussed in Section 2.7.3, the vehicles wake fraction and thrust deduction factor must also be determined and accounted for.

3.3.6 Control Input

An assumption is necessary to separate the effects of control surface deflections from the added mass and viscous damping characteristics of a vehicle. As a change in control surface deflection is actually a change in body geometry, that change must be assumed to be sufficiently small such that changes to the body's overall added mass and damping characteristics are small as well. In general, this is a fairly good assumption for current vehicle designs, which employ relatively small control surfaces located at the aft of the body, and therefore do not influence the flow over downstream surfaces.[1]

3.3.6.1 Lift Coefficient Methods

The method adopted in many models accounts for the effect of control surface deflections by means of a linear lift coefficient, $C_{L_\delta} = \frac{\partial C_L}{\partial \delta}$. By taking into account the location of the

[1] A completely rigorous state-space vehicle model would need to contain a complete representation of the vehicle's hydrodynamics (added mass and damping characteristics) for any possible combination of control surface deflections. In effect, every combination of control surface deflections would represent a unique vehicle. Again, this is unlikely to be necessary for traditional vehicle geometries. However, this type of an approach would likely be important if considering vehicles with large actuated control surfaces, such as seen in many biomimetic vehicles currently being considered by researchers [92–95].

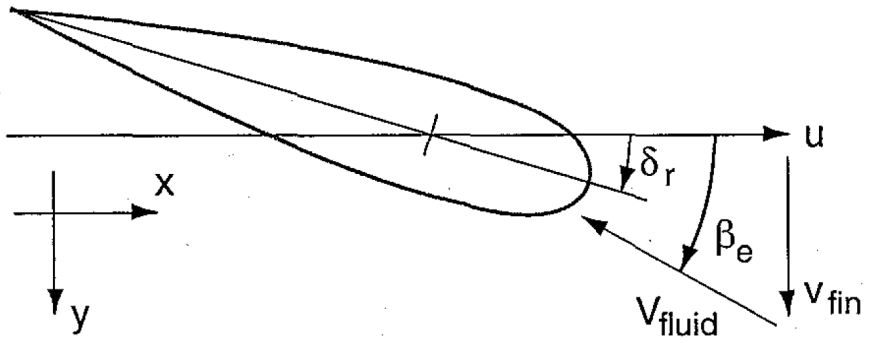


Figure 3.9: Effective angle of attack of a rudder [96].

control surface, this method is able to account for both forces and moments. For example, the sway and yaw reactions due to a rudder deflection could be given by

$$Y_{\text{rud}} = \frac{1}{2}\rho C_{L_\delta} S_{\text{rud}} \delta_{\text{rud}} \tag{3.58}$$

$$N_{\text{rud}} = \frac{1}{2}\rho C_{L_\delta} S_{\text{rud}} x_{\text{rud}} \delta_{\text{rud}}. \tag{3.59}$$

The terms S_{rud}, x_{rud} and δ_{rud} are the rudder's surface area, x-position and angle of deflection.

Often, in order to account for side-slip and rotational velocity of the vehicle, an effective angle of attack is substituted for the rudder's deflection (see Figure 3.9). If a control surface's axis of rotation is located at $r_{\text{rud}} = \begin{bmatrix} x_{\text{rud}} & y_{\text{rud}} & z_{\text{rud}} \end{bmatrix}^T$, the local flow velocity can be approximated by[1]

$$\begin{aligned} u_{\text{rud}} &= u + z_{\text{rud}} q - y_{\text{rud}} r \\ v_{\text{rud}} &= v + x_{\text{rud}} r - z_{\text{rud}} p \\ w_{\text{rud}} &= w + y_{\text{rud}} p - x_{\text{rud}} q. \end{aligned} \tag{3.60}$$

Based on this local velocity, the rudder's effective angle of attack can written as

[1]This assumes that the flow is undisturbed by the body upstream of the control surfaces. In reality, the flow is likely to become somewhat aligned with the hull of the vehicle, reducing the effect of side-slip and rotational velocity, thus resulting in an angle of attack somewhere between the commanded angle and that predicted by this calculation.

$$\begin{aligned} \delta_{e_{\text{rud}}} &= \delta_{\text{rud}} - \beta_{e_{\text{rud}}} \\ &\text{where } \beta_{e_{\text{rud}}} = \tan^{-1}\left(\frac{v_{\text{rud}}}{u_{\text{rud}}}\right). \end{aligned} \tag{3.61}$$

Similarly, for a dive-plane, the effective angle can be approximated by

$$\begin{aligned} \delta_{E_{\text{dive}}} &= \delta_{\text{dive}} + \beta_{e_{\text{dive}}} \\ &\text{where } \beta_{e_{\text{dive}}} = \tan^{-1}\left(\frac{w_{\text{dive}}}{u_{\text{dive}}}\right). \end{aligned} \tag{3.62}$$

Generally, the incidence angle, β_e, is considered to be small enough to approximate the tangent by a simple fraction.

3.3.6.2 Parametric Control Models

Another method of modeling the effect of control surfaces involves characterizing the reaction in each degree-of-freedom (X, Y, Z, K, M, N) for each control surface.

The effects due to the deflection of four control surfaces can be accounted for via a constant coefficient 6×4 matrix.

$$\vec{\tau}_c = \mathbf{T}_\delta \vec{\delta}_E = \begin{bmatrix} X_{\delta_{E_1}} & X_{\delta_{E_2}} & X_{\delta_{E_3}} & X_{\delta_{E_4}} \\ Y_{\delta_{E_1}} & Y_{\delta_{E_2}} & Y_{\delta_{E_3}} & Y_{\delta_{E_4}} \\ Z_{\delta_{E_1}} & Z_{\delta_{E_2}} & Z_{\delta_{E_3}} & Z_{\delta_{E_4}} \\ K_{\delta_{E_1}} & K_{\delta_{E_2}} & K_{\delta_{E_3}} & K_{\delta_{E_4}} \\ M_{\delta_{E_1}} & M_{\delta_{E_2}} & M_{\delta_{E_3}} & M_{\delta_{E_4}} \\ N_{\delta_{E_1}} & N_{\delta_{E_2}} & N_{\delta_{E_3}} & N_{\delta_{E_4}} \end{bmatrix} \begin{bmatrix} \delta_{E_1} \\ \delta_{E_2} \\ \delta_{E_3} \\ \delta_{E_4} \end{bmatrix} \tag{3.63}$$

The simulations and analysis discussed in Section 3.6.1 provides this information for the GPAUV.

3.4 Complete Models

The following sections provide a brief explanation of the state-space models (and styles of models) most commonly used to predict the maneuvering behavior of underwater vehicles. These models employ a mixture of the modeling approaches discussed in Section 3.3.

3.4.1 Component-Form Models

Both the Gertler-Hagen and Feldman equations introduced in Section 1.3.1 use a simplified version of the full added mass representation, (3.44-3.49), and a viscous damping formulation that combines cross-flow and Taylor series expansion damping. Many of the off-diagonal linear added mass terms (those other than $X_{\dot{u}}\dot{u}$, $Y_{\dot{v}}\dot{v}$ and $Z_{\dot{w}}\dot{w}$) are neglected. This omission has fairly good support in sensitivity analyses [89].

Nonlinear added mass effects (those in (3.44-3.49) that are multiplied by velocity squared) are accounted for using coefficients that include closely related viscous effects. This approach is based on the interaction of viscous and inviscid effects discussed in Section 3.3.3.3. This can be observed by comparing the second line of the Gertler-Hagen equation for surge motion, (A.3), and the corresponding inviscid equation, (3.44). The three terms from the Gertler-Hagen equation ($X_{qq}q^2$, $X_{rr}r^2$, $X_{rp}rp$) correspond to three of the terms in the inviscid equation ($Z_{\dot{q}}q^2$, $Y_{\dot{r}}r^2$, $Y_{\dot{p}}rp$). These terms represent similar phenomena, except that the Gertler-Hagen terms account for viscous effects in addition to the inviscid.

$$X_{qq}q^2 = Z_{\dot{q}}q^2 + \text{viscous effects} \tag{3.64}$$

In the inviscid equations, the value $Z_{\dot{q}}$ must be set so as to represent the Z-force due to an acceleration in q as well as the X-force due to the velocity q^2. While these two phenomena are directly linked in an inviscid fluid, this is not necessarily the case in a real fluid. This issue is also discussed in Section 3.3.3.4.

The Gertler-Hagen equations also include cross-flow terms to account for viscous damping, as discussed in Section 3.3.4. For instance, the second-to-last line of (A.10) accounts for sway forces created by the hull-bound vortex emanating from a submarine's sail (similar terms appear in the other equations as well, e.g., (A.11 - A.14)). In these equations, the integration bound x_2 is set at either the aft perpendicular or the some more forward point, based on whether or not the hull-bound vortex is believed to have separated from the hull at the current angle of drift (i. e., side-slip). The behavior of this bound with respect to vehicle side-slip must therefore be determined as part of the model population process.

3.4.2 Vector Models

Many present-day studies employ vectorized state-space models. The framework for writing these equations was popularized by Fossen [14–17]. Fossen's models employ the same basic nomenclature as the Gertler-Hagen equations, but do not necessarily include all the same terms. The six equations of motion for the vehicle can be written as

$$\underbrace{\mathbf{M}\dot{\vec{\nu}}}_{\text{inertial}} + \underbrace{\mathbf{C}\left(\vec{\nu}\right)\vec{\nu}}_{\substack{\text{Coriolis} \\ \text{-centripetal}}} + \underbrace{\mathbf{D}\left(\vec{\nu}\right)\vec{\nu}}_{\text{damping}} + \underbrace{\vec{g}\left(\vec{\eta}\right)}_{\substack{\text{gravity} \\ \text{\& buoyancy}}} + \underbrace{\vec{g}_0}_{\text{ballast}} = \underbrace{\vec{\tau}_c}_{\text{control}} + \underbrace{\vec{\tau}_{env}}_{\text{currents}} \tag{3.65}$$

Here, $\mathbf{M}$, $\mathbf{C}$, $\mathbf{D}$ are six-by-six matrices. The matrix $\mathbf{M}$ includes the contributions of rigid-body mass, $\mathbf{M}_{RB}$, as well as added mass, $\mathbf{M}_A$.

$$\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A \tag{3.66}$$

The added mass matrix uses the same form as defined in (3.36). The rigid-body matrix can be written as

$$\mathbf{M}_{RB} = \begin{bmatrix} m\mathbf{I}_{3\times 3} & -m\mathbf{S}\left(r_{CG}\right) \\ m\mathbf{S}\left(r_{CG}\right) & \mathbf{I}_{RB} \end{bmatrix} \tag{3.67}$$

Here, $\mathbf{I}_{3\times 3}$ is a 3×3 identity matrix The symbol $\mathbf{S}\left(\lambda\right)$ represents a skew-symmetric matrix for a vector $\lambda = \left[\lambda_1 \ \lambda_2 \ \lambda_3\right]^T$.

$$\mathbf{S}\left(\lambda\right) = -\mathbf{S}^T\left(\lambda\right) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix}. \tag{3.68}$$

The rigid-body inertial matrix in (3.67) is written as

$$\mathbf{I}_{RB} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{3.69}$$

Coriolis-centripetal effects, present as a result of the use of a body-fixed (non-inertial) reference frame, for the rigid-body, $\mathbf{C}_{RB}\left(\vec{\nu}\right)$, and added mass, $\mathbf{C}_A\left(\vec{\nu}\right)$, are encompassed by

$$\mathbf{C}\left(\vec{\nu}\right) = \mathbf{C}_{RB}\left(\vec{\nu}\right) + \mathbf{C}_A\left(\vec{\nu}\right). \tag{3.70}$$

These Coriolis-centripetal matrices are dependent on the rigid-body properties, added mass properties and instantaneous velocity of the vehicle. If the added mass matrix is given by

$$\mathbf{M}_A = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix}, \tag{3.71}$$

then the Coriolis-centripetal added mass matrix can be defined as

$$\mathbf{C}_A\left(\vec{\nu}\right) = \begin{bmatrix} \mathbf{0}_{3x3} & -\mathbf{S}\left(\mathbf{A}_{11}\vec{\nu}_1 + \mathbf{A}_{12}\vec{\nu}_2\right) \\ -\mathbf{S}\left(\mathbf{A}_{11}\vec{\nu}_1 + \mathbf{A}_{12}\vec{\nu}_2\right) & -\mathbf{S}\left(\mathbf{A}_{21}\vec{\nu}_1 + \mathbf{A}_{22}\vec{\nu}_2\right). \end{bmatrix}, \tag{3.72}$$

While the physics represented by (3.72) are accredited to Euler and Newton, this vector formulation is due to Fossen [14]. The combination of $\mathbf{M}_A\dot{\vec{\nu}}$ and $\mathbf{C}_A\left(\vec{\nu}\right)\vec{\nu}$ gives the ideal fluid equations (3.44-3.49). The rigid-body Coriolis-centripetal matrix, $\mathbf{C}_{RB}$, can be found using the same steps.

Forces and moments due to viscous damping effects are represented by $\mathbf{D}$. The damping matrix in (3.65) can take a variety of forms depending on the chosen modeling approach. A linear damping approximation would be represented by

$$\vec{\tau}_{\text{damping}} = \mathbf{D}\vec{\nu}, \tag{3.73}$$

with a damping matrix of the form

$$\mathbf{D} = \begin{bmatrix} X_u & X_v & X_w & X_p & X_q & X_r \\ Y_u & Y_v & Y_w & Y_p & Y_q & Y_r \\ Z_u & Z_v & Z_w & Z_p & Z_q & Z_r \\ K_u & K_v & K_w & K_p & K_q & K_r \\ M_u & M_v & M_w & M_p & M_q & M_r \\ N_u & N_v & N_w & N_p & N_q & N_r \end{bmatrix}. \tag{3.74}$$

Following the same notation used for added mass terms, the term X_v represents the viscous damping in the x-direction due to a unit of velocity in y-direction. As the viscous damping is well known to be a nonlinear phenomenon, this linear damping matrix is often augmented or replaced with a nonlinear damping matrix. A nonlinear formulation of the damping forces and moments might be obtained by

$$\vec{\tau}_{\text{damping}} = \mathbf{D}_{\vec{\nu}^2}\vec{\nu}^2. \tag{3.75}$$

A detailed discussion on modeling viscous damping is presented in Section 3.3.4.

The parameter $\vec{g}\left(\vec{\eta}\right)$ in (3.65) represents the hydrostatic balance discussed in Section 3.3.2. The $\vec{g}_0$ term is used for ballasting. The $\vec{\tau}$ terms on the right-hand side of (3.65) can include any additional components, such as control inputs (see Section 3.3.6) and currents.

3.5 Parameter Identification Methods

Quasi-steady state-space models can be populated via a number of approaches. The major groups of methods considered in this study are:

1. **Semi-Empirical Methods** - Semi-empirical methods, relying on a mixture of theoretical and empirical regressions, can be used to determine values for the constant coefficients that make up a state-space model.
2. **Captive Model Tests** - In a captive test approach, a vehicle is moved in a number of prescribed motions; the resulting forces and moments are measured and related back to those motions.
3. **System Identification (SI)** - In a system identification approach, a vehicle is monitored while performing a series of maneuvers. The data collected during these maneuvers is then used a create an optimization problem where a state-space model is fit to the collected data. The data used to inform an SI process can come from maneuvers of a prescribed style (i. e., captive model testing) or from fully free-running maneuvers.

Captive model testing and system identification methods have some areas of overlap. In this dissertation, the distinction is made that captive testing methods use specific tests to determine specific model coefficients, whereas SI methods allow the coefficients for a model to be optimized over an extended test or series of tests that excites the necessary degrees-of-freedom.

3.5.1 Semi-Empirical Database Model Population

As marine vehicles often tend to be of similar shapes (most submarines and AUVs have long slender cylindrical geometries with similar sails and control surfaces), the use of an empirical approach, in which coefficients are determined based on an interpolation from similar geometries, is quite common. As the GPAUV has a shape similar to other AUVs and submarines, this approach offers a good option for population of a state-space model.

A semi-empirical program designed to predict state-space model parameters for full-scale submarines was made available by Defense Research and Development Canada (DRDC) for use in this study [97–99]. The program, which is referred to as DRDC Submarine Simulation Program (DSSP), uses empirical database interpolation as well as analytic methods to provide the coefficients for state-space models. DSSP uses a description of geometry that includes the vehicle's hull, lifting surfaces (e. g., the sail) and "lump loads."[1] The hull itself is described

[1] DSSP's "lump loads" provide a means of including geometric features for which the hydrodynamic characteristics are already known. For example, an antenna could be included based on a circular cylinder's sectional drag coefficient. This function was not used in creating a model for the GPAUV.

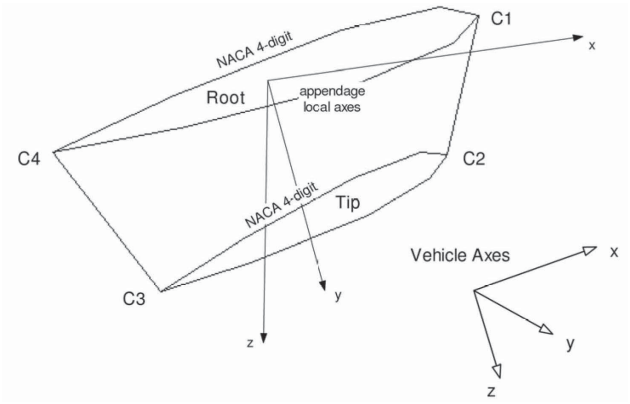


Figure 3.10: DSSP lift component input nomenclature [98].

by its overall length and the varying diameter along that length. Lift components are defined by four corner points, C1 through C4, as shown in Figure 3.10, and a thickness-over-chord ratio (ToC) to produce a NACA 4-digit airfoil shape.

3.5.2 Captive Model Testing

Captive model testing is generally considered to be the most effective means of populating a state-space model. As with any experimental approach, captive testing can also be quite expensive. By prescribing the motion of a model-scale vehicle in a controlled environment and measuring the forces and moments that it experiences, one can obtain an estimate for the coefficients in a state-space model.

3.5.2.1 Rotating-Arm Testing

Depictions of a rotating-arm facility and experimental set-up are shown in Figures 3.11 and 3.12. These apparatuses are used to measure rigid-body forces and moments due to rotational velocities (e. g., Y_r, M_q, ...). If the vehicle is mounted in a upright orientation (with z facing up or down) at a radius R, and is moving at a forward speed u, its yaw rate is given by

$$r = \frac{u}{R}. \tag{3.76}$$

The mounting radius and rotational speed of a rotating-arm can be varied to achieve a range of conditions. When testing underwater vehicles, tests can be carried out with the vehicle oriented on its side (z facing horizontally) to perform tests involving the pitching rate, q.

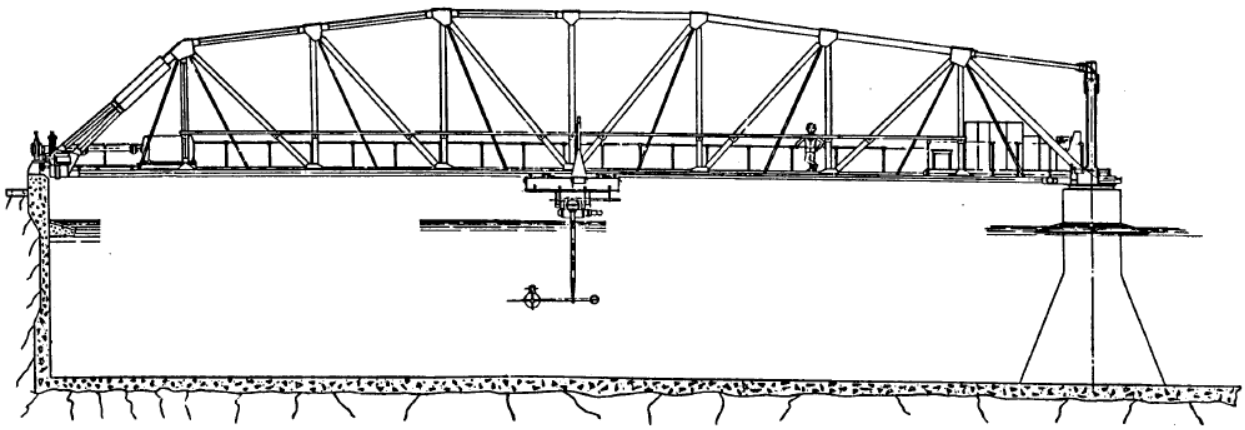


Figure 3.11: Illustration of rotating-arm testing facility [100].

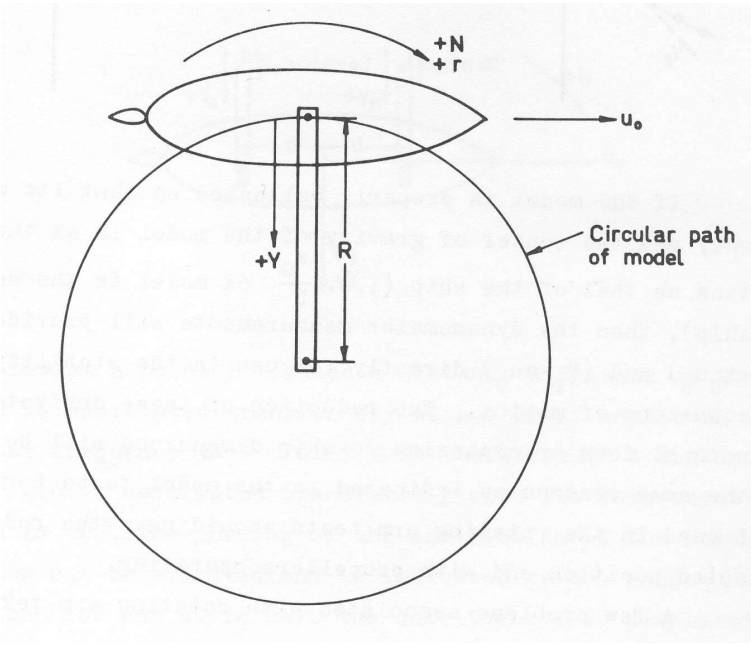


Figure 3.12: Diagram of rotating-arm testing method [11].

Significant limitations of a rotating-arm test, as noted by Crane et al. [90], are:

1. The need to complete testing within one revolution of the arm so as to avoid taking data while the model operates in its own wake.
2. The need for a large facility to allow for the testing of models at a large radius so that data can be taken at rotation velocities approaching zero.

Both of these issues can be averted by performing tests of a similar nature in CFD-based environments [33, 101, 102].

3.5.2.2 Planar Motion Mechanism (PMM) Testing

PMM testing, in which a vehicle is oscillated in a chosen degree-of-freedom while traveling at a constant forward speed to correspond to a standard operating condition, can be used to obtain maneuvering coefficients related to vehicle velocity and acceleration simultaneously. The forward speed and amplitude of oscillation are generally chosen to maintain dynamic similitude with a vehicle's expected or known behavior.

Oscillatory Motion Although PMM tests can be performed in any (axial or radial) degree-of-freedom of interest, consider for the sake of discussion a test in the heave direction, z, of a vehicle with length L. As the vehicle translates at a constant forward speed, U, a sinusoidal heave oscillation with given nondimensional amplitude, $a'_0 = \frac{a_0}{L}$, and frequency, $\omega'_0 = \frac{\omega L}{U}$, can be performed. The resulting nondimensional position, z', velocity, w', and acceleration, $\dot{w}'$, in heave are

$$\begin{aligned} z' &= a'_0 sin\left(\omega' t\right) \\ w' &= \dot{z}' = a'_0 \omega' cos\left(\omega' t'\right) \\ \dot{w}' &= \ddot{z}' = -a'_0 {\omega'}^2 cos\left(\omega' t'\right), \end{aligned} \tag{3.77}$$

with time nondimensionalized as $t' = \frac{tU}{L}$ and

PMM Parameter Identification The fluid force acting on the body as a result of its motion can be decomposed in a number of ways. In the earliest PMM tests, Goodman [5] used a linearized model where the total force, Z', is considered to be composed of a damping component, Z_w, and an added mass component, $Z'_{\dot{w}}$, which are related to velocity, w', and acceleration, $\dot{w}'$, respectively.

$$Z' = \dot{w}'\left(Z_{\dot{w}} - m'\right) + Z'_w w' \tag{3.78}$$

The total force can be separated into components proportional to the velocity and acceleration using a 1st-order Fourier analysis or any preferred optimization algorithm.

Alternatively, higher order representations can be applied. Morison's equation is appealing in that it uses a quadratic drag force term [10].

$$Z' = \dot{w}' \left({Z_{\dot{w}}}' - m \right) + {Z_{w|w|}}' w' \left| w' \right| \tag{3.79}$$

Abkowitz also introduced a popular viscous damping decomposition model based on a truncated Taylor series expansion [11].

$$Z' = \dot{w}' \left({Z_{\dot{w}}}' - m \right) + {Z_{w}}' w' + {Z_{w^3}}' w'^3 \tag{3.80}$$

Optimization algorithms can be used to decompose a total force history into the Morison and Abkowitz style models.

Virtual PMM Testing While PMM testing has traditionally been performed in tow tanks with scaled models and prototypes, the current state of computing power allows for tests to be carried out virtually in numerical simulations [103–106]. Additionally CFD-based testing has a number of advantages over traditional tow tank testing. Unless one is attempting to replicate the results from a tow tank test, simulations may be carried out without the presence of adjacent walls. This eliminates effects such as blockage and tank resonance. Traditional experimentation also relies on mechanical devices to execute the vehicle motions and make measurements of the reactions. The limitations of experimental mechanisms (i.e. actuators and strain gauges) may restrict the oscillation frequency and amplitude which can effectively be produced and measured during a test [107]. As with most numerical testing, virtual PMM tests also have a definite cost advantage over traditional tests.

PMM Validation and Exploration To provide a general demonstration, a 6:1 (fineness ratio) prolate spheroid of length $L = 2\,\text{m}$ was analyzed [108, 109]. A closed form solution for the added mass of a prolate spheroid, based the on potential flow theory formulation discussed in Section 3.3.3, is used as a comparison point for the results obtained in this analysis. While the viscous simulations should not be expected to precisely match this inviscid value, it can still serve as an instructive point of comparison. Tests were performed at a Reynolds number of $4.5{\times}10^6$ to correspond with the operating condition of the GPAUV.

The solution domain, containing 1.09×10^6 finite volume cells, was bounded by a single symmetry plane (shown in Figure 3.13) and a downstream pressure outlet, with the remaining faces designated as velocity inlets acting to passively produce the desired oscillatory motion of the body (see Section 2.8 for a discussion of this method). The details of the simulation and mesh are directly based on those assessed in Section 2.10.2.3 and tabulated in Table 2.4.

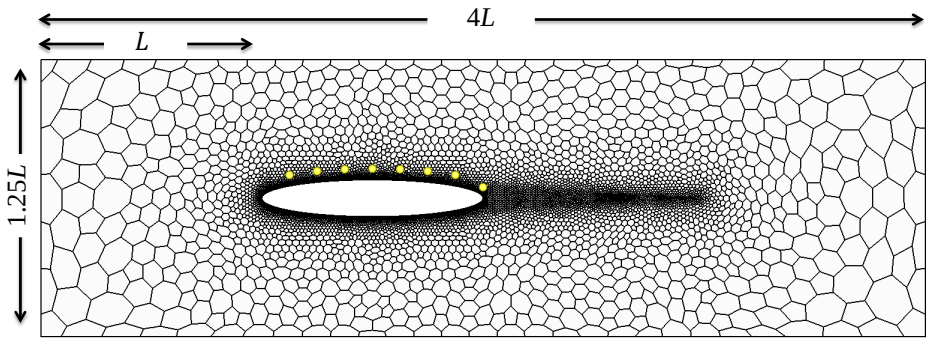


Figure 3.13: Computational domain (shown on symmetry plane) for spheroid PMM tests with fluid velocity probe points (yellow).

Table 3.1: Spheroid added mass predictions.

Amplitude, a'_0	Added Mass, $Z'_{\dot{w}}$			
	Theoretical	Linear	Morison	Abkowitz
0.005	-0.02668	-0.02663	-0.02664	-0.02663
0.01	-0.02668	-0.02657	-0.02657	-0.02657
0.02	-0.02668	-0.02661	-0.02661	-0.02661
0.025	-0.02668	-0.02661	-0.02661	-0.02661
0.038	-0.02668	-0.02652	-0.02652	-0.02652
0.05	-0.02668	-0.02633	-0.02633	-0.02633
0.1	-0.02668	-0.02445	-0.02445	-0.02445
0.2	-0.02668	-0.01936	-0.01936	-0.01936
0.25	-0.02668	-0.01809	-0.01809	-0.01809
0.5	-0.02668	-0.01587	-0.01587	-0.01587
1	-0.02668	-0.01196	-0.01196	-0.01196

Each virtual test was performed for three cycles of oscillation, with 200 time-steps per oscillation. To reduce the presence of any transient behavior, only the 3$^{\text{rd}}$ oscillation was post-processed for parameter identification. In addition to recording the integrated pressure and shear forces acting on the body during oscillation, fluid velocity was recorded at points distributed along the length of the body at a vertical distance of $0.05L$ above the surface of the body (these points are shown as yellow circles in Figure 3.13).

Each damping model, (3.78, 3.79, 3.80), was evaluated against the forces returned from the PMM tests run at a nondimensional frequency of 5 with nondimensional amplitudes ranging from 0.005 to 1. Table 3.1 shows the added mass and damping coefficients determined by each model.[1]

[1]The damping coefficients from each model are not directly comparable due to their different formulations.

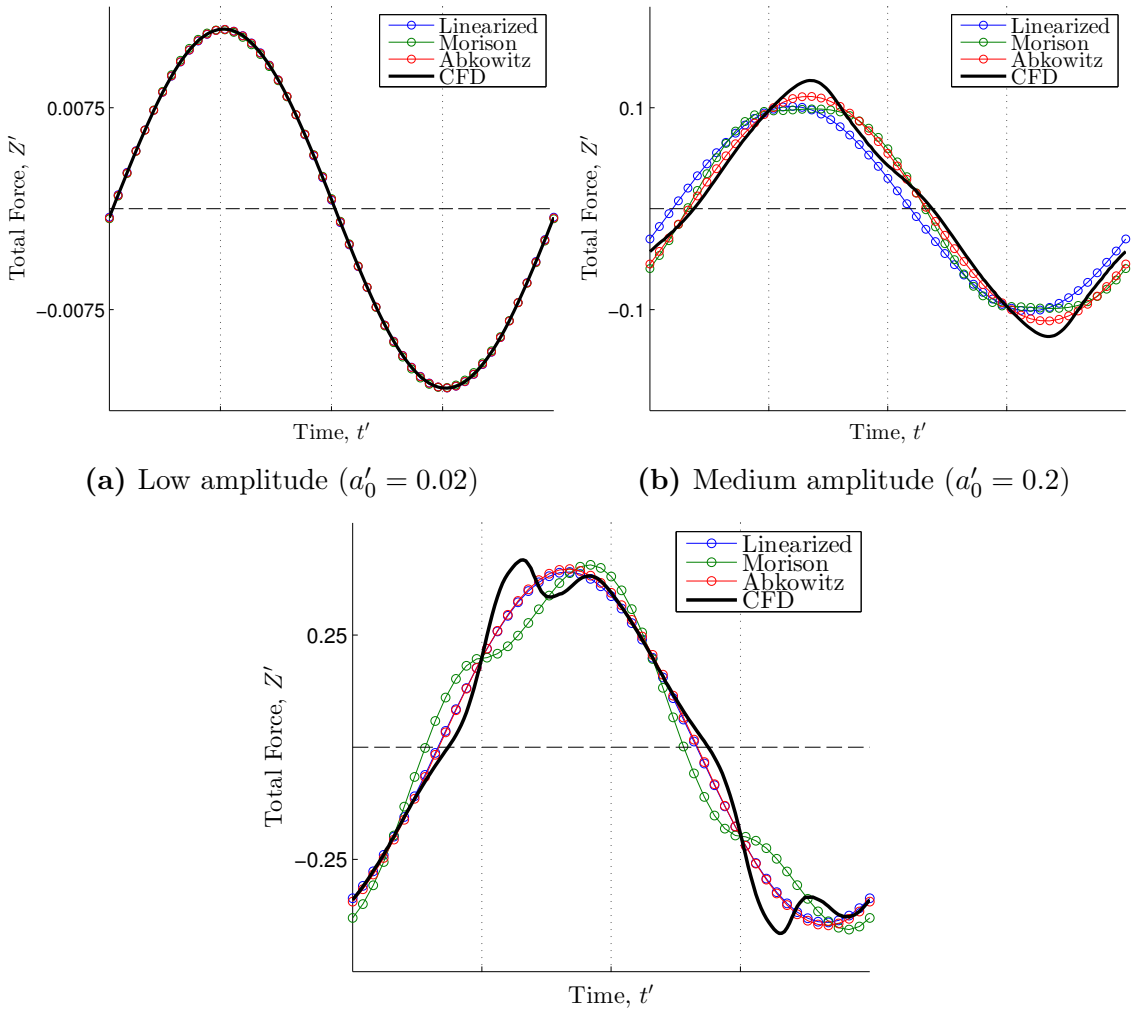
Table 3.2: Spheroid damping predictions.

Amplitude, a'_0	Damping, Z'_w			
	Linear, Z'_w	Morison, ${Z_{w\|w\|}}'$	Abkowitz, ${Z_w}'$	Abkowitz, ${Z_{w^3}}'$
0.005	-0.00387	-0.32656	-0.0041	-5.58239
0.01	-0.00442	-0.14295	-0.00448	-0.88844
0.02	-0.00636	-0.0746	-0.00552	-0.1372
0.025	-0.00671	-0.0599	-0.00565	-0.07903
0.038	-0.00693	-0.04003	-0.00573	-0.03376
0.05	-0.00636	-0.03047	-0.00655	-0.00474
0.1	-0.01067	-0.03022	-0.01365	-0.00014
0.2	-0.02997	-0.05936	-0.05483	0
0.25	-0.05809	-0.06422	-0.07278	0
0.5	-0.1346	-0.06088	-0.13758	0
1	-0.2879	-0.05761	-0.22103	-0.00171

Model Comparison The degree to which a force decomposition model captures the force predicted by a simulation can be assessed by comparing the sum of the models' components to the original force. A model that truly captures all physical phenomena in the flow would match the force from the simulation exactly. Figure 3.14 shows the forces reconstructed from each model along with the original simulation force (titled 'CFD' in the legend) for one cycle of oscillation. It can be seen here that each model performs very well (and almost identically) in low amplitude tests. The models are able to match the force obtained from the simulation with very little error. Distinct differences between the models and the force on the body from the simulation can be seen in higher amplitude tests. None of the applied models do particularly well at capturing the shape of the force response in the high amplitude test (Figure 3.14c).

Each test can be examined more closely by examining the constituent components individually. Figure 3.15 shows the added mass and damping components of each force decomposition model for the medium amplitude test. As would be expected based on the similarity in the formulation of each model, the added mass predictions of each model are nearly identical. The predicted viscous damping forces (Figure 3.15b) vary more widely.

Variation of Model Accuracy with Amplitude Figure 3.16 shows the relative error between each model and the simulated total force across the range of tested oscillation amplitudes. In general, the Abkowitz model was able to match the force obtained from simulations with the least error. This, however, may stem more from the optimization advantage of having one extra parameter with which to "fit the curve," rather than being a more accurate representation of the physics. The results in Figure 3.16 show a distinct increase in the growth rate of the error in each model above nondimensional amplitudes of about 0.05.

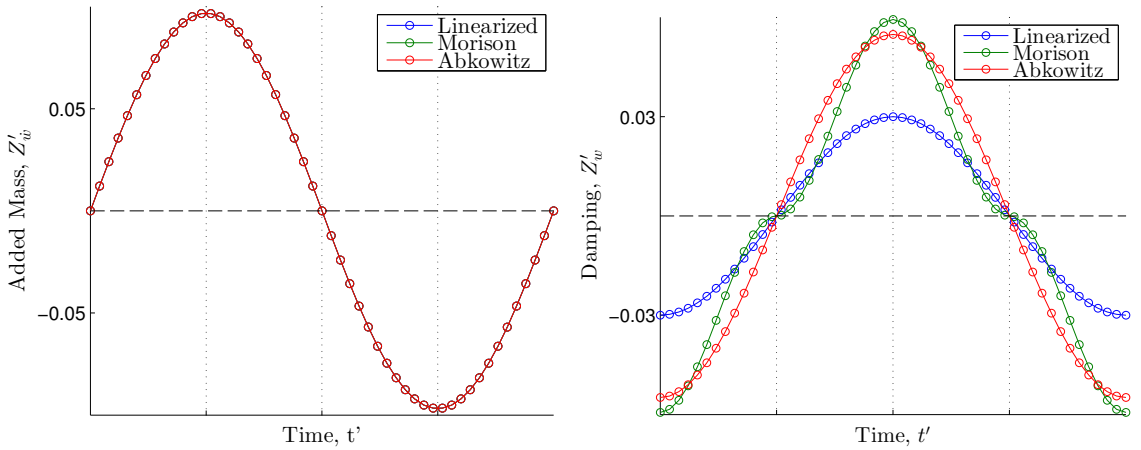


(a) Low amplitude ($a'_0 = 0.02$)

(b) Medium amplitude ($a'_0 = 0.2$)

(c) High amplitude ($a'_0 = 0.5$)

Figure 3.14: Simulation and model force matching comparison of PMM tests using a range of oscillation amplitudes.



(a) Added mass **(b)** Viscous damping

Figure 3.15: Medium oscillation amplitude ($a'_0 = 0.2$) test model predictions.

Figure 3.17 shows the error of the Abkowitz model over one cycle of oscillation at a range of oscillation amplitudes (with higher amplitudes shown closest to the viewer for clarity). This figure is representative of the other models, in that a high frequency discrepancy (larger than ω') between the model and the force obtained from the simulation increases rapidly with oscillation amplitude for $a'_0 > 0.05$.

However, close analysis of Figure 3.17 shows that the structure of the error does not grow continuously with amplitude, but appears to have at least two stages. Peaks in the error at medium amplitudes do not necessarily continue to grow at higher amplitudes, but instead may begin to weaken or become reversed in sign as amplitude grows. This multistage development is also visible in Figure 3.14.

Unmodeled Heterogeneous Flow Each of the examined models are quasi-steady in the sense that they implicitly assume that the nature of the flow field instantaneously tracks the state of the body. The models treat parameters related to velocity and acceleration as if those quantities were instantaneously constant, i.e., the drag force is modeled as if the flow field around the body instantaneously tracked the changes in the body velocity and acceleration. Figure 3.18 shows the body and near-field flow velocity in the z-direction during low, medium and high amplitude oscillation tests (near-field flow velocities were measured at the probe points shown in Figure 3.13). In low amplitude tests (Figure 3.18a), the phases of the nearly sinusoidal local flow velocity variations closely match that of the body (with the exception of the aft most point). At higher amplitudes of oscillation (Figures 3.18b and 3.18c), inertial and viscous effects cause the fluid velocities to diverge from the sinusoidal oscillation of the body (sinusoidal fits have been omitted here in the interest of clarity). The asymmetric nature of this oscillatory pattern is due to the windward-leeward location of the probes during each phase of oscillation.

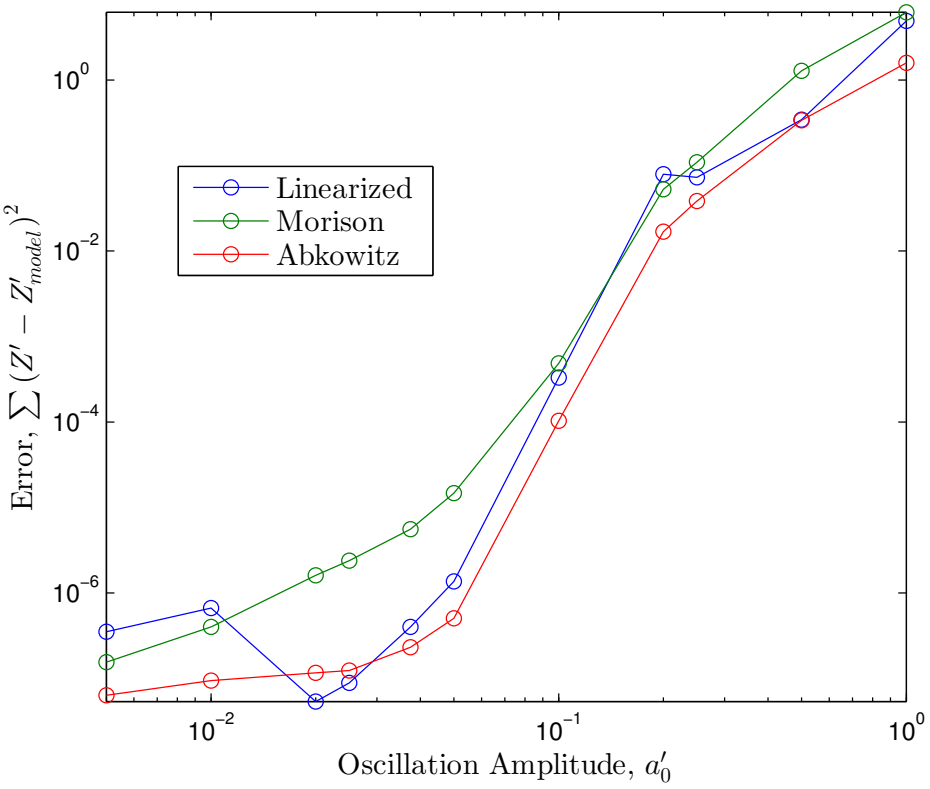


Figure 3.16: Error of force response modeling at a range of oscillation amplitudes.

Between these examples, the flow has undergone a drastic change in character. During low amplitude cases, the flow generally remains attached to the body, while at higher amplitudes flow separation and vortex shedding create a highly irregular response. While this change is by no means surprising, it is nonetheless ignored by the force decomposition models examined in this study.

Addressing this change in flow behavior is an enticing task, but devising a universal model may not be possible. The nature with which flow response and separation occur are so intimately defined by body geometry, that any model could only perform well for a small set of similar vehicles undergoing similar motions.

Assessment of PMM and Model Validity Planar motion mechanism tests can provide accurate vehicle performance characteristic data for use in quasi-steady state-space models. The force decomposition models examined in this study provided relatively similar results; no single model appears to outperform the others for the case studied. The analysis of PMM tests across a range of motion amplitudes suggests that tests should be executed at nondimensional amplitudes less than 0.05 to insure reasonable accuracy of the derived coefficients in the sense that traditional force models using those coefficients reasonably match the observed

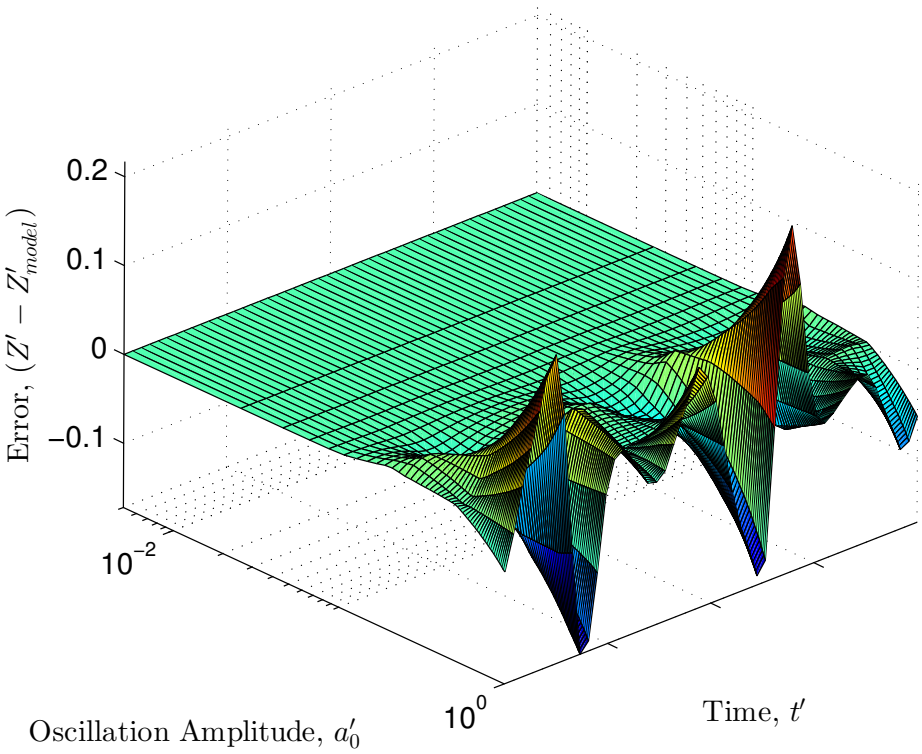


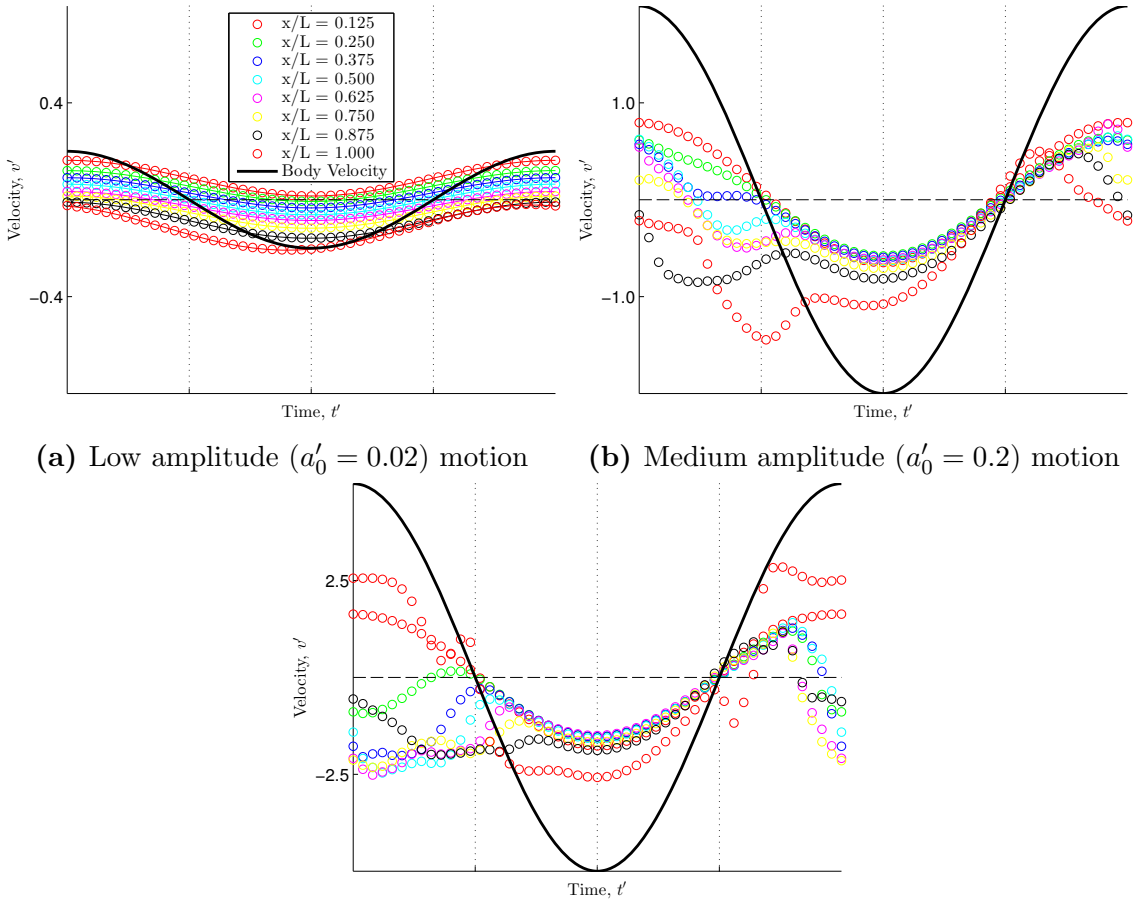
Figure 3.17: Local (temporal) error of Abkowitz model at a range of oscillation amplitudes.

hydrodynamic forces. These results highlight the short-comings of state-space models when concerned with such a vehicle maneuver.

3.5.3 System Identification Approach

A system identification (SI) approach to model population has been used by a number of researchers for both model and full-scale vehicles [110–114]. In such an approach, an optimization method is used to minimize the error between experimental data and a state-space model's prediction. By allowing the optimization algorithm to manipulate a model's constant coefficients, it can be fit to the experimental data.

An SI approach can be applied in a number of ways. The first issue at hand is the nature of vehicle maneuvers on which the optimization is performed. Often, SI is done using a free-running maneuver in which the vehicle operates under its own power and control. The vehicle's state $(\eta, \nu, \dot{\nu})$ can be compared against that predicted by a model. This version of an SI approach is particularly useful for developing a state-space model when the only method of experimental measurement involves a full scale vehicle. Another option, explored in this study, is to use the forces and moments experienced by the vehicle during captive motions



(a) Low amplitude ($a'_0 = 0.02$) motion

(b) Medium amplitude ($a'_0 = 0.2$) motion

(c) High amplitude ($a'_0 = 0.5$) motion

Figure 3.18: Spheroid body (black line) and near-field flow (colored markers) vertical velocity during PMM tests of varying amplitude. (Legend shown in (a) applies to (b) and (c) as well.)

as the input for the SI process. This process is similar to the captive model test methods discussed in Section 3.5.2, but allows each coefficient to be defined using information from multiple tests.[1]

3.5.3.1 Optimization Approach

A constrained minimization of an objective function was used to obtain the coefficients for state-space models in this study.

[1] Considering the added mass equations (3.44-3.49) for an example, one can see that the term $X_{\dot{w}}$ appears not only in the X-force equation, but in the other five force and moment equations as well.

Objective Function The calculation of this objective function involves the summation of errors across three levels: (1) the six forces and moments that the model is designed to predict, $\vec{\tau}$, (2) the n "tests" required to excite all necessary degrees-of-freedom and (3) the k discrete time-steps in each of those tests. The time-dependent error in the i^{th} test can be defined by taking the difference between the current model prediction, $\vec{\tau}^{\,\text{model}}$, and the results from the CFD simulation, $\vec{\tau}^{\,\text{in}}$.

$$\gamma_i(t) = \left|\left|\vec{\tau}_i^{\,\text{model}}(t) - \vec{\tau}_i^{\,\text{in}}(t)\right|\right| \tag{3.81}$$

As the l^2 norm is taken, $\gamma_i(t)$ represents the error in all six forces and moments for a single time-step in a single test. The total error in each test can then be found by summing over all values of t

$$\underline{\gamma}_i = \sum_{t=0}^{k} \gamma_i(t)\,. \tag{3.82}$$

The norm of the vector, $\vec{\underline{\gamma}}$, can then be taken to give the objective function.

$$\Gamma = \left|\left|\vec{\underline{\gamma}}\right|\right|. \tag{3.83}$$

Constraints and Initial Conditions Based on the symmetry of the GPAUV, the added mass matrix was constrained to be of the sparse (and symmetric) form shown in (3.29). Initial experimentation in this process using a prolate spheroid showed a slight tendency of the identified added mass matrix to stray from the theoretical when high amplitude motions were used to inform the optimization.[1] Based on this observation, an added mass matrix obtained from a vortex panel code was used to constrain the added mass matrix during the optimization process [115]. Thus the optimizer was only allowed to vary the damping coefficients.

Tests to Inform the Optimization A series of six virtual PMM tests (one in each degree-of-freedom) were conducted to inform the SI process. The oscillation amplitude and frequency in each test was chosen so as to match typical velocities and accelerations experienced by the GPAUV. The these parameters were considered in reference to the vehicle's forward speed to insure a scenario in which the vehicle interaction with wake formed by its oscillatory motion was minimal.

[1] This is similar to the phenomena discussed in Section 3.5.2.2 and shown in Table 3.1.

Numerical Implementation Matlab's constrained optimization function, *fmincon*, was used to minimize the objective function, Γ. The optimization algorithm was partially parallelized to allow for full models to be obtained from previously completed CFD results in approximately 2 minutes on Breaker (see Appendix C). A listing of the scripts used in this process is provided in Appendix A.4.

3.6 State-Space Models for the GPAUV

Three state-space models to predict the maneuvering of the GPAUV were created using the methods discussed in Chapter 3. They are compared against each other, experimental testing data and VFRM simulations. The details of each model are given in Table 3.3. The coefficients that make up the models are provided in Appendix A.2.

3.6.1 Control Surface Sub-model

All of the state-space models of the GPAUV considered for this study are of a parametric form, as described in Section 3.3.6.2. Steady-state simulations, in which the GPAUV's control surfaces were deflected throughout their operating ranges, served as the basis for this model. In each simulation, a uniform freestream velocity of 2 m/s was imposed to match the vehicle's nominal operating condition. The vehicle was assumed to be aligned with the oncoming

Table 3.3: GPAUV state-space models.

	DSSP Model	**Linear-Damping Model**	**Nonlinear-Damping Model**
Population method	Semi-empirical	System identification	
Added mass	Full nonlinear with viscous corrections	Full nonlinear inviscid	
Viscous damping	Gertler-Hagen (nonlinear)	Linear	Taylor series expansion
Control input	Parametric		

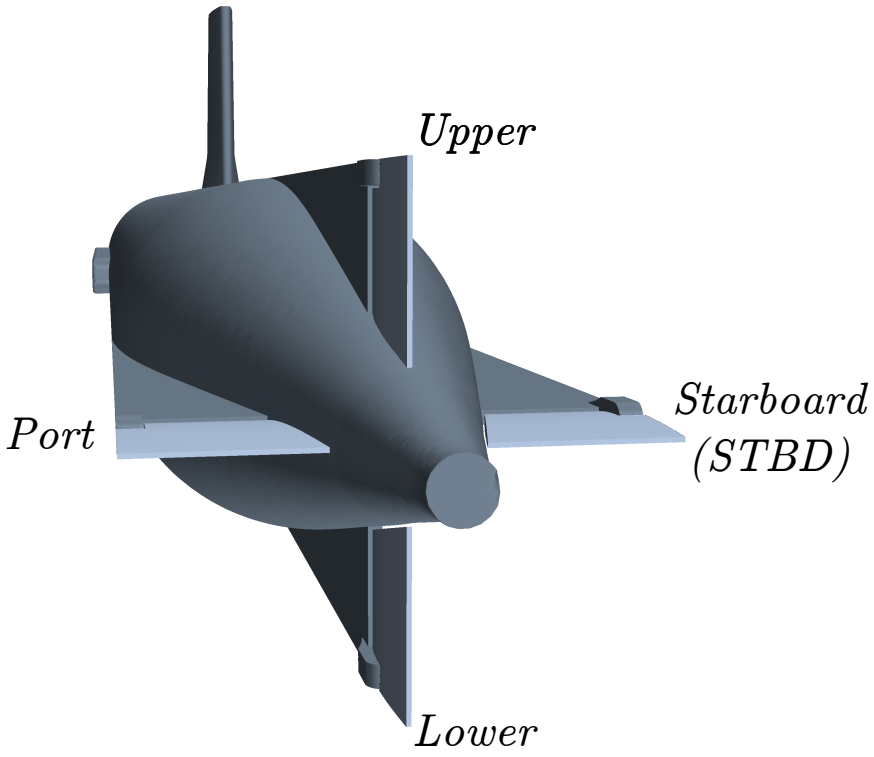


Figure 3.19: Nomenclature for the GPAUV's control surfaces.

flow; side-slip effects were not considered. The coefficients for this model are provided in Appendix A.2.1. The nomenclature convention for the GPAUV's control surfaces is shown in Figure 3.19. This series of simulations led to questions about the relative importance of accounting for some of the complexities inherent to control surface performance [116].

3.6.1.1 Control Surface Asymmetry

Motivation Wakes propagating downstream from the GPAUV's sail (mounted on top of the vehicle at $\frac{x}{L} \approx 0.65$) and external sensor appendages (mounted on the bottom of the vehicle at $\frac{x}{L} \approx 0.25$) provide distinct inflow conditions for each control surface ($\frac{x}{L} \approx 0.95$) respectively. The resulting difference in pressure induced by a 20° deflection of the upper and lower rudders can be seen in Figure 3.20, where the pressure coefficient, C_p, has been plotted on the surface of the AUV's tail, stationary strakes and control surfaces.

$$C_p = \frac{p}{\frac{1}{2}\rho U^2} \tag{3.84}$$

From Figure 3.20, it is clear that the upper rudder, as well as the adjacent hull and strake, experiences a greater pressure than the lower rudder and its surrounding surfaces.

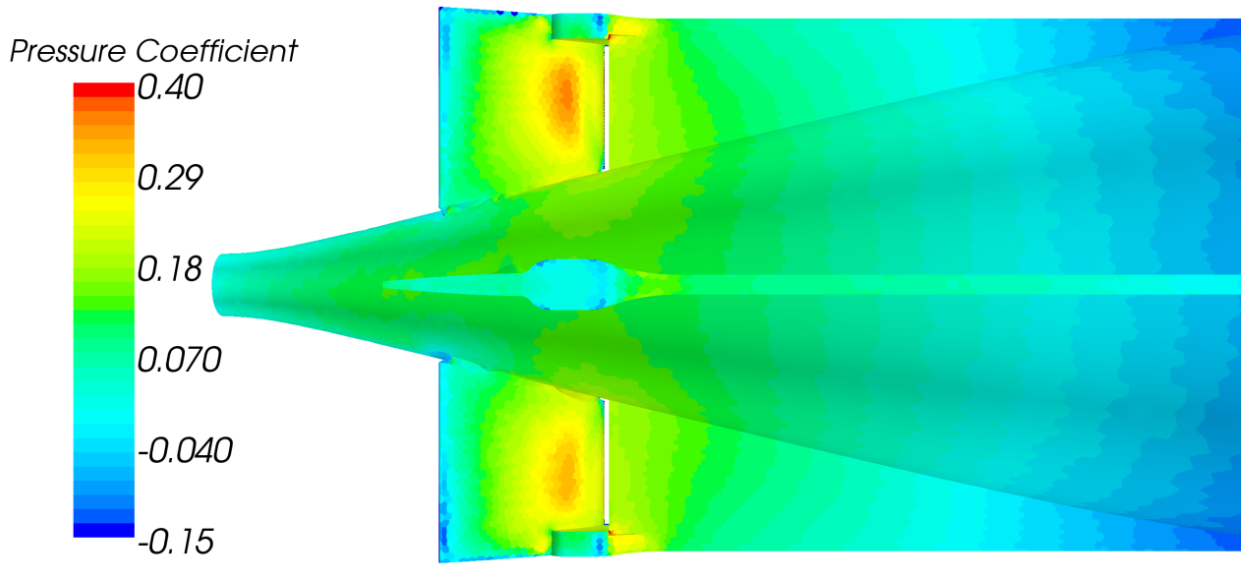


Figure 3.20: Upper and lower rudder (pressure side) pressure coefficient when each is deflected to 20° while the GPAUV is moving forward at 2 m/s.

Results As a result of these asymmetric conditions, the upper and lower rudders have dissimilar performances, as shown in Figure 3.21. The maximum measured discrepancy occurs at a deflection angle of 20°, where the lower rudder produces 7.4% less sway force and 10.4% less yaw moment than the upper rudder.

These discrepancies appear to occur largely due to a difference in the stall characteristics of each control surface, with the lower rudder stalling slightly earlier (and less abruptly) than the upper rudder. This difference in separation can be seen in Figure 3.22, which shows the flow field near the upper and lower rudders when deflected to 20°. At this same deflection, the lower rudder experiences significant flow separation (shown by the low speed flow on the suction surface), while the upper rudder experiences relatively little. This is likely due to disparities in the two inflows (flow speed, direction and turbulence) for each control surface.

While this analysis shows that the total sway and yaw moment supplied by the two rudders may differ slightly from a more simplistic prediction (i. e., using the same performance characteristics for each control surface), the creation of a small but unexpected secondary moment may be more critical. With both rudders equally deflected, an AUV would be expected to experience substantially increased drag, sway and yaw reactions, but the asymmetric nature of the true system also results in secondary moments in pitch and roll. With both rudders deflected to 20°, the GPAUV experiences secondary pitching and rolling moments, with magnitudes 6.5% and 2.5% the magnitude of the primary yawing moment respectively. The secondary rolling moment, shown over the range of examined control surface deflections in Figure 3.23, peaks at nearly 2.5 times the magnitude of the torque created by the GPAUV's propeller at this speed.

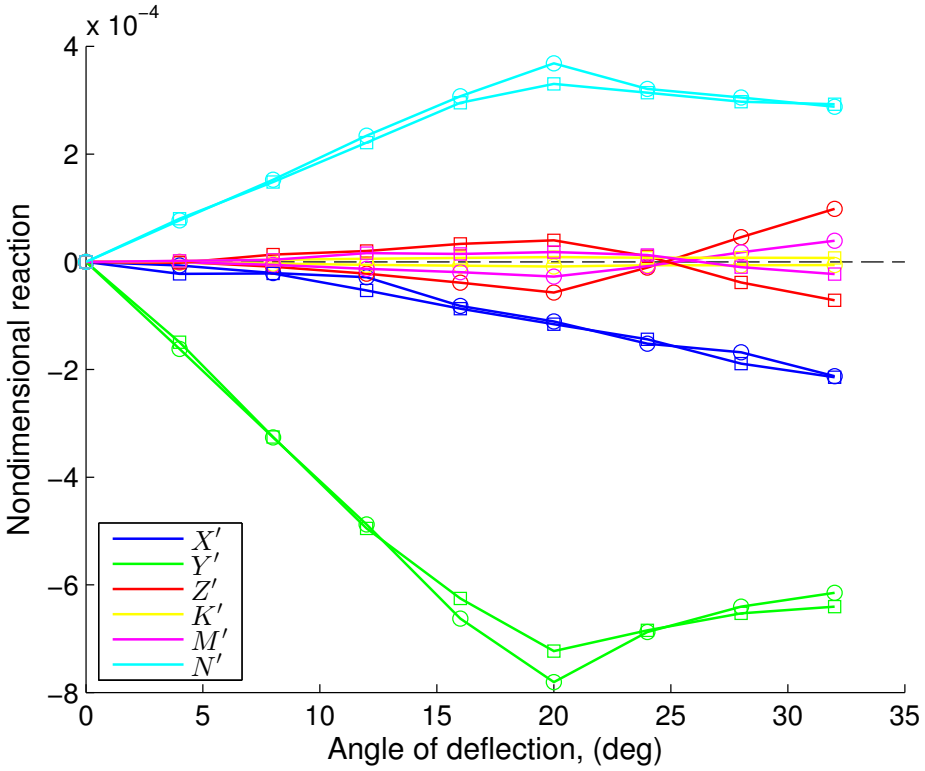


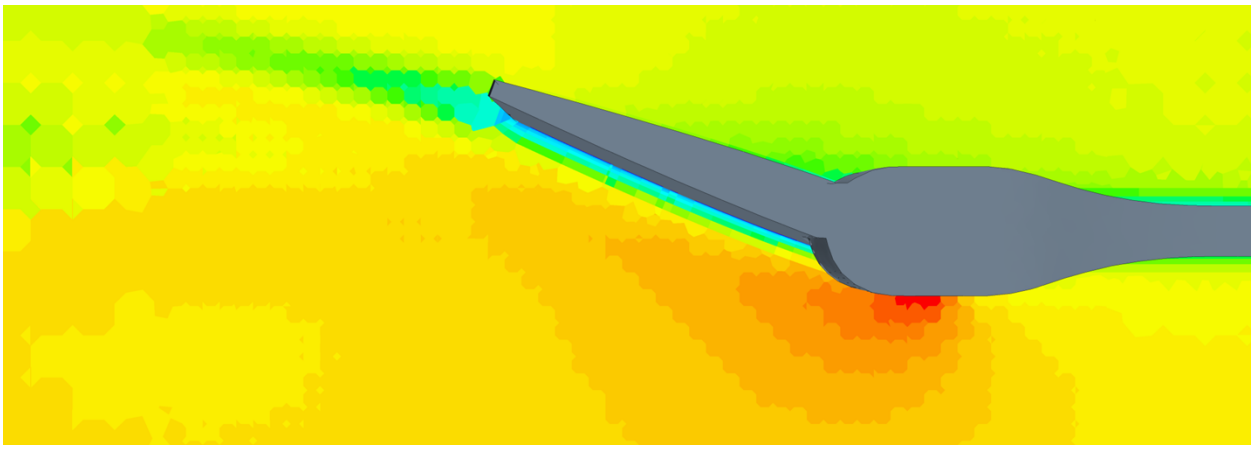
Figure 3.21: Upper (circle) and lower (square) rudder induced forces and moments.

3.6.1.2 Propeller effects

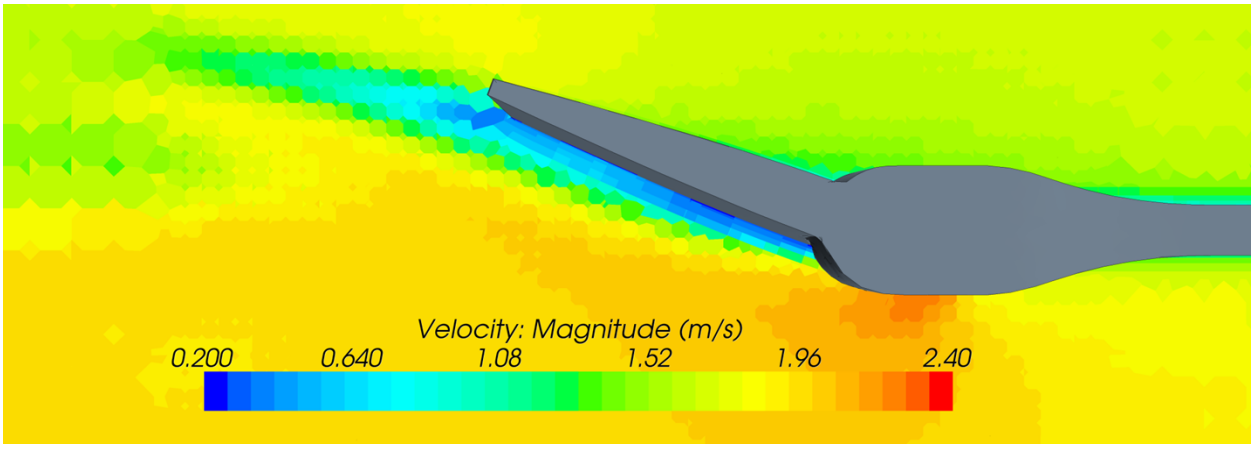
Motivation As discussed in Section 2.7.1, an operating propeller induces an increased velocity and decreased pressure in upstream flow. The potential for a propeller to affect adjacent control surfaces is well known and included in some state-space models [117]. As the GPAUV is configured with its control surfaces located directly upstream of the propeller, it follows that some effect on the performance of the control surfaces is likely. A study comparing the performance of the GPAUV's control surfaces with and without the actuator disk in operation was conducted to better understand this effect.

Results Figure 3.24 shows the flow field near the upper rudder of the GPAUV with and without propeller operation. Noticeable flow separation, indicative of airfoil stall, is visible on the suction side of the rudder without the propeller in operation (see Figure 3.24b).

The results of such differences in flow conditions are shown in Figure 3.25, which plots the forces and moments generated by the vehicle's upper rudder with and without the propeller in operation. The presence of the propeller alters (generally increases) the forces and moments created by the upper rudder by approximately 5% throughout the range of deflections. This



(a) Upper rudder



(b) Lower rudder

Figure 3.22: Velocity magnitude near upper and lower rudders when deflected to 20° (upper rudder image has been reversed horizontally to aid comparison; color bar in (b) applies to both figures).

difference appears to be due to the propeller's ability to increase the flow velocity over the rudder and reduce flow separation on the suction side of the deflected control surface.

3.6.2 DSSP Model

A maneuvering model for the GPAUV was created using DSSP, as described in Section 3.5.1. The formulation employed by this model is nearly identical to the Gertler-Hagen equations shown in Appendix A.1.2. DSSP's geometric representation of the GPAUV is shown in Figure 3.26. Twenty-one stations were used to define the GPAUV's hull (the is the maximum number of stations accepted by DSSP). All appendages of the GPAUV are defined as lifting

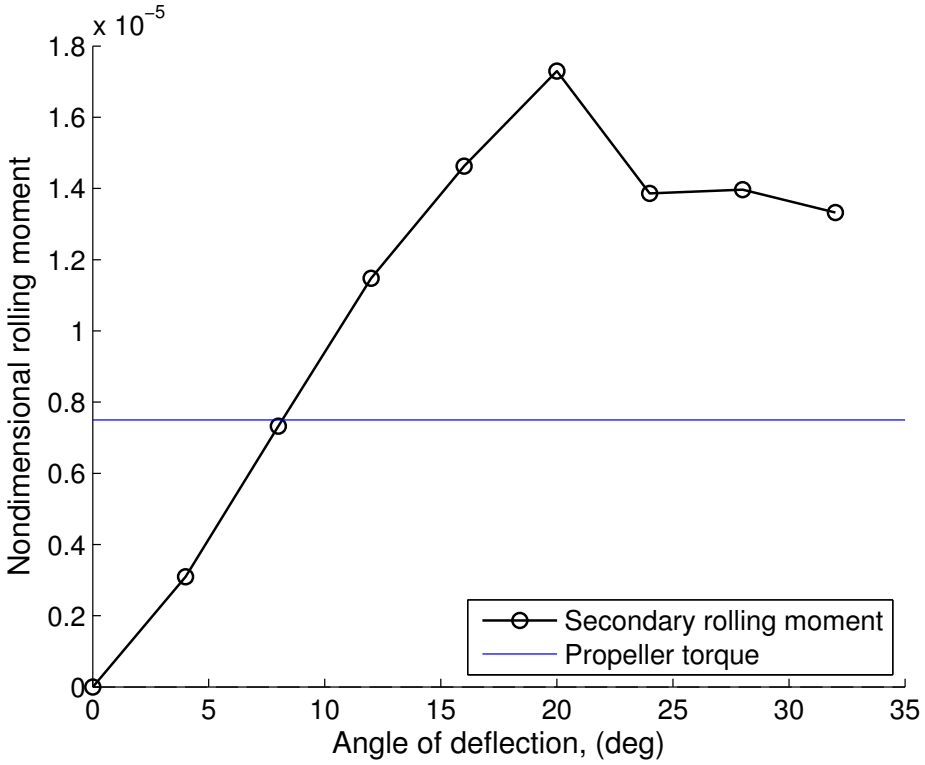


Figure 3.23: Secondary rolling moment created by equal deflection of both rudders, shown with torque produced by the vehicles propeller when operating at a forward speed of 2 m/s.

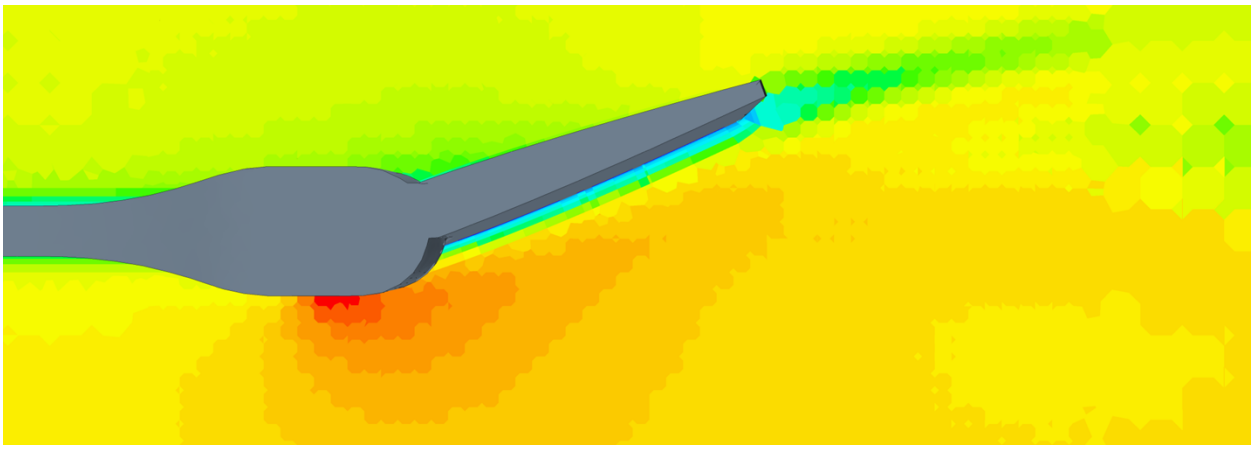
surfaces. The coefficients produced by DSSP for the GPAUV and the input file used to run DSSP are provided in Appendix A.2.2.

3.6.3 CFD-based Models

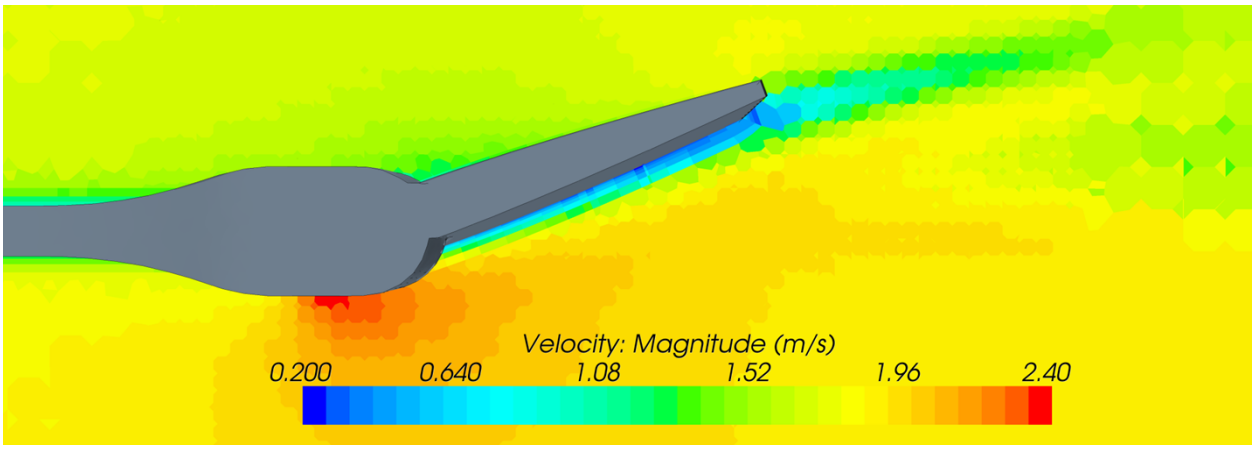
Two state-space models were constructed for the GPAUV were constructed using the SI process discussed in Section 3.5.3 to obtained model coefficients from the results of CFD simulations.

3.6.3.1 Virtual PMM Tests of GPAUV

A set of six virtual PMM tests served to inform the SI process for both of these models. The methodology used for these simulations follows that discussed in Section 3.5.2.2. A minimum of six oscillatory tests (one for each degree-of-freedom) are required to identify the coefficients for a six degree-of-freedom model. The parameters for these tests, chosen to match the known operating regime of the GPAUV, are presented in Table 3.4.



(a) With propeller in operation



(b) Without propeller in operation

Figure 3.24: Velocity magnitude near upper rudder when deflected to 20° with and without propeller in operation (color bar in (b) applies to both figures). Note: (a) is identical to Figure 3.22b.

Table 3.4: Virtual PMM tests of GPAUV for SI and state-space model population.

Degree of Excitation	Nondimensional amplitude, a'_0	Nondimensional frequency, ω'	Maximum velocity	Maximum acceleration
x (surge)	0.015	5	2.15 m/s	0.739 m/s^2
y (sway)	0.015	5	0.15 m/s	0.739 m/s^2
z (heave)	0.015	5	0.15 m/s	0.739 m/s^2
p (roll)	0.01	5	5.7 deg/s	28.2 deg/s^2
q (pitch)	0.01	5	5.7 deg/s	28.2 deg/s^2
r (yaw)	0.01	5	5.7 deg/s	28.2 deg/s^2

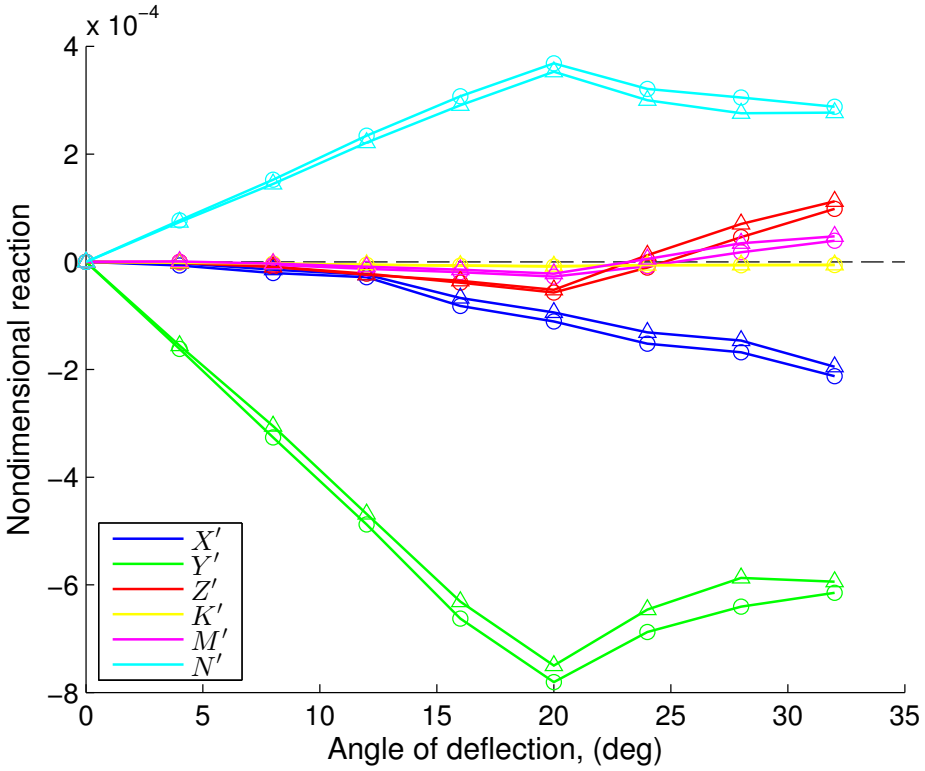


Figure 3.25: Upper rudder performance with (circle) and without (triangle) propeller.

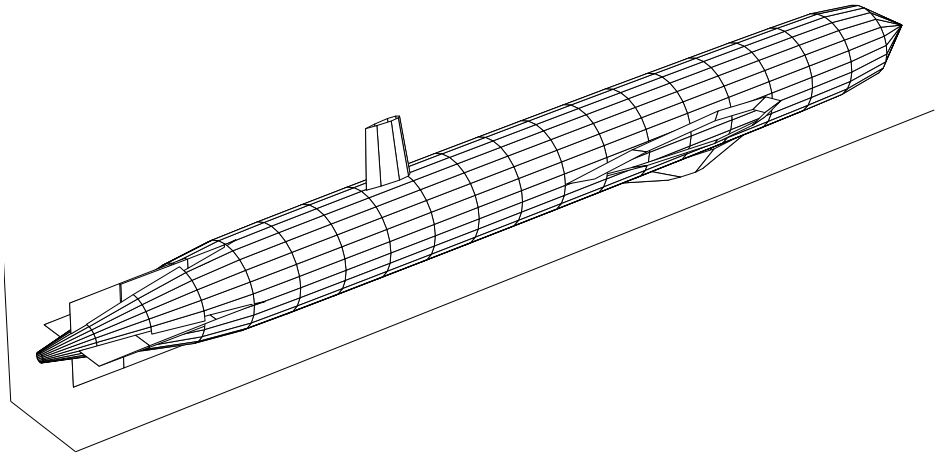
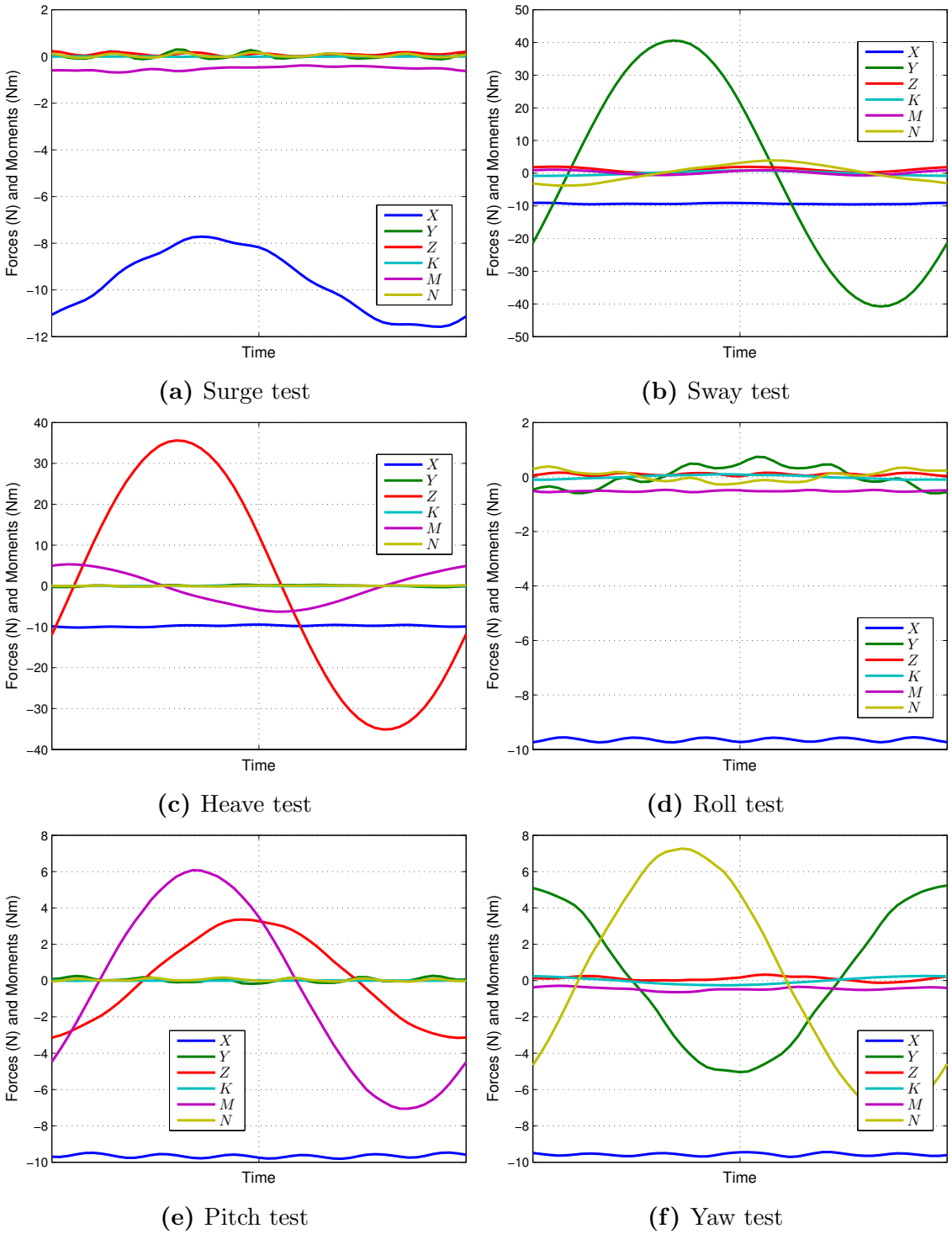


Figure 3.26: DSSP representation of the GPAUV.

Each simulation was performed for three periods of oscillation, with only the final period being used in the SI process. Figure 3.27 shows the forces and moments over the last period of oscillation from each of these tests.



(a) Surge test

(b) Sway test

(c) Heave test

(d) Roll test

(e) Pitch test

(f) Yaw test

Figure 3.27: Force and moments predicted in virtual PMM tests of the GPAUV. (Legend shown in (a) applies to all other sub-figures.)

3.6.3.2 Linear-Damping and Nonlinear-Damping Models

The models populated using the results from these virtual PMM tests both use the same basic vector form shown in (3.65). The only difference between these models is the way in which viscous damping is modeled. The "Linear Model" uses a simple linear representation of damping.

$$\vec{\tau}_{\text{damping}} = \mathbf{D}_{\vec{\nu}}\vec{\nu} \tag{3.85}$$

The shortcomings of employing a linear formulation for viscous damping are well known and this model is presented as somewhat of a baseline comparison.

In the "Nonlinear Model", the Taylor series expansion illustrated by Figure 3.7 is employed to model damping. This is a novel formulation developed during the course of this study. The model can be written mathematically as

$$\vec{\tau}_{\text{damping}} = \mathbf{D}_{\vec{\nu}^2}\vec{\nu}^2 + \mathbf{D}_{\vec{\nu}|\vec{\nu}|}\vec{\nu}\,|\vec{\nu}|\,, \tag{3.86}$$

with the damping matrices defined as

$$\mathbf{D}_{\nu^2} = \begin{bmatrix} X_{u^2} & X_{v^2} & X_{w^2} & X_{p^2} & X_{q^2} & X_{r^2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ Z_{u^2} & Z_{v^2} & Z_{w^2} & Z_{p^2} & Z_{q^2} & Z_{r^2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ M_{u^2} & M_{v^2} & M_{w^2} & M_{p^2} & M_{q^2} & M_{r^2} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.87}$$

$$\mathbf{D}_{\nu|\nu|} = \begin{bmatrix} X_{u|u|} & 0 & X_{w|w|} & 0 & X_{q|q|} & 0 \\ 0 & Y_{v|v|} & 0 & Y_{p|p|} & 0 & Y_{r|r|} \\ Z_{u|u|} & 0 & Z_{w|w|} & 0 & Z_{q|q|} & 0 \\ 0 & K_{v|v|} & 0 & K_{p|p|} & 0 & K_{r|r|} \\ M_{u|u|} & 0 & M_{w|w|} & 0 & M_{q|q|} & 0 \\ 0 & N_{v|v|} & 0 & N_{p|p|} & 0 & N_{r|r|} \end{bmatrix}. \tag{3.88}$$

The coefficients that comprise each model for the GPAUV are provided in Appendix A.2.3.

3.6.4 Numerical Implementation

State-space models for the GPAUV were written in Matlab using object-oriented framework [118]. Models were run on Breaker (see Appendix C) and required approximately 3 minutes to compute a 3.5 minute long maneuver.

Chapter 4

Virtual Free-Running Model (VFRM) Simulations

This chapter presents the implementation and results of VFRM simulations. These simulations rely heavily on the methodology discussed in Chapter 2. Details specific to VFRM simulations are also presented. The results from VFRM simulations of a series of maneuvers are compared with those from the state-space models described in Chapter 3, as well as results from experimental field trials.

4.1 Simulation Methodology

4.1.1 VFRM Programming Structure

VFRM simulations completed for this study employed a Java macro to control the solution process. A listing of this macro is provided in Appendix B.2. The interaction scheme that occurs between the macro and the CFD simulation during a VFRM simulation is illustrated in Figure 4.1. The process begins with the transfer of the vehicle's state $(\eta, \nu, \dot{\nu})$ from the CFD simulation to the Java macro. With this information, the macro evaluates the control algorithm to compute a set of control inputs. These include control surface deflections, $\vec{\delta}$, and propeller RPM.[1]

The implementation of dynamic control surface deflections in the CFD simulation dictates that rotation rates, instead of deflection angles, be input. Therefore, control surface rotation rates, $\dot{\vec{\delta}}$, must be determined from the commanded control surface deflections. This task is

[1] In this study, no speed controller was employed; instead propeller RPM was defined as a constant for each maneuver.

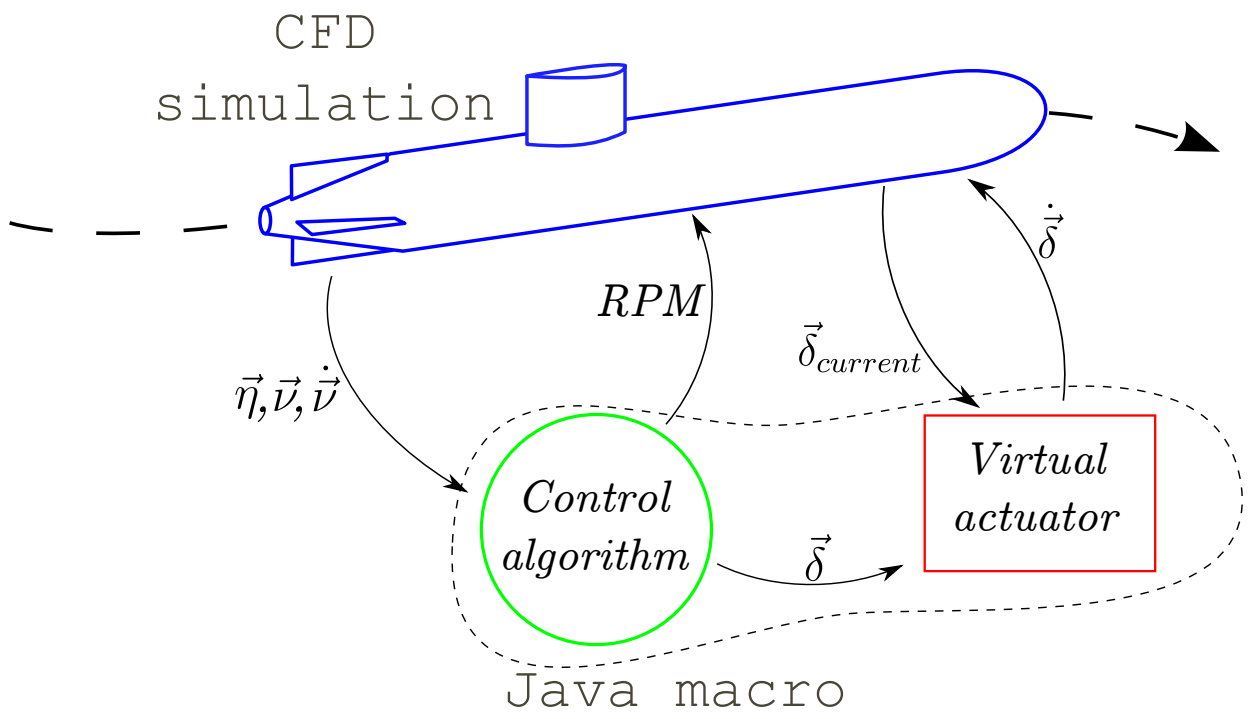


Figure 4.1: VFRM simulation exchange process.

accomplished by a portion of the macro referred to as the "virtual actuator." The virtual actuator takes the current positions of the vehicle's control surfaces, $\vec{\delta}_{current}$, and adjusts their rotation rates, $\dot{\delta}$, so as to move them to their commanded deflections. The rotation rate is selected by one of two rules, depending on the magnitude of the error between a control surface's current and commanded deflection.

$$\vec{\delta}_{error} = \vec{\delta} - \vec{\delta}_{current} \tag{4.1}$$

1. If $\vec{\delta}_{error}$ is greater than the angle that can be covered in one time step, the maximum rotation rate[1] ($\dot{\delta}_{max} = 6.3\,\text{rad/s}$) is applied.

2. If $\vec{\delta}_{error}$ is less than the angle that can be covered in one time step, a rotation rate is set so that the control surface will reach its desired deflection by the end of the next time-step. As discussed in Section 2.8.2.5, the precision of the applied rotation rate is limited in order to prevent a control surface from occupying an untested position.

[1] The maximum rotation rate of the GPAUV's control surfaces was determined experimentally by commanding one of the control surfaces to deflect back-and-forth between its maximum deflections ($\delta_{max} = \pm 18°$) and measuring the number of cycles completed over a given period of time.

$$\dot{\delta} = \begin{cases} \frac{\delta_{\text{error}}}{|\delta_{\text{error}}|}\dot{\delta}_{max}, & \text{for } |\delta_{\text{error}}| \geq \dot{\delta}_{\text{max}}\Delta t \\ \frac{\delta_{\text{error}}}{\Delta t}, & \text{for } |\delta_{\text{error}}| < \dot{\delta}_{\text{max}}\Delta t \end{cases} \tag{4.2}$$

To perform both control algorithm and virtual actuator iterations, information is actually passed back and forth between the Java macro and running CFD simulation at two different frequencies. The virtual actuator portion of the macro must be evaluated at every time-step. The interval at which the control algorithm is evaluated is dependent on the operating frequency of the actual vehicle's navigation system, $f_{\text{Controller}}$, and the time-step size of the CFD simulation, Δt. The controller must be evaluated every λ time-steps to provide a new set of control surface deflection commands.

$$\lambda = \frac{1}{f_{\text{Controller}}\Delta t} \tag{4.3}$$

For the GPAUV, $f_{\text{Controller}} = 10\,\text{Hz}$. Note that this creates a requirement that the selected CFD simulation time-step be a divisor of the controller frequency.

$$\text{mod}\left(\frac{f_{\text{Controller}}}{\Delta t}\right) = 0 \tag{4.4}$$

4.1.2 Simulation Initialization

Arbitrary initialization of CFD simulations involving fluid structure interaction is often problematic. As the flow field is uniform (with the velocity in the entire domain equal to the freestream velocity), the forces and moments on the body are unrealistic and meaningless. Allowing the body to react to this unrealistic condition introduces unnecessary transients into the solution. In some cases this may result in a divergent solution; in others it may create a situation that necessitates further iteration for the body and solution to recover to a desired state. Even in this second scenario, expensive computational time is wasted collecting irrelevant data.

An attractive alternative, which was employed in this study, is to initialize the unsteady simulation with the solution of a steady simulation. For instance, when running a simulation of a maneuver starting from a steady forward speed of 2 m/s, a steady simulation can be run at 2 m/s to provide an initial flow field. For a VFRM simulation, it is important that steady simulation be executed using a translating reference frame (with velocity inlet boundaries set to zero), to provide the proper flow field (see Section 2.8 for more on this concept).

4.1.3 Modes of Model Comparison

VFRM simulations present a unique maneuvering analysis option with a number of benefits and drawbacks. To better understand the abilities of this method, comparisons were made with state-space models and experimental testing measurements. Figure 4.2 shows the two modes of maneuvering model comparison considered in this study.

The most straightforward comparison between two maneuvering prediction methods is that in which the paths predicted by each method for a single reference (i. e., "commanded") path are compared (Figure 4.2a). While this approach allows one to assess the overall discrepancy between the methods, it does little to indicate that discrepancy's source. Any difference between the two paths propagates forward with time, making direct comparisons between the two predictions ineffective.

A more direct comparison is one in which a single state history (including vehicle motion and control input) is used by each model to predict forces and moments on the vehicle. This "prescribed motion" comparison is depicted in Figure 4.2b. This mode of comparison is meant to make the results from the models more directly comparable at each time-location during a maneuver. Another advantage of a prescribed motion comparison is the ability to consider a reduced set of each model's components. If, for instance, two models employ the same control sub-model, as is the case for the state-space models considered in this study, it may be beneficial to discard the reactions produced by the control surface sub-model in the interest of more clearly identifying other effects.

4.1.4 Metrics for Maneuvering Comparison

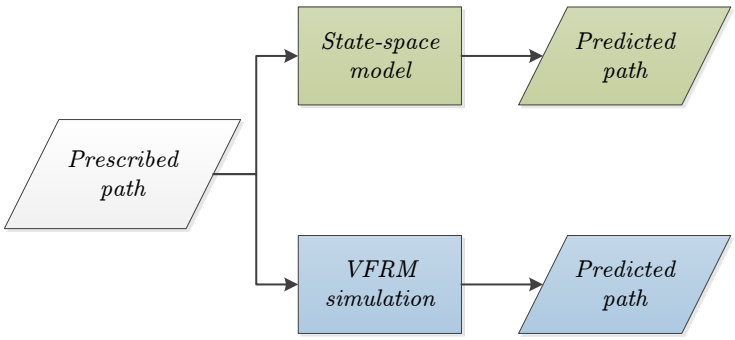
In addition comparing general trends and magnitudes of the predictions from each method, a number of metrics are used to provide a more quantitative means of comparison. Figure 4.3 illustrates the nomenclature to describe a system's step response.

The damping ratio, ζ, can be computed for a step response as

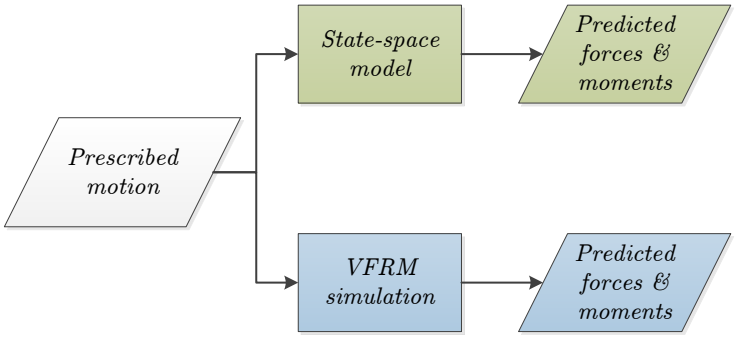
$$\zeta = \sqrt{\frac{\left(\ln \frac{PO}{100}\right)^2}{\pi^2 + \left(\ln \frac{PO}{100}\right)^2}}, \tag{4.5}$$

where PO is the percentage overshoot. The percentage overshoot for the i^{th} element in the position and orientation vector, $\vec{\eta}$, can be defined, as illustrated in Figure 4.3, by

$$PO_i = \frac{\eta_i^{max} - \eta_i^{ref}}{\eta_i^{ref}}. \tag{4.6}$$



(a) Prescribed path.



(b) Prescribed motion.

Figure 4.2: Modes of maneuvering model comparison.

4.1.5 Computational Cost

The solution process for any VFRM simulation, no matter the degree of computational optimization, will require extensive computational resources. Based on this reality, an optimization of computational cost of a VFRM simulation is essential. To allow for a more direct comparison with other CFD simulations, the computational cost of a simulation can be defined as the number of CPU-seconds required to advance the simulation by one time-step divided by the total number of finite volume cells in the domain. Simulation cost data will be reported in the results sections of this chapter.

$$\text{COST}_{\text{CFD}} = \frac{\text{CPU-seconds per time-step}}{\text{Number of FV cells}}. \tag{4.7}$$

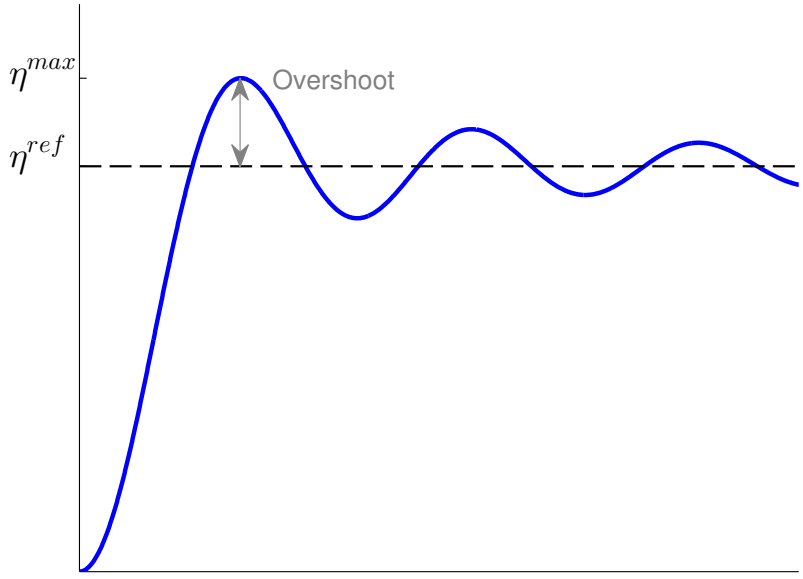


Figure 4.3: Step response overshoot.

4.2 Prolate Spheroid VFRM Simulations

To further verify the methods to be applied to the GPAUV, an analysis of the maneuvering behavior of a prolate spheroid was conducted [119]. As with the GPAUV, a full 6-DoF state-space model was developed via an SI process. While this approach involves a minimum of six unsteady CFD simulations for an arbitrary body (one for each degree-of-freedom), the three planes of symmetry present for a prolate spheroid reduce the number of unique motions to four (surge, heave, roll and pitch). As the prolate spheroid does not represent an actual vehicle, but instead a canonical geometry for the demonstration and verification an analytical approach, it has no standard operating regime. Instead, the spheroid was tested in a regime similar to that of the GPAUV.

4.2.1 Approach

A prescribed motion comparison, as depicted in Figure 4.2b, was used to analyze the performance of the VFRM and state-space methods on a prolate spheroid. The task of selecting a "realistic" motion to be prescribed to the numerical models for evaluation is somewhat challenging. To be realistic, a motion should not force the vehicle to move or accelerate in a way that is dissimilar to what it could accomplish in self-power flight. To circumvent the task of selecting such a motion, a modified version of a prescribed motion approach shown in Figure 4.2b was employed. The procedure applied in this analysis was as follows:

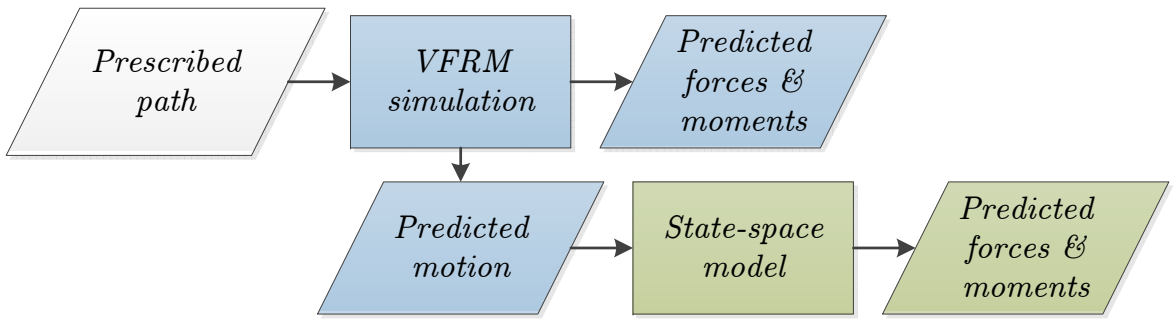


Figure 4.4: Modified prescribed motion comparison.

1. Identify a state-space model to represent the 6:1 prolate spheroid using unsteady CFD simulations and an SI process.
2. Using the state-space model, tune a controller capable of "flying" the spheroid.
3. Perform a VFRM simulation in which that controller is used to perform a simple maneuver. The motion followed by the spheroid during this simulation is taken to be the "prescribed" motion.
4. Evaluate the state-space model for the motion predicted in that VFRM simulation.
5. Compare the forces and moments predicted by the state-space model to those computed in the VFRM simulation.

Figure 4.4 illustrates this modified prescribed motion comparison approach.

For this study, a single maneuver was analyzed in which the spheroid is initially traveling at a forward speed of 1 m/s. At a time of 2 s into the test, the spheroid is commanded to a new yaw heading angle of $\psi = 4°$. The resulting maneuver was monitored up to a time of 30 s.

4.2.2 State-Space Model of a 6:1 Prolate Spheroid

A state-space model was developed for the 6:1 prolate spheroid in the form of (3.65), with damping represented by a linear model for simplicity. The SI process described in Section 3.5.3 was used to populate the model's coefficients for an operating speed of 2 m/s. The added mass matrix of the spheroid was determined from the analytic solution. The coefficients of this model are provided in Appendix A.3.

The model was employed in two modes of operation:

1. **Integration mode:** When being used to tune the PI controller, the state-space model is integrated using a Dormand-Prince implementation of the Runge-Kutta method. This is the standard mode of operation for a state-space model.

2. **Evaluation mode:** When evaluated for comparison against the VFRM simulation, no integration scheme is required as (3.65) is reformulated to

$$\vec{\tau}_{\text{state-space}} = M_A\dot{\vec{\nu}} + C_A\left(\vec{\nu}\right)\vec{\nu} + D\vec{\nu} + \vec{g}\left(\vec{\eta}\right). \tag{4.8}$$

   In this mode of operation, the model serves only to return a set of predicted forces and moments as a function of state (no controller is employed).

4.2.3 PI Controller

A prolate spheroid is inherently unstable due to the Munk moments discussed in Section 3.3.3.4. Unless stabilized by an external control force, a spheroid will tend to tumble end-over-end during flight, as any perturbation from purely axial flow will induce a destabilizing moment. A standard PI controller was tuned to stabilize the spheroid based on state-space model discussed in Section 4.2.2.

As a spheroid has no actual control surfaces to direct its path, the effect of a set of realistically-sized control surfaces were provided to the controller. The performance of these control surfaces was based on the control sub-model developed for the GPAUV (see Section 3.6.1). In order to control the highly-unstable spheroid, the effectiveness of these control surfaces was scaled by a factor of three (only the pitching moment, M, and yawing moment, N, performances were scaled). These control surfaces were fictitious in the sense that their geometry was not modeled in the VFRM simulation. Instead, external forces for the state-space model were applied to in the CFD simulation. As with the GPAUV, the rotational velocity of the control surfaces was limited to 6.3 rad/s. The same actuator disk used in simulations of the GPAUV was also applied to these simulations with the prolate spheroid (see Section 2.7.2).

4.2.4 VFRM Simulation of a Prolate Spheroid

Details of the CFD methodology used in the VFRM simulation of the prolate spheroid follow the methods used for the validation studies discussed in Section 2.10.2. Details specific to this simulation are shown in Table 4.1. The same computational domain used for pitch-up maneuver simulations of the prolate spheroid (see Section 2.10.2.3 and Figure 2.50) was used for VFRM simulations. The time-step of $\Delta t = 0.025\,\text{s}$ corresponds to a nondimensional time-step size of $\Delta t' = 0.0125$.

During the initial two seconds of the simulation, the spheroid was "locked" in all but the x-degree-of-freedom to allow the simulation to initialize as described in Section 4.1.2. In this locked state, the body is allowed to accelerate only in the x-direction.

Table 4.1: Prolate spheroid VFRM simulation details.

Property	Value
Time-step size (s)	0.025
Average CFL number	0.1
Control algorithm frequency (Hz)	10

This simulation was computed on Arcturus (see Appendix C) using 48 cores. Each time-step required 6.9 s of wall time (5.5 min of CPU-time) to compute. The entire 30 s maneuver required 2.26 hr of wall time to complete. The computational cost of the simulation was $COST_{CFD} = 1.26 \times 10^{-4}$ CPU-seconds-per-time-step-per-cell (see Section 4.1.5).

4.2.5 Results and Discussion

The motion of the prolate spheroid during its maneuver is illustrated in Figures 4.5 and 4.6. Figure 4.5 shows the orientation of the spheroid as it attempts to realize a new heading of $\psi = 4°$. Figure 4.6 shows the spheroid's velocity during this maneuver. The large rotational accelerations in Figure 4.6b are caused by the simulation's powerful control surfaces and the controller's ability to change their deflections quite quickly. The roll oscillation in Figure 4.5 results from the torque applied by the simulated propeller. As a spheroid has no damping or added mass in roll, excitations in this degree-of-freedom are only damped by the hydrostatic righting moment due to the vertical separation of the body's centers of buoyancy and gravity. The rotation in pitch is linked to that in roll in that once rolled to a non-zero angle, the vehicle's control surfaces no longer act in mostly decoupled horizontal and vertical planes.

Figure 4.7 shows the forces and moments predicted by the VFRM simulation and state-space model for the motion determined from the VFRM simulation. The forces and moments plotted in Figure 4.7 include contributions from surface pressure and shear stress on the body of the spheroid. Forces and moments from the control surface and propeller sub-models are not included as the same sub-models were used in the VFRM simulation and state-space model.

Overall, the two methods return fairly similar predictions. The major discrepancy in the X-force, shown in Figure 4.7a, is due to the fact that the SI process for the state-space model was performed about a forward operating speed of 2 m/s, while the test shown in Figure 4.7 occurs at approximately 1 m/s. This difference in speeds was a result of difficulties is designing a controller to stabilize the spheroid. This highlights the limitations of using a linear representation for viscous damping.

A more interesting disparity between the predictions is shown in Figure 4.7b. The state-space model and VFRM simulation results show different magnitudes of the peak in the N-moment

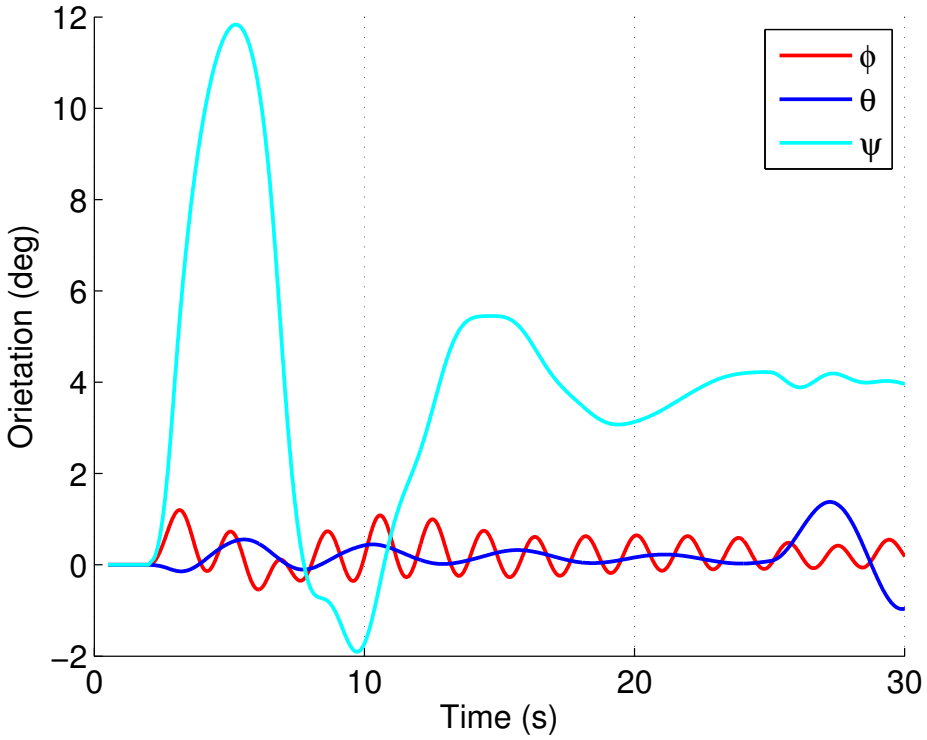
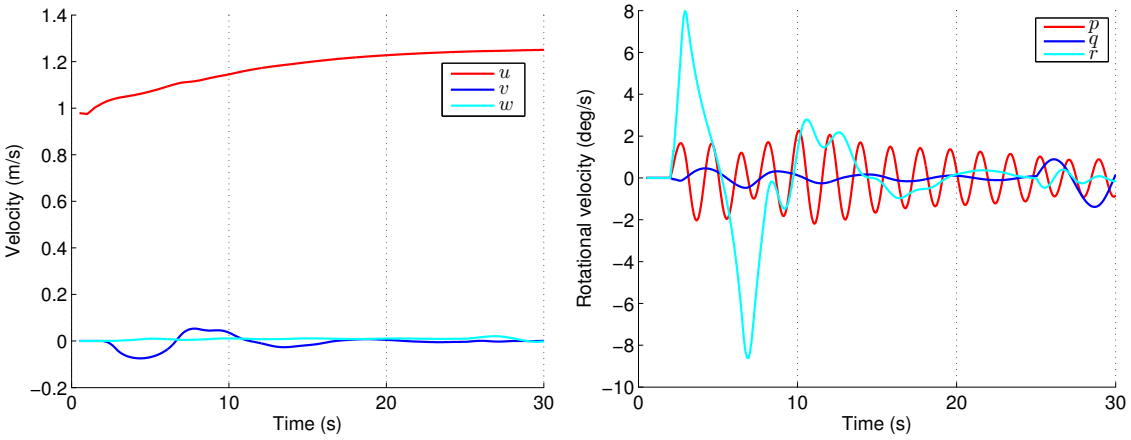


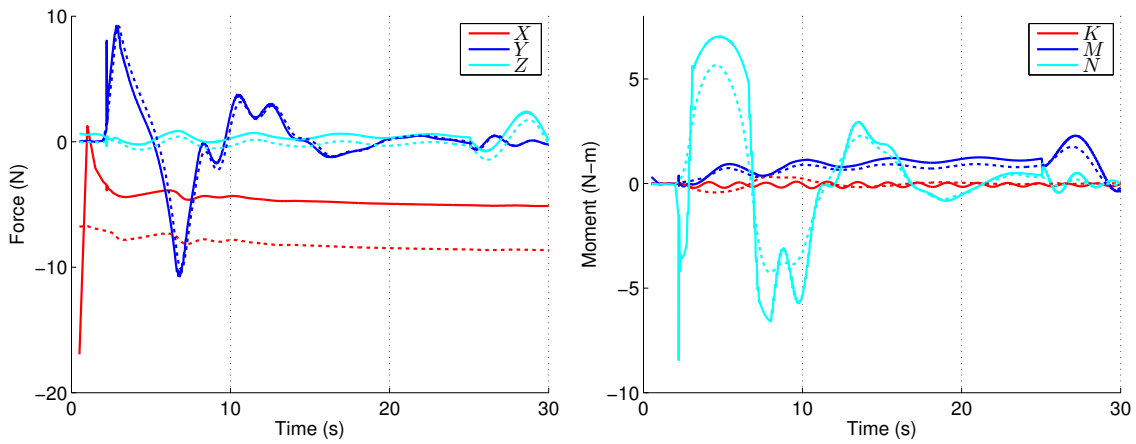
Figure 4.5: Orientation of prolate spheroid during maneuver.



(a) Linear velocity **(b)** Rotational velocity

Figure 4.6: Velocity of prolate spheroid during maneuver.

that occurs approximately 5 s into the test. A major discrepancy can also be seen in the predictions of the N-moment just before 10 s. The VFRM simulation shows a reaction with multiple peaks, suggesting a complex flow separation and reattachment behavior, while the



(a) Forces **(b)** Moments

Figure 4.7: Prescribed motion test predicted reaction from prolate spheroid VFRM simulation (solid lines) and state-space model (dashed lines).

state-space model captures a more general trend.

The comparisons presented in this study show clear similarities and differences between the CFD-based VFRM simulation and the linear state-space model. Forces and moments due to constant accelerations are predicted similarly by both methods. Large differences occur when the spheroid has a rapid change in acceleration (i. e., "jerk"). The results of this study focused on a prolate spheroid provide a basis for analysis of the GPAUV.

4.3 GPAUV VFRM Simulations

4.3.1 Simulation Domain and Boundary Conditions

A conic frustum shaped domain, based on that used for the prolate spheroid pitch-up simulations discussed in Section 2.10.2.3, was chosen for VFRM simulations of the GPAUV. Figure 4.8 shows a cross section of the domain and volume mesh as well as its boundary conditions. A closer view of the volume mesh near the GPAUV is shown in Figure 4.9 . The details of this simulation are presented in Table 4.2.

4.3.2 Coordinate Systems

A number of inertial and body fixed coordinate systems were employed in the VFRM simulations presented in this study. The locations of these coordinate systems are given with respect to the GPAUV's CoB Table 4.3 (the locations of vehicle's CoB and CoG are defined

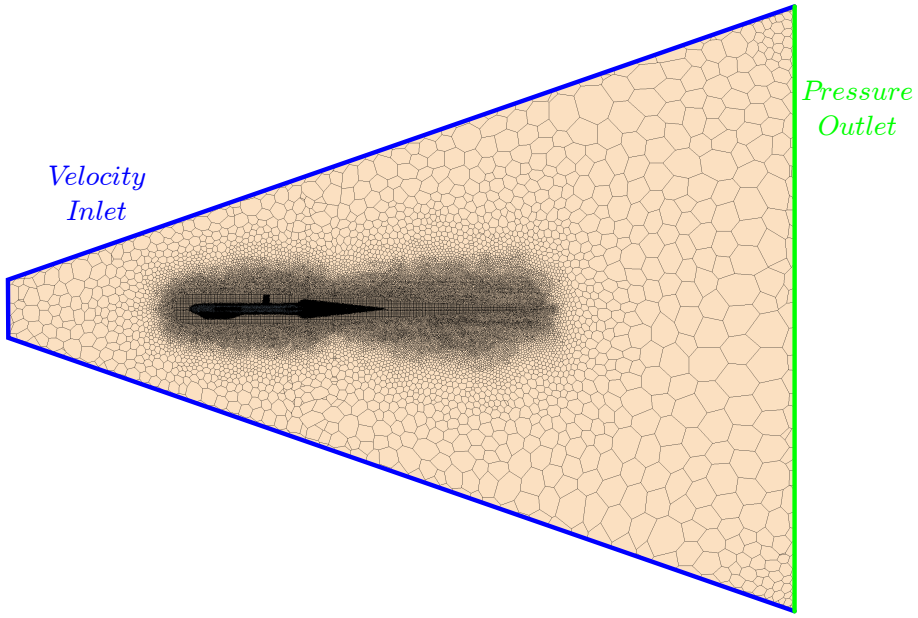


Figure 4.8: The computational domain used for VFRM simulations of the GPAUV.

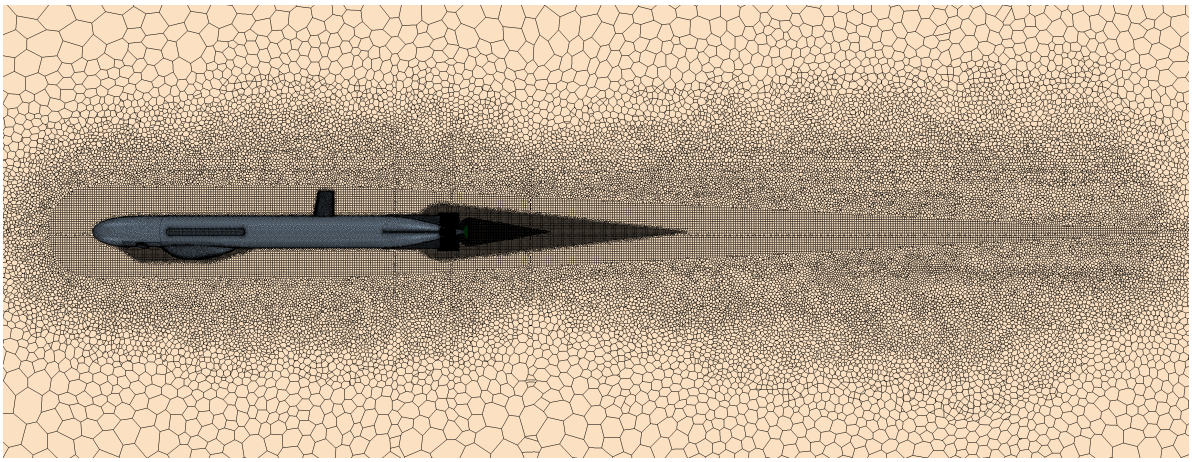


Figure 4.9: Near-body computational mesh used for VFRM simulations of the GPAUV.

Table 4.2: GPAUV VFRM simulation details.

Property	Value
Number of finite volume cells in domain	13.2×10^{6}
Average surface cell size ($\frac{A}{L^2}$)	
- Hull	1.3×10^{-5}
- Control surfaces	1.3×10^{-7}
- Other appendages	1.9×10^{-6}
Prism layer thickness ($\frac{t}{L}$)	
- Hull	4.9×10^{-4}
- Control surfaces	2.0×10^{-4}
- Other appendages	4.9×10^{-4}
Number of prism layers	
- Hull	4
- Control surfaces	6
- Other appendages	4
Average wall y^+	2.75
Time-step size (s)	[0.0125, 0.025 0.05]
Inner iterations per time-step	5
Average CFL number	[0.13, 0.26, 0.51]

Table 4.3: GPAUV coordinate systems

Coordinate System	**Abbreviation**	**Location in CoB**	**Orientation**
Inertial reference frame	NED-CSys	NA	North-East-down
Center of buoyancy	CoB-CSys	NA	forward-starboard-down
Center of gravity	CoG-CSys	[0, 0, 0.0254] m	forward-starboard-down
Actuator disk	Disk-CSys	[-1.115, 0.0, -0.0254] m	forward-starboard-down
Control surfaces		$(x = -0.996)$ m	x-forward, z-outward

relative to the vehicle's nose in Table 1.1). A *North-East-down* coordinate system, like that shown in Figure 3.2, was chosen to provide an inertial reference for maneuvers. Similarly, a *forward-starboard-down* body fixed coordinate system, like that shown in Figure 3.1, served as the main body-fixed system. Additional body-fixed systems were located at the propeller and each control surface.

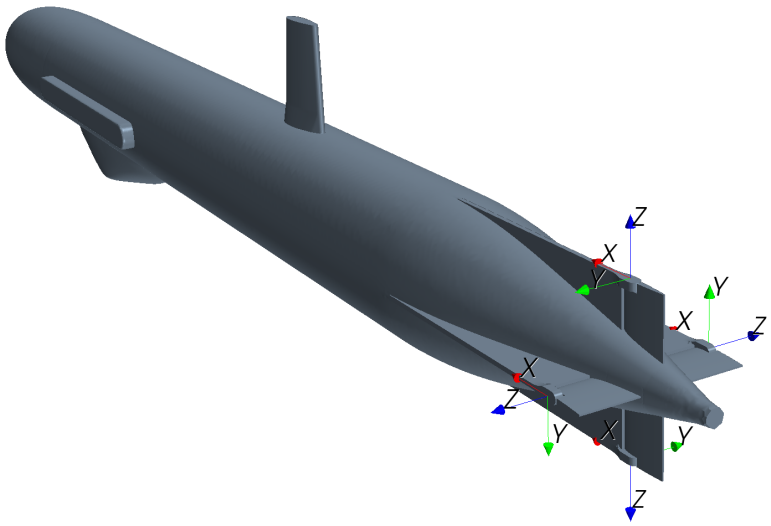


Figure 4.10: GPAUV tail section with local control surface coordinate systems.

The local control surface coordinate systems were each located 0.996 m aft of the center of buoyancy (see Figure 4.10). Each of these coordinate systems follows the motion of its respective control surface and is oriented with x-forward and z-outward. These coordinate systems serve to provide the VFRM macro with each control surface's orientation. This is accomplished by comparing each control surface coordinate system to the CoB coordinate system. This approach is considered to be less susceptible to integration errors when compared to the alternative method of integrating the commanded control surface rotation over time.

4.3.3 Computational Cost and Parallel Speed-up

Table 4.4 shows the computational cost of a number of simulations employing a variety of velocity solver under-relaxation factors. The pressure, turbulence and turbulent viscosity solver under-relaxation factors were set a 0.2, 0.8 and 1 respectively in each of these cases. These simulations were conducted on Blueridge (see Appendix C) using 320 cores. Of the configurations tested, the Case B solver settings shown in Table 4.4 give the most efficient solution process. These are the settings recommended by STAR-CCM+'s help manual for an unsteady incompressible flow [34].

Figure 4.11 shows the wall time required to advance the VFRM simulation by one time-step using different numbers of computational cores. This plot also shows the calculated COST_{CFD} value. Results plotted in Figure 4.11 are from simulations run on Blueridge. As expected, while the wall time required to iterate the VFRM simulation decreases with use of additional computing cores, the gains in solution speed do not keep pace with the increase in computational resources being employed.

Table 4.4: VFRM simulation computational cost optimization.

Case	A	B	C
Velocity under-relaxation factor	0.8	0.7	0.9
Wall time per time-step (min)	7.42	7.77	9.63
CPU time per time-step (s)	1.42×10^5	1.49×10^5	1.85×10^5
$COST_{\text{CFD}}$	1.27×10^{-2}	1.33×10^{-2}	1.65×10^{-2}

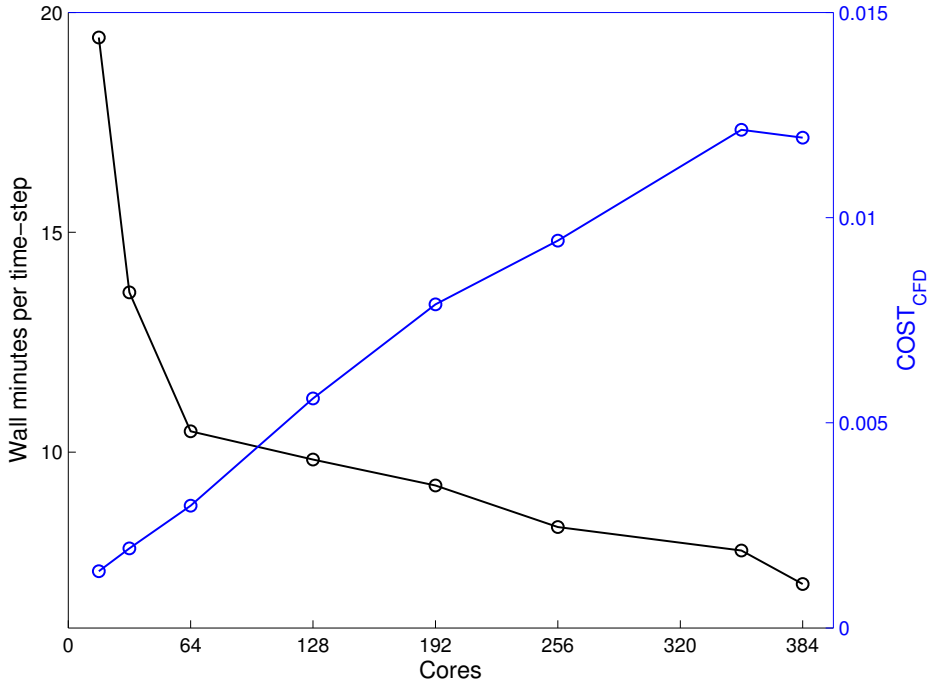


Figure 4.11: Wall time required to advance a VFRM simulation by one time-step and COST_{CFD} (see (4.7)).

Note that the computational cost of these simulations is roughly one order of magnitude greater than the prolate spheroid VFRM simulations (see Section 4.2.4). For the spheroid simulation, $COST_{CFD}(48 \text{ cores}) \approx 1 \times 10^{-4}$ CPU-seconds-per-time-step-per-cell. In simulations for the GPAUV on a similar number of cores, the value is roughly $2{\times}10^{-3}$ CPU-seconds-per-time-step-per-cell. Much of this additional computational load is due to the use of overset meshes in the VFRM simulations of GPAUV, which were not present the prolate spheroid simulation.

While the $COST_{CFD}$ metric is not widely used by other researchers, the parameters needed to compute it are often reported in publications. The submarine maneuvering simulations computed by Dreyer and Boger [22], which used fin models to account for the forces and moments induced from the deflection of control surfaces, required roughly $4{\times}10^{-3}$ CPU-seconds-per-time-step-per-cell on 88 cores. Although no dynamic fin deflections were included in this simulation, it did use overset meshes to allow for a maneuver involving both a submarine and nearby surface ship (see Section 1.3.2 for more details on this study). Maneuvering simulations of the SUBOFF completed by Pankajakshan et al. [27] required roughly $2{\times}10^{-4}$ CPU-seconds-per-time-step-per-cell on 57 cores. This study used fin models and a static computational mesh. The simulation methodology and $COST_{CFD}$ value for the simulations performed by Pankajakshan et al. are similar to those of the prolate spheroid maneuvering simulation discussed in Section 4.2.4.

4.3.4 Vehicle Control Algorithm

In all maneuvers considered in this dissertation, the GPAUV was controlled in yaw and pitch using an multi-input multi-output (MIMO) controller. As is the case with many underwater vehicles, the GPAUV uses passive roll stabilization. The controller was developed by Stilwell for a predecessor of the GPAUV that performed in a similar operating regime using an H_∞ control architecture [120, 121]. A fin-mixing law is employed to map the controller's output onto the vehicle's vertical rudders and horizontal dive planes as a function of roll angle.

There is a slight discrepancy between the controller employed on the experimental vehicle and the one used in the numerical simulations presented in this dissertation. On the actual GPAUV, an outer-loop proportional integral (PI) controller is used to stabilize the vehicle's depth. The PI controller produces a pitch command which is used, in combination with the yaw command, by the H_∞ controller to find a set of control surface deflections. In the numerical simulations for this dissertation, no depth controller was employed. Thus, instead of obtaining a slightly negative pitch orientation ($\theta \approx -1^\circ$) to counteract the positive buoyancy and maintain a constant depth, the vehicle simply was commanded to maintain a zero-pitch orientation. Future work to employ a depth controller in the numerical simulations is discussed in Section 5.1.1.6.

Table 4.5: Maneuvering comparison cases for the GPAUV.

Maneuver	Field experiment	VFRM simulation	State-space models
Step-turn			
20°		X	X
35°	X	X	X
60°		X	X
Pseudo zig-zag			
(0° to -20°), 0.1 Hz	X	X	X
(60° to -60°), 0.1 Hz		X	X

4.3.5 Cases for Comparison with State-Space Models and Field Tests

To better analyze the considered methods of maneuvering analysis (experimental, state-space model and VFRM simulation), comparisons were conducted on five different vehicle maneuvers. These maneuvers are listed in Table 4.5. Each row in the table represents a different maneuver case and each column, an analysis method. An X mark in a column indicates that data from an experimental trial or a prediction from that column's method is considered for the maneuver in that row. All maneuvers were performed at a forward speed of roughly 1.25 m/s, which corresponds to a propeller RPM setting of 1500. This speed, which is lower than the GPAUV's design speed of 2 m/s, was chosen to correspond more directly with a limited set of experimental data.

Step-turns to an offset of ψ = -20° were conducted in a VFRM simulation and with the state-space models. Three simulations with time-step sizes of $\Delta t = 0.05, 0.025, 0.0125$ s (these correspond to dimensionless time-step sizes of $\Delta t' = 0.081, 0.041, 0.020$) were completed for this 20° maneuver to analyze the effect of time discretization. A VFRM simulation and state-space model runs of a 35° step-turn maneuver were completed to correspond with previously collected field trial data. In order to elicit additional nonlinear phenomena, 60° step-turn maneuvers were also computed in a VFRM simulation and with the state-space models.

The "pseudo zig-zag" maneuvers presented in this chapter differ slightly from traditional zig-zag tests. In a standard zig-zag maneuver, a vehicle's rudder(s) are set to some constant

angle, δ_{zz}, until the change in the vehicle's heading is equal to that angle, $\Delta\psi = \delta_{zz}$. Once this change in heading is achieved, the rudder(s) are deflected to the opposite direction, $-\delta_{zz}$, and the processes is repeated [90]. In the pseudo zig-zag maneuvers completed for this study, instead of being locked to some constant deflection, the control algorithm is allowed to determined the proper rudder angle dynamically. Also, the frequency at which turns are executed within the maneuver is predefined, instead of being based on when some desired heading is reached. In this way, these maneuvers are actually more like a series of tightly-spaced step-turns. A pseudo zig-zag maneuver between yaw headings of 0° and -20° at a rate of 0.1 Hz was performed. This maneuver was selected specifically to correspond with previously performed experimental trials with GPAUV. A more aggressive pseudo zig-zag maneuver, between ±60° at a rate of 0.1 Hz was also considered using the numerical models (i. e., VFRM simulation and state-space models).

While maneuvers can also be performed in the vertical (x-z) plane, maneuvers in the horizontal plane were expected to produce more of the nonlinear effects that are of interest to this study. This expectation is based on the fact the GPAUV's largest appendages (the sail and appendage fairing) are mounted on its upper and lower surfaces (see Figure 1.1).

4.3.6 Field Tests

Field test data presented in this chapter were collected from tests conducted in Claytor Lake, VA. All maneuvers were performed at a depth of roughly 2 m. The reported measurements were obtained from the GPAUV's onboard navigational sensors. The GPAUV uses an attitude and heading reference system (AHRS) as well as a Doppler velocity log (DVL) and pressure sensor. Table 4.6 gives the source of each parameter recorded by the GPAUV's navigation system. The vehicle's position in the horizontal plane is approximated by integrating the signal from the DVL. Likewise, its linear accelerations are obtained by taking the derivative of the smoothed DVL signal. (While the AHRS measures linear accelerations, its signal is more noisy than that of the DVL.)

4.3.7 Results

4.3.7.1 20° Step-turn

Time-step Independence Figure 4.12 shows the yaw heading from the 20° step-turn VFRM simulations computed with time-step sizes of $\Delta t = 0.05, 0.025$ and 0.0125 s. Due to the computational expense of running simulations with small time-steps, the simulations using $\Delta t = 0.025$ and 0.0125 s were only solved to 4 and 5 s respectively. The paths predicted by these three simulations are nearly identical.

The forces and moments predicted during each of these 20° step-turn VFRM simulations are shown in Figure 4.13. The simulations employing smaller time-steps show increasingly large

Table 4.6: GPAUV sensors used for state estimation.

Parameters	Measurement device	Measurement frequency (Hz)
$[x, y]$	DVL (integrated)	5
$[z]$	Pressure sensor	10
$[\phi, \theta, \psi]$	AHRS	1000
$[u, v, w]$	DVL	5
$[p, q, r]$	AHRS	1000
$[\dot{u}, \dot{v}, \dot{w}]$	DVL (derivative)	5
$[\dot{p}, \dot{q}, \dot{r}]$	AHRS	1000

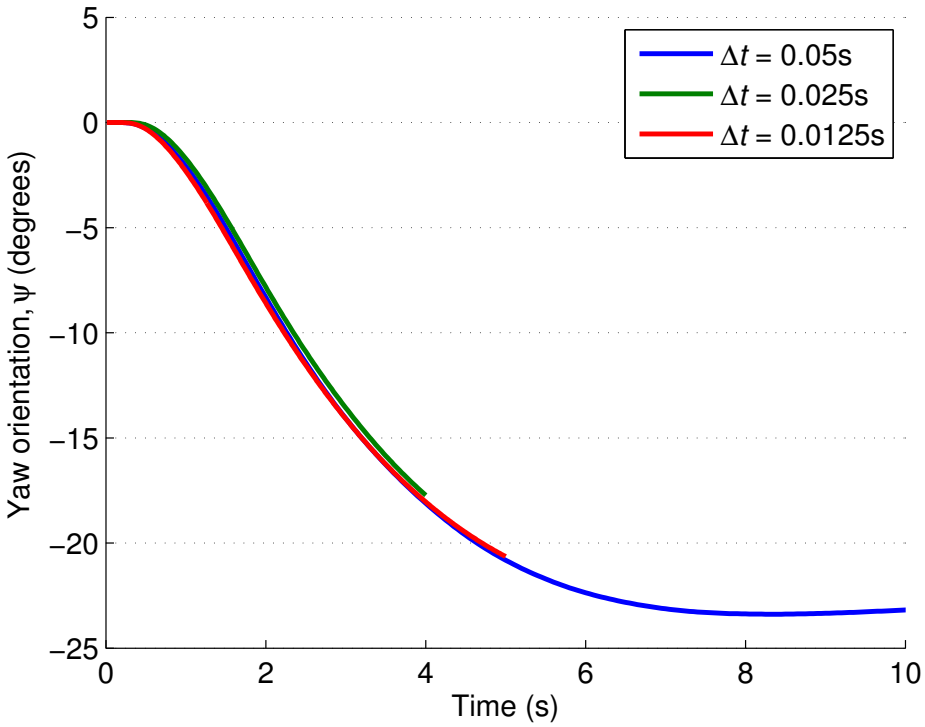


Figure 4.12: Yaw heading, ψ, during 20° step-turn maneuver from VFRM simulations with $\Delta t = 0.05, 0.025$ and $0.0125\,\mathrm{s}$

spikes in the predicted force and moment reactions on the GPAUV. These spikes coincide with, and are due to, fast rotations (and accelerations) of the GPAUV's control surfaces. Figure 4.14 shows the rotation rate of the GPAUV's control surfaces during these simulations.

Note that, as the GPAUV controller outputs only two control surface commands (i. e., one for the horizontal dive planes and one for the vertical rudders), only the starboard and lower control surface rates are plotted. Spikes in forces and moments in the horizontal plane (Y and N) are closely related to the rotational rate the GPAUV's rudders (Figure 4.14b), while spikes in forces and moments in the vertical plane (Z and M) are closely related to the rotational rate of the dive planes (Figure 4.14b). The X-force and K-moment are influenced more equally by the movement of both rudders and dive planes.

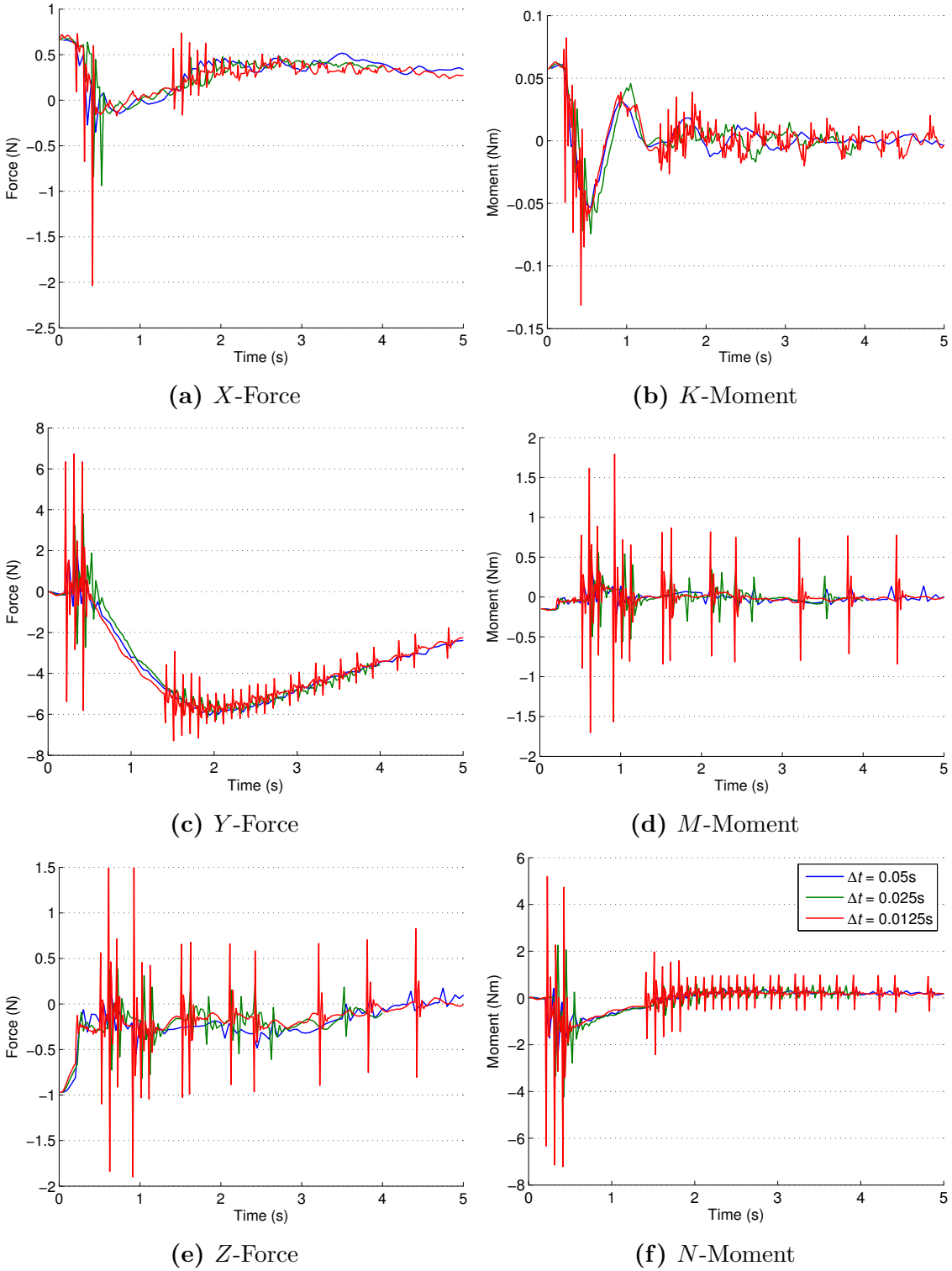
Two factors combine (and compound) to produce this inverse relationship between time-step size and the force and moment spikes seen in Figure 4.13:

1. As discussed in Section 4.1.1, the control surface rotation rate chosen by the VFRM Java macro is partially based on the angular distance traversed in a single time-step. Given a smaller time-step size, a smaller angular range will be swept by a control surface in a single time-step. Thus, the macro can use higher rotation rates (as long as they remain below the maximum rate) to the deflect the GPAUV's control surfaces.

2. As discussed in Section 2.10.2.3, prescribed accelerations are inversely proportional to time-step size. Because of this, the simulations with smaller time-steps experience larger accelerations for the same "instantaneous" changes in velocity. These higher accelerations result in larger added mass forces on the control surfaces.

Prescribed Path Comparison A comparison can be made of the results from a VFRM simulation and the three state-space models (Linear, Nonlinear and DSSP) for a single reference path input, following the method illustrated in Figure 4.2a. Each model was used to predict the vehicle's path for a 20° step-turn reference path in which, at $t = 0$, the vehicle is commanded from its current heading of $\psi = 0°$ to a new heading of $\psi = -20°$. As mentioned in Section 4.3.4, a single controller was employed across all three state-space models and the VFRM simulation. Figure 4.15 shows the results from each method for the vehicle's roll, pitch and yaw. Figures 4.15b and 4.15c include a black dotted line to indicate the reference path that the vehicle is attempting to follow (no reference for ϕ was given as the GPAUV does not actively stabilize in roll).

As the vehicle turns to port ($\psi < 0$), all four methods predict that it will roll into the turn ($\phi < 0$) and pitch slightly downward ($\theta < 0$). All three of the state space models predict larger angles of roll than the VFRM simulation (Figure 4.15a). The pitch response predictions, shown in Figure 4.15b, are all of the same (relatively small) order of magnitude, but vary widely in trend shape. Among the state-space models, only the DSSP model is in agreement with the VFRM simulation's prediction of a positive pitch disturbance early in the maneuver. The DSSP and Nonlinear models agree quite well with the VFRM simulation in yaw during the first 6 seconds of the maneuver, but show different trends as the vehicle attempts to correct for its overshoot (see Figure 4.15c). The Nonlinear model exhibits a ringing behavior in

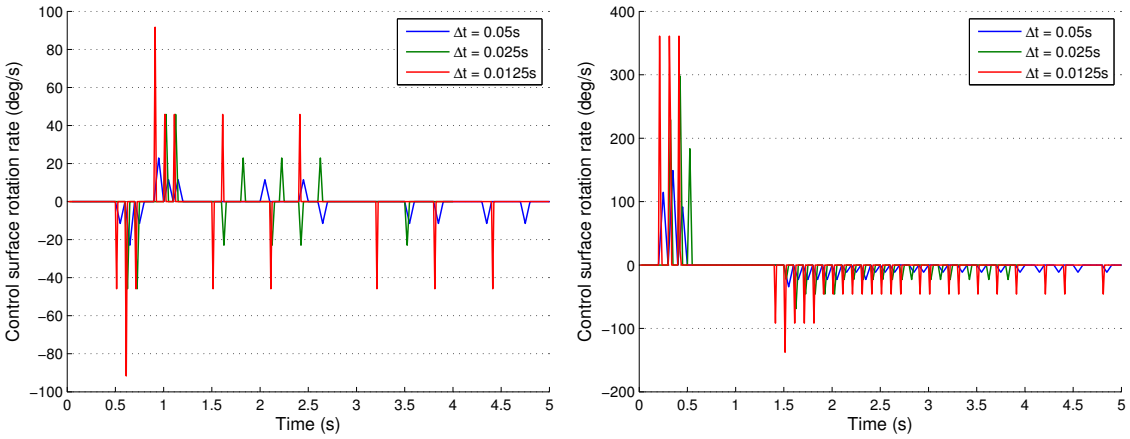


(a) X-Force **(b)** K-Moment

(c) Y-Force **(d)** M-Moment

(e) Z-Force **(f)** N-Moment

Figure 4.13: Predicted forces and moments during 20° step-turn maneuver from VFRM simulations with $\Delta t = 0.05, 0.025$ and $0.0125\,\mathrm{s}$. (Legend shown in (f) applies to all other sub-figures.)



(a) Starboard dive plane **(b)** Lower rudder

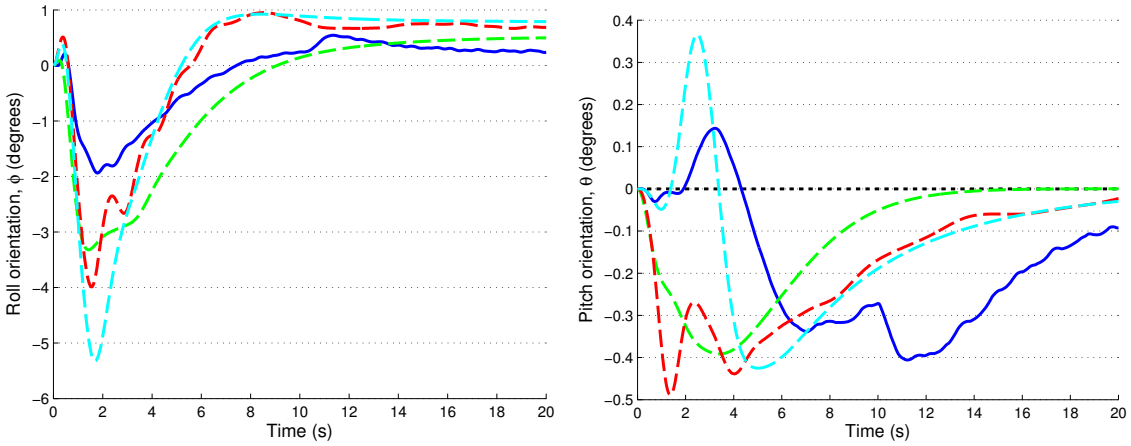
Figure 4.14: GPAUV control surface rotation rate during 20° step-turn maneuver from VFRM simulations with $\Delta t = 0.05, 0.025$ and $0.0125\,\text{s}$.

Table 4.7: Percentage overshoot and damping ratio for 20° step-turn maneuver.

Method	**Percentage overshoot,** PO	**Damping ratio,** ζ
VFRM simulation	16.9	0.492
Linear SSM	5.36	0.682
Nonlinear SSM	17.6	0.484
DSSP SSM	9.41	0.601

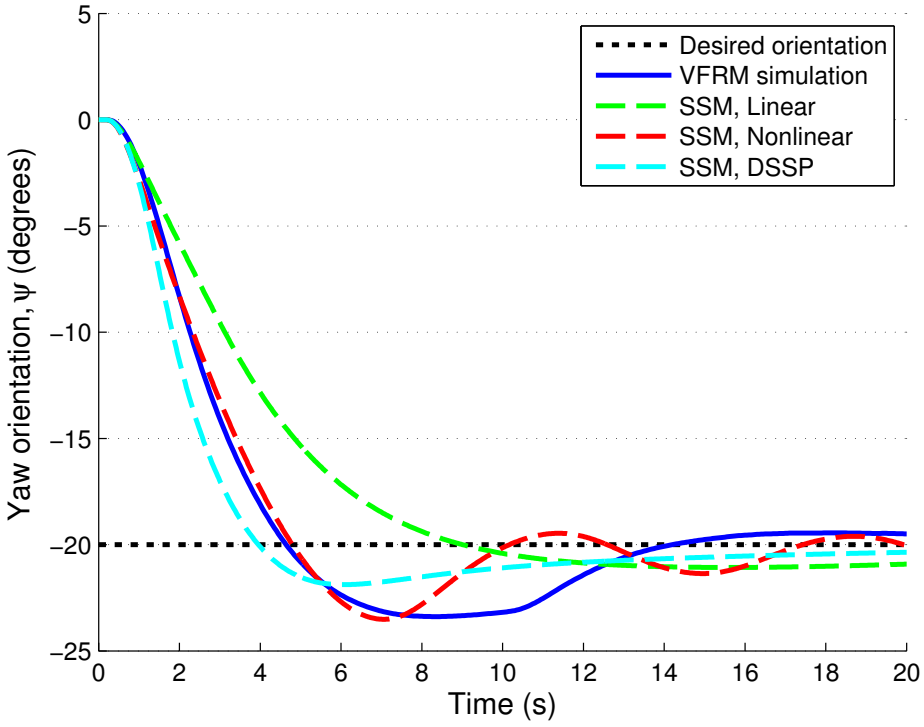
yaw not seen in predictions from the other methods. The percentage overshoot and damping ratio from each method are listed in Table 4.7.

Figure 4.16 shows the positions of the GPAUV's control surfaces during the 20° step-turn maneuver. When viewed alongside Figure 4.15, Figure 4.16 can be used to better understand the control response to the commanded heading change (at $t = 0$) and vehicle state. The orientation convention for these deflections is based on the coordinate systems shown in Figure 4.10. Note that in the Matlab implementation of the state-space models, the vehicle's controller is represented by a continuous bilinear approximation to allow for more straightforward incorporation with other model components. As a result, the control surface positions plotted in Figure 4.16 for the state-space models are at a much higher frequency than those of the VFRM simulation. With the different control surface deflection histories shown in Figure 4.16, it is clear that any comparison of the vehicle trajectories in Figure 4.15 must be made with deference to the fact that each model is operating independently.



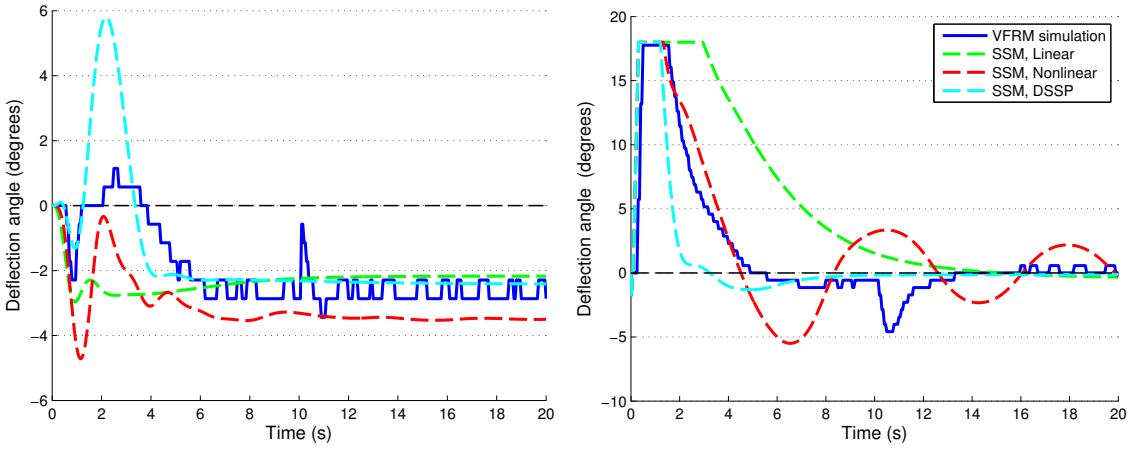
(a) Roll, ϕ

(b) Pitch, θ



(c) Yaw, ψ

Figure 4.15: GPAUV orientation during 20° step-turn maneuver from VFRM simulation and state-space models. (Legend shown in (c) applies to all other sub-figures.)



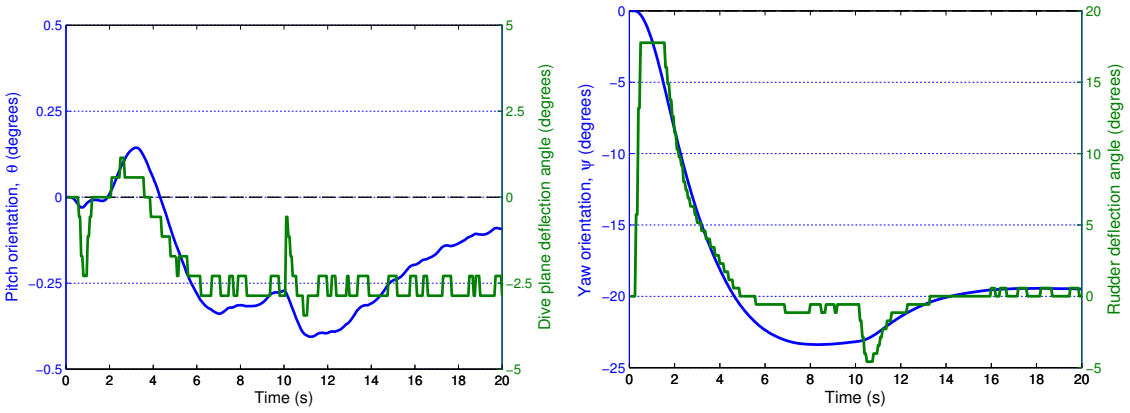
(a) Dive planes **(b)** Rudders

Figure 4.16: GPAUV control surface positions during 20° step-turn maneuver. (Legend shown in (b) applies to both sub-figures.)

Figure 4.17 shows time histories of vehicle orientation and control surface deflection from the VFRM simulation of the 20° step-turn maneuver. Figure 4.17a shows of the vehicle's pitch angle along with its dive plane deflection while Figure 4.17b shows of yaw angle and rudder deflection. Although any non-zero roll angle makes the coupling in either of these planes indirect, some interesting relations can still be made. From Figure 4.17a, one can see that pitch excursion beginning at $t = 10\,\text{s}$ is a reaction to dive plane deflection (not the other way around). This phenomenon was determined to be due to a "reset" in the vehicle controller's state caused by the a file-save operation during simulation of the maneuver. While the vehicle state, χ, was preserved, the controller state, which includes error integral terms, was not. This reset of the controller state caused the large reaction in the vehicle's dive plan angle seen in Figure 4.17b. A means of transferring the controller state has been developed to prevent this issue in future simulations.

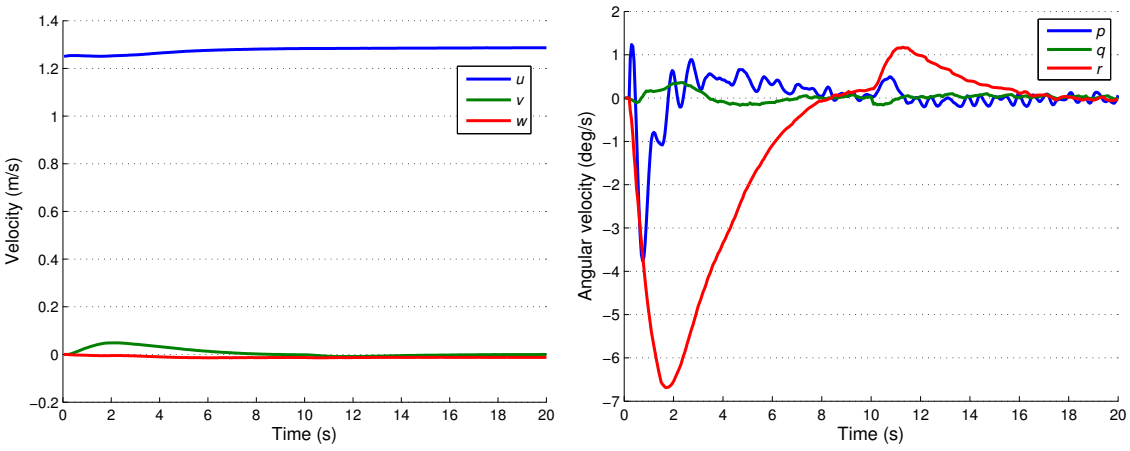
Prescribed Motion Comparison A prescribed motion comparison, following the concept illustrated in Figure 4.4, was conducted using the motion from the VFRM simulation of the 20° step-turn maneuver. Figure 4.18 shows the velocities of this motion that serve as the input (along with $\eta, \dot{nu}, \dot{\delta}$ and propeller RPM) to the three state-space models. This motion corresponds to the orientations for the VFRM simulation shown in Figure 4.15. Figure 4.19 shows the forces and moments predicted by the VFRM simulation and three state-space models for the motion path shown in Figure 4.18. The forces and moments plotted in Figure 4.19 include all hydrodynamic components (damping, added mass, control and propulsion) as well as the gravitational and hydrostatic components.

Agreement between the forces and moments from the state-space models and VFRM simulation is fairly good. Differences between the results from each method are somewhat more



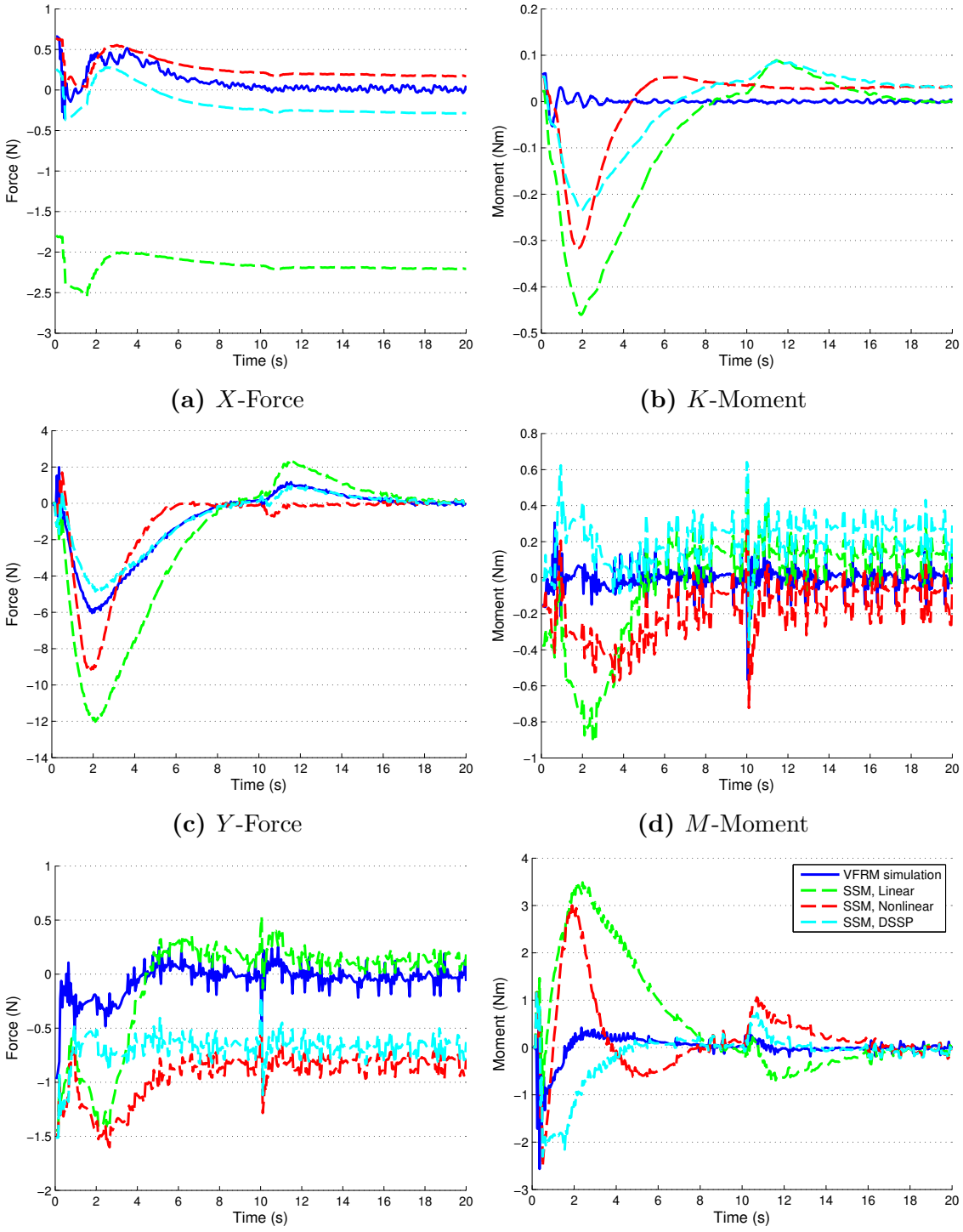
(a) Pitch orientation and dive plane angle **(b)** Yaw orientation and rudder angle

Figure 4.17: GPAUV orientation and control surface positions during VFRM simulation of 20° step-turn maneuver.



(a) Linear velocities **(b)** Angular velocities

Figure 4.18: Vehicle velocities predicted in VFRM simulation of 20° step-turn maneuver and used as input to state-space models in prescribed motion comparison.



(a) X-Force **(b)** K-Moment

(c) Y-Force **(d)** M-Moment

(e) Z-Force **(f)** N-Moment

Figure 4.19: Predicted forces and moments during prescribed motion 20° step-turn maneuver from VFRM simulation and state-space models. (Legend shown in (f) applies to all other sub-figures.)

amplified in this mode of comparison than in the prescribed path comparison shown in Figure 4.15. As with the comparison shown in Figure 4.7 for the spheroid maneuvers, the linear damping model predicts a substantially larger drag than the other methods (see Figure 4.19a). This is not surprising, as the model was linearized about a forward speed of $u = 2\,\mathrm{m/s}$, but is being asked to predict the drag $u = 1.25\,\mathrm{m/s}$ The X and Y-force predictions from the DSSP and Nonlinear state-space models show good agreement with the VFRM simulation (see Figures 4.19a and 4.19c). The Z-force history, shown in Figure 4.19e, from each of the models is relatively similar. while the Linear model matches the constant magnitude of Z-force for $t > 6\,\mathrm{s}$ in the VFRM simulation, the DSSP and Nonlinear models do a better job of capturing the trend shape.

The moment predictions, shown in Figures 4.19b, 4.19d and 4.19f, have much larger discrepancies than the force predictions. The state-space models all predict a destabilizing rolling moment that is an order of magnitude greater than the response of the VFRM simulation (see Figure 4.19b). This negative disturbance in roll moment predicted by the state-space models corresponds directly with the yaw velocity shown in Figure 4.18b. The source of this discrepancy between the VFRM simulation and state-space models' rolling moment predictions may be related to the fact that the state-space models were designed for use at a forward speed of $u = 2\,\mathrm{m/s}$ ($Re = 4.5 \times 10^6$), but are being employed analyze maneuvers at $u \approx 1.25\,\mathrm{m/s}$ ($Re = 2.8 \times 10^6$). It is possible that these two forward speeds create drastically different flow regimes, and therefore different viscous damping reactions. This hypothesis is partially supported by the plot of the GPAUV's drag coefficient as a function of Reynolds number shown in Figure D.1. The pitching moment responses, shown in Figure 4.19d, are all of a similarly small magnitude, however the Linear model shows a distinct disturbance at $t \approx 2.5\,\mathrm{s}$ that is much larger than any of the other models. The yawing moment responses in Figure 4.19f show an interesting variation. While the DSSP state-space model's yawing moment prediction matches the VFRM simulation rather closely, the Nonlinear and Linear model results show a large positive peak at $t \approx 2\,\mathrm{s}$. As with the trends discussed for the rolling moment, this spike is directly linked the yaw velocity shown in Figure 4.18b, and attributable to the viscous damping sub-model.

Part of the discrepancy between the VFRM simulation and state-space models appears to emanate from the dynamic motion of the GPAUV's control surfaces. Figure 4.20 shows the normal force on the GPAUV's rudders along with their deflection angle. Although the normal force on the rudders reaches a peak shortly after the rudders reach their saturation point of 18°, a large force is maintained as the rudders' deflections are reduced. Note that the normal force plotted in Figure 4.20 does not include forces induced on adjacent surfaces as a result of the rudders' deflections (see Appendix D.3 for more on this concept). The indirect relationship between the normal force and rudder deflection plotted in Figure 4.20 is also likely a result of non-zero side-slip of the vehicle. The side-slip of the GPAUV during the VFRM simulation of the 20° step-turn maneuver is plotted in Figure 4.21. While it would be possible to include terms in a state-space model to account for forces and moments created by the motion of a vehicle's control surfaces and non-zero side-slip, this is not commonly

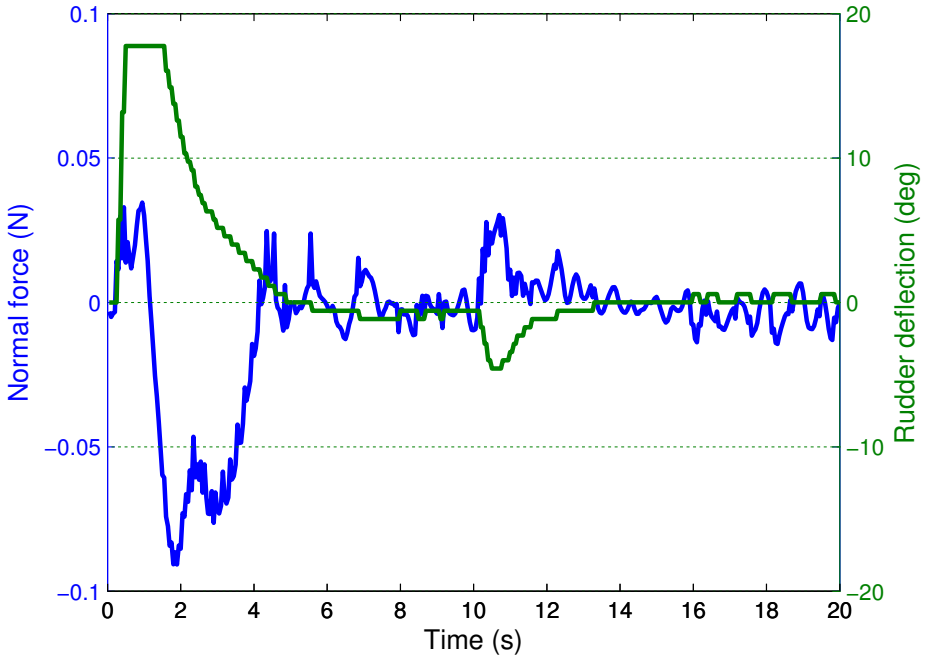


Figure 4.20: Normal force on GPAUV's rudders (Upper and Lower in Figure 3.19) from VFRM simulation of 20° step-turn maneuver. Lower rudder deflection also shown on right axis.

done and such effects were not considered when creating the models used in this study. It appears that while the GPAUV's control surfaces are small relative to the size of the vehicle as a whole, their rotational motion does manage to create significant forces and moments.

4.3.7.2 35° Step-turn

Prescribed Path Comparison Figure 4.22 shows the GPAUV's orientation during the 35° step-turn maneuver. Datasets from an experimental trial, a VFRM simulation and the three state-space models are included. While both the experimental measurements and numerical predictions are composed of discrete data points, all are plotted here with solid lines for improved clarity. Figures 4.22b and 4.22c include a black dotted line to indicate the reference path that the vehicle is attempting to follow.

Overall, agreement between the numerical predictions and the experimental data is fair at best. Issues with the experimental trials that may contribute to discrepancies between the experimental data and numerical predictions are discussed in Section 4.3.8. While all the models show fair agreement with experimental measurements of the GPAUV's roll (see Figure 4.22a), the Nonlinear model captures the low as well as the high frequency oscillations at $2 < t < 8\,\text{s}$. The numerical models all show fair agreement with the experimental measurements of pitch (see Figure 4.22a), especially when considering the initial offset of the

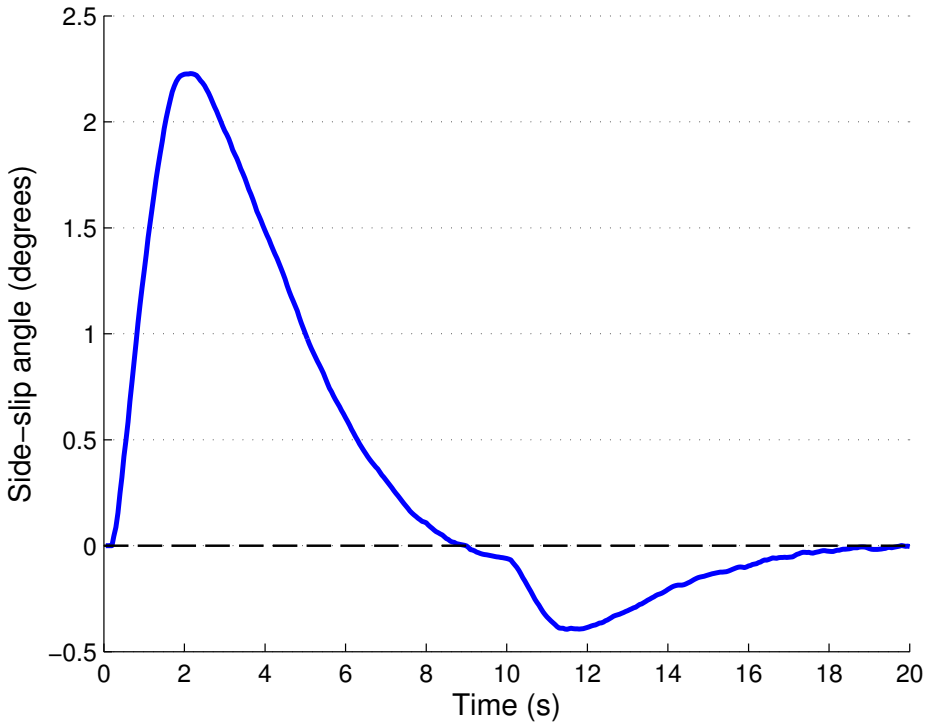
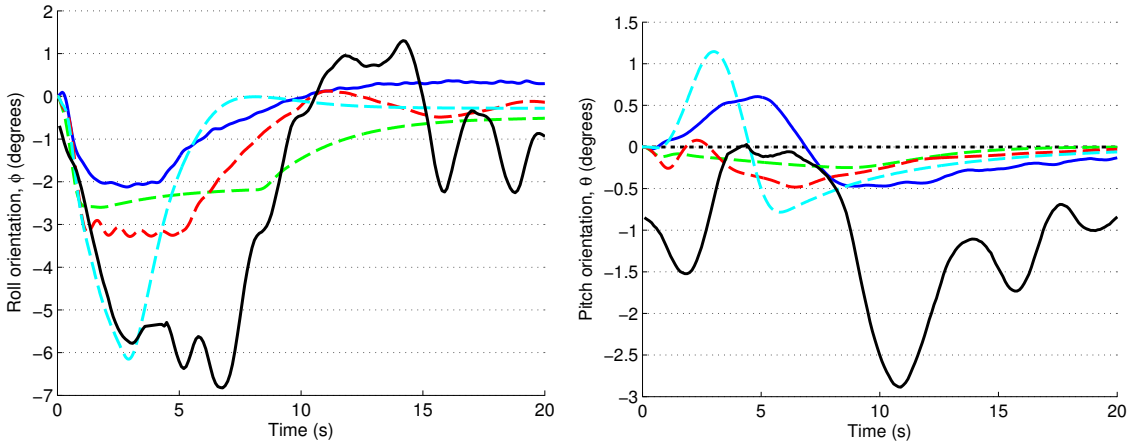


Figure 4.21: GPAUV side-slip angle ($\tan^{-1}(v/u)$) during VFRM simulation of 20° step-turn maneuver.

Table 4.8: Percentage overshoot and damping ratio for 35° step-turn maneuver.

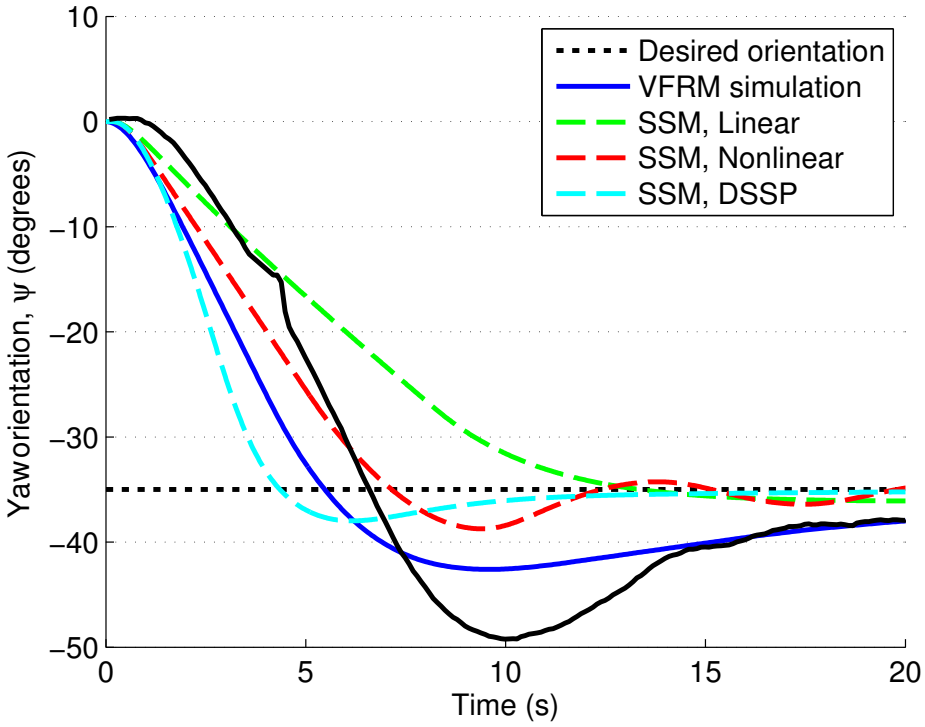
Method	**Percentage overshoot,** PO	**Damping ratio,** ζ
Experimental	33.2	0.331
VFRM simulation	21.7	0.492
Linear SSM	3.09	0.742
Nonlinear SSM	10.6	0.582
DSSP SSM	7.99	0.627

GPAUV in the experimental case. While the overall trends in the yaw heading predictions and measurements shown in Figure 4.22c are fairly similar, the experimental measurements show an overshoot roughly 10% larger than the VFRM simulation and 20% larger than any of the state-space models. The percentage overshoot and damping ratio values for paths predicted by the different methods are shown in Table 4.8.



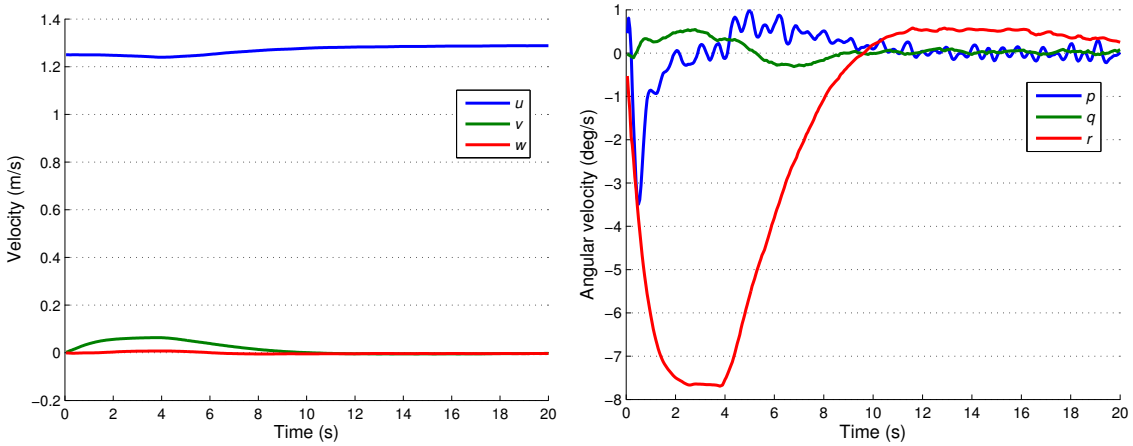
(a) Roll, ϕ

(b) Pitch, θ



(c) Yaw, ψ

Figure 4.22: GPAUV orientation during 35° step-turn maneuver from experimental trial, VFRM simulation and state-space models. (Legend shown in (c) applies to all other sub-figures.)



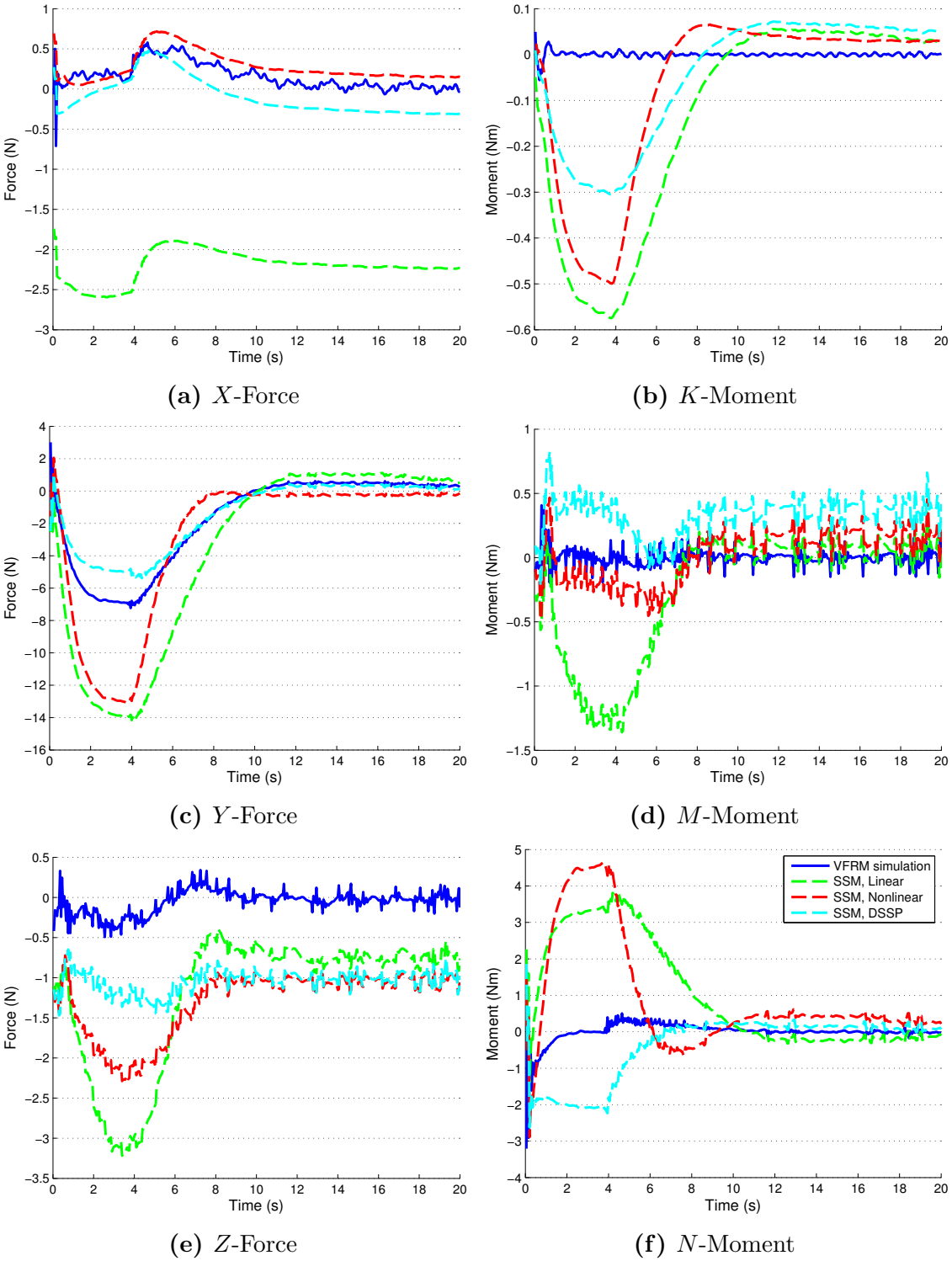
(a) Linear velocities **(b)** Angular velocities

Figure 4.23: Vehicle velocities predicted in VFRM simulation of 35° step-turn maneuver and used as input to state-space models in prescribed motion comparison.

Prescribed Motion Comparison The velocities predicted by prescribed path VFRM simulation and used as the input for prescribed motion evaluations of the state-space models are shown in Figure 4.23. These velocities correspond to the motion plotted for the VFRM simulation in Figure 4.22. While an experimental trial was conducted for this maneuver, no direct measurement of the forces and moments on the vehicle is available. The forces and moments predicted by the VFRM simulation and state-space models for this motion, are shown in Figure 4.24.

The trends seen in the comparison of the 20° maneuver (see Figure 4.19) are also present in this 35° maneuver (see Figure 4.24). Not surprisingly, the differences between the state-space models and the VFRM simulation are larger in the more aggressive 35° maneuver. Much more variability between the state-space models' Y-force predictions is visible in Figure 4.24c than in Figure 4.19c. This is not surprising, as each of the state-space models uses the same added mass coefficients but different damping models. This outcome is highlighted by the difference in motions from each maneuver (Figure 4.18 versus Figure 4.23). The vehicle reaches a prolonged state of steady yaw rotation only in the 35° maneuver ($2 < t < 4\,\mathrm{s}$), thus eliciting a response more directly dependent on viscous damping modeling.

Figure 4.25 shows images from the VFRM simulation throughout the first 10 s of the 35° step-turn maneuver. Transverse planes, colored with the local velocity magnitude, are used to show the shape of the vehicle's wake. Each image was captured from a vantage point defined in the body-fixed frame; thus the "camera" moves with vehicle. The velocity magnitude scalar in these images is taken in the inertial reference frame. An increase in the local flow velocity created by the actuator disk is particularly visible in Figure 4.25a. Note how the port dive plane operates in a more intense wake than the starboard dive plane for much of the maneuver. The local side-slip angle (see Section 3.3.6.1) at the dive planes' axis of rotation

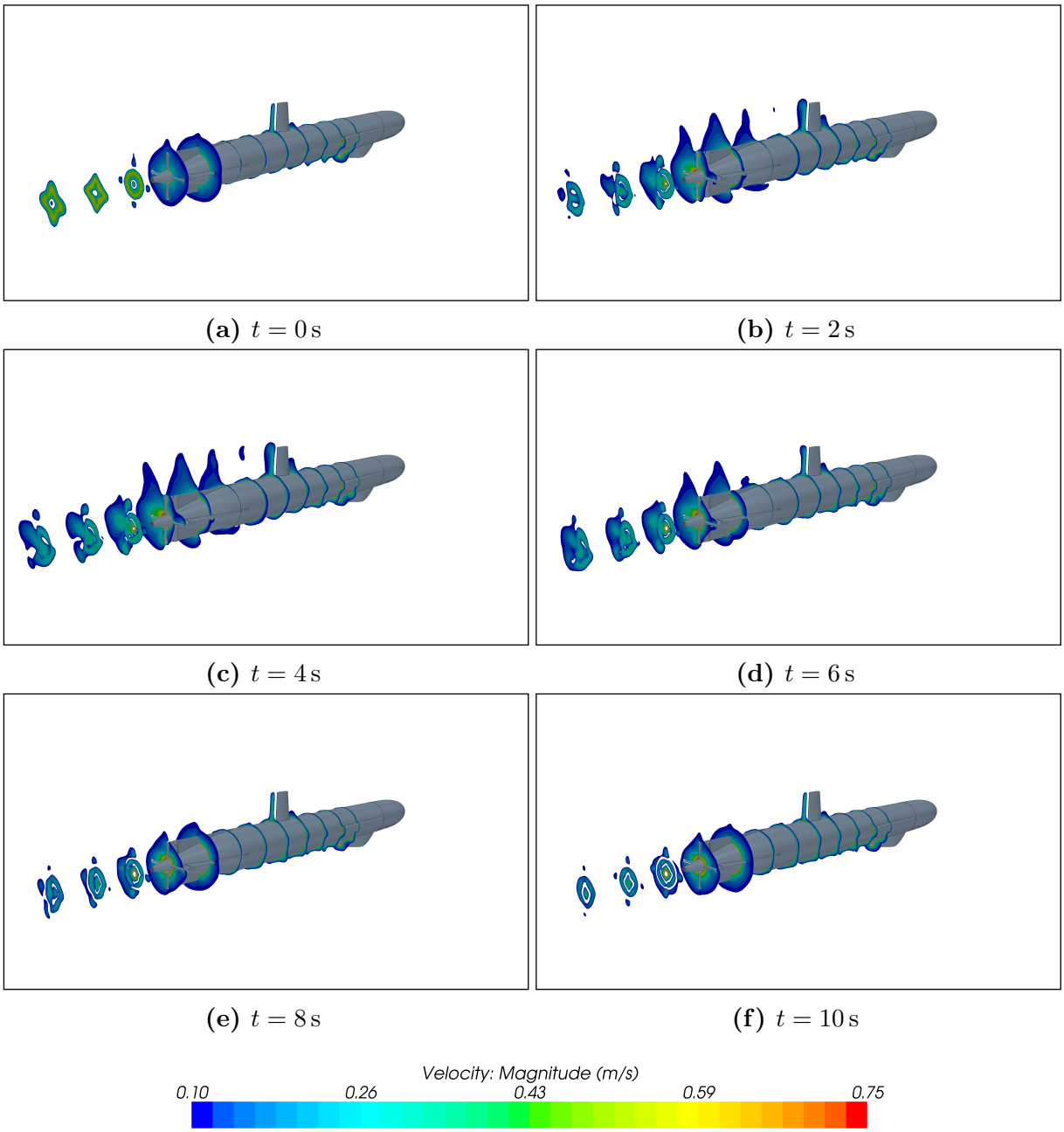


(a) X-Force **(b)** K-Moment

(c) Y-Force **(d)** M-Moment

(e) Z-Force **(f)** N-Moment

Figure 4.24: Forces and moments on GPAUV during prescribed motion 35° step-turn maneuver. (Legend shown in (f) applies to all other sub-figures.)



(a) $t = 0\,\mathrm{s}$ **(b)** $t = 2\,\mathrm{s}$

(c) $t = 4\,\mathrm{s}$ **(d)** $t = 6\,\mathrm{s}$

(e) $t = 8\,\mathrm{s}$ **(f)** $t = 10\,\mathrm{s}$

Figure 4.25: VFRM simulation of 35° step-turn maneuver showing velocity magnitude on transverse planes.

is roughly 6.5° at $t = 4\,\mathrm{s}$ (this instant in the maneuver is shown in Figure 4.25c). An vortex trailing from the tip of the GPAUV's sail is also evident in at this instant.

4.3.7.3 60° Step-turn

Prescribed Path Comparison Predictions from a VFRM simulation and the three state-space models for the GPAUV's position and orientation during a 60° step-turn maneuver are shown in Figure 4.26. Figures 4.26d and 4.26f include a black dotted line to indicate the vehicle's reference path.

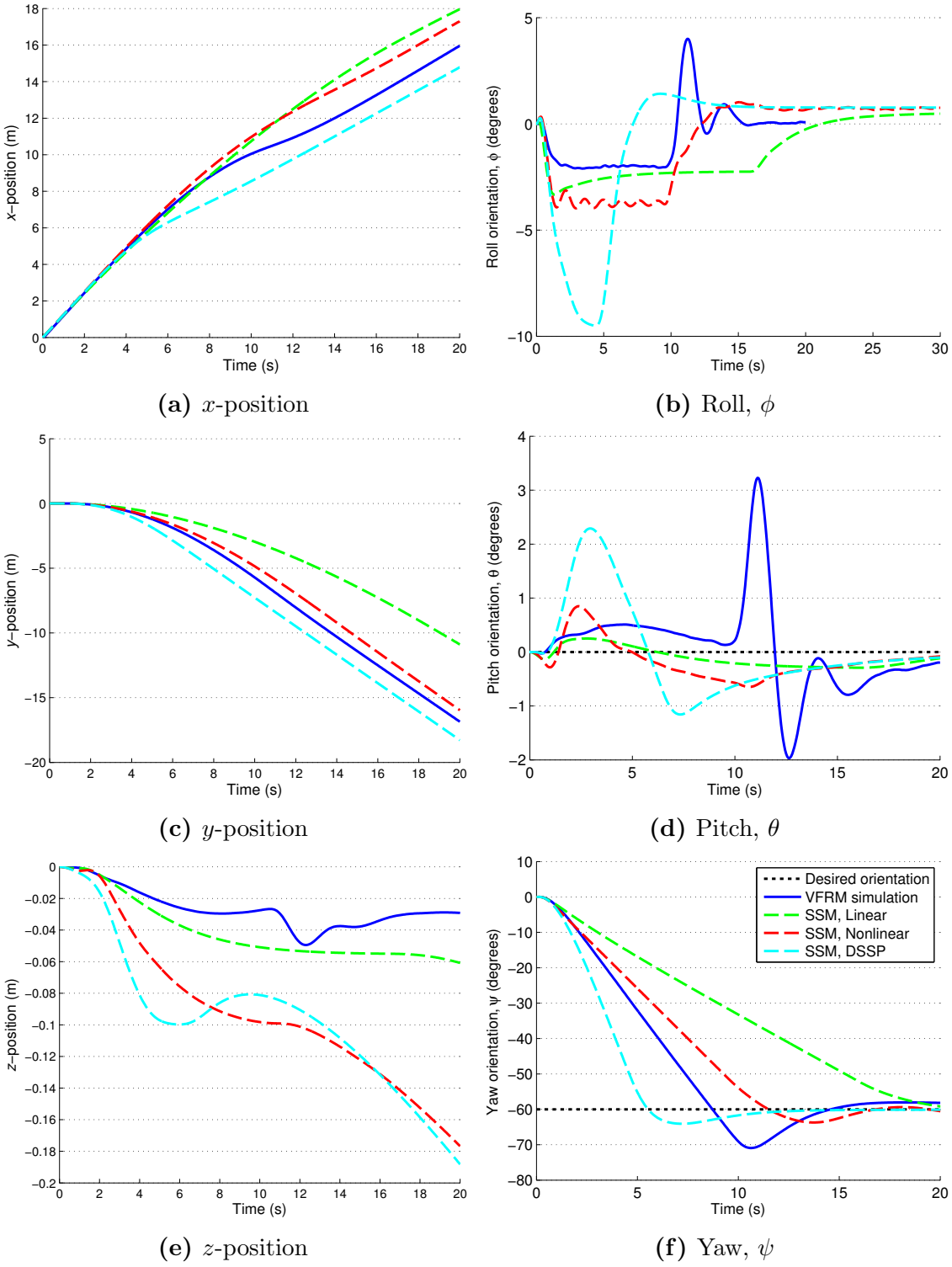
The position plots (Figure 4.26a, c and e) show relatively similar trends. While the vertical motion predictions, shown in Figure 4.26e, are all small in magnitude, an interesting grouping, where the VFRM simulation and Linear model are grouped away from the DSSP and Nonlinear models, is present. Compared to the 20° and 30° step-turn maneuvers shown in Figures 4.15 and 4.22, the numerical predictions shown in Figure 4.26 have substantially larger out-of-plane reactions. The roll magnitude predictions, plotted in Figure 4.26b, have similar trend shapes but with a variety of magnitudes and response frequencies. The large pitch disturbance predicted by the VFRM simulation at $t \approx 10\,\mathrm{s}$ is not seen at all in the results from the state-space models. This disturbance is directly linked to the vehicle's dive-plane deflection, which is plotted for each state-space model and the VFRM simulation in Figure 4.27. This large deflection is a result of the same controller state reset issue discussed in Section 4.3.7.1. The wide variety of dive plane deflection histories shown in Figure 4.27 illustrates the degree to which small differences in these analysis methods can create large differences in vehicle motion and control response. These effects can compound over time to produce vastly different vehicle responses.

From Figure 4.26f, the VFRM simulation and state-space models all appear to predict a path in which the GPAUV achieves a maximum yaw rate early in the maneuver. In fact, by considering the yaw rates plotted in Figure 4.28, one can see that the DSSP model predicts the yaw rate to continue increasing until the rudder's are reversed by the controller at $t \approx 4\,\mathrm{s}$. This may indicate that the terms accounting for yaw damping due to yaw velocity (e. g., $N_{r|r|}, N_r, N_{|v|r}$ and $N_{|r|\delta r}$ in (A.8)) may be too small.

The percentage overshoot and damping ratios computed from the yaw headings predicted for this maneuver are shown in Table 4.9. As the Linear model predicts a relatively small yaw rate (see Figure 4.28), it is not surprising that it has such a small overshoot.

Prescribed Motion Comparison A prescribed motion comparison was conducted for the 60° step-turn maneuver. The motions predicted by the VFRM simulation for the prescribed path comparison were used as the input to evaluate the three state-space models. Figure 4.29 shows the velocities for this motion. The forces and moments predicted by the VFRM simulation and state-space models for this motion are shown in Figure 4.30.

The more aggressive nature of this 60° step-turn maneuver provokes force and moment responses not seen from the 20° and 35° maneuvers. A interesting relation can be made by considering the large discrepancy in yaw moments predicted by the different methods (see Figure 4.30f) in concert with the different yaw rates predicted in the prescribed path



(a) x-position **(b)** Roll, ϕ

(c) y-position **(d)** Pitch, θ

(e) z-position **(f)** Yaw, ψ

Figure 4.26: GPAUV position and orientation during 60° step-turn maneuver from VFRM simulation and state-space models. (Legend shown in (f) applies to all other sub-figures.)

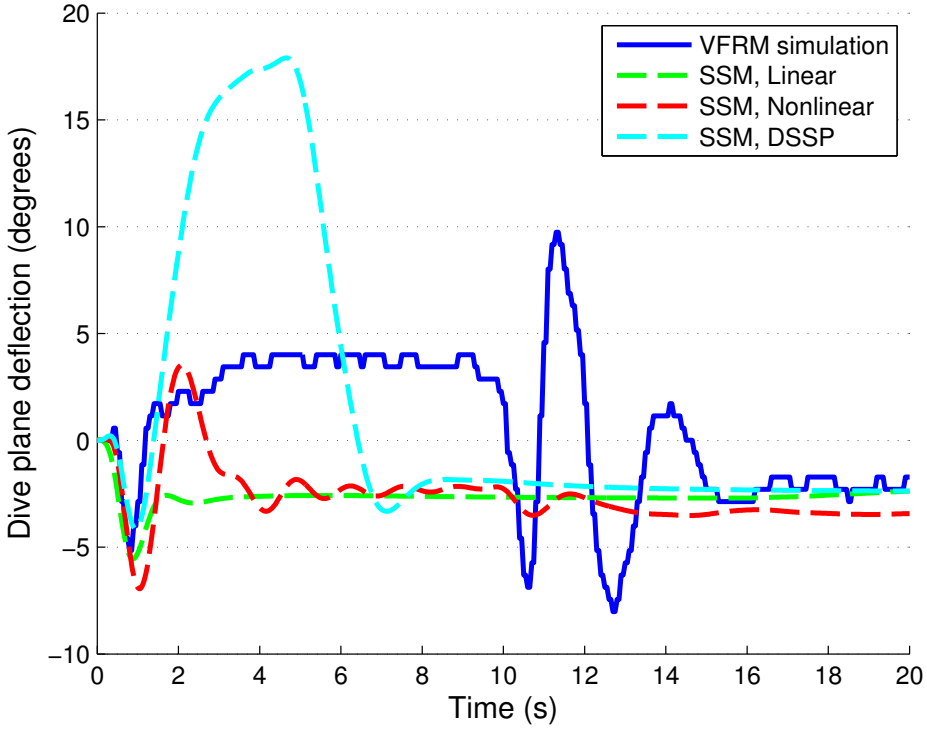


Figure 4.27: Rudder deflection command histories during 60° step-turn maneuver.

Table 4.9: Percentage overshoot and damping ratio for 60° step-turn maneuver.

Method	**Percentage overshoot,** PO	**Damping ratio,** ζ
VFRM simulation	18.2	0.477
Linear SSM	1.80	0.788
Nonlinear SSM	6.19	0.663
DSSP SSM	6.78	0.651

comparison (see Figure 4.26f). During the time period $2 < t < 9\,\text{s}$, when the yaw velocity is relatively constant (this is visible in Figure 4.29b), the VFRM simulation predicts almost no yaw moment ($N \approx 0$). (There is no moment in yaw; therefore no angular acceleration and the yaw rate remains constant.) The DSSP state-space model predicts $N < 0$. Correspondingly, the prescribed path result from the DSSP model has a negative yaw rate of larger magnitude than that of the VFRM simulation. The opposite is true of the Linear and Nonlinear state-space models.

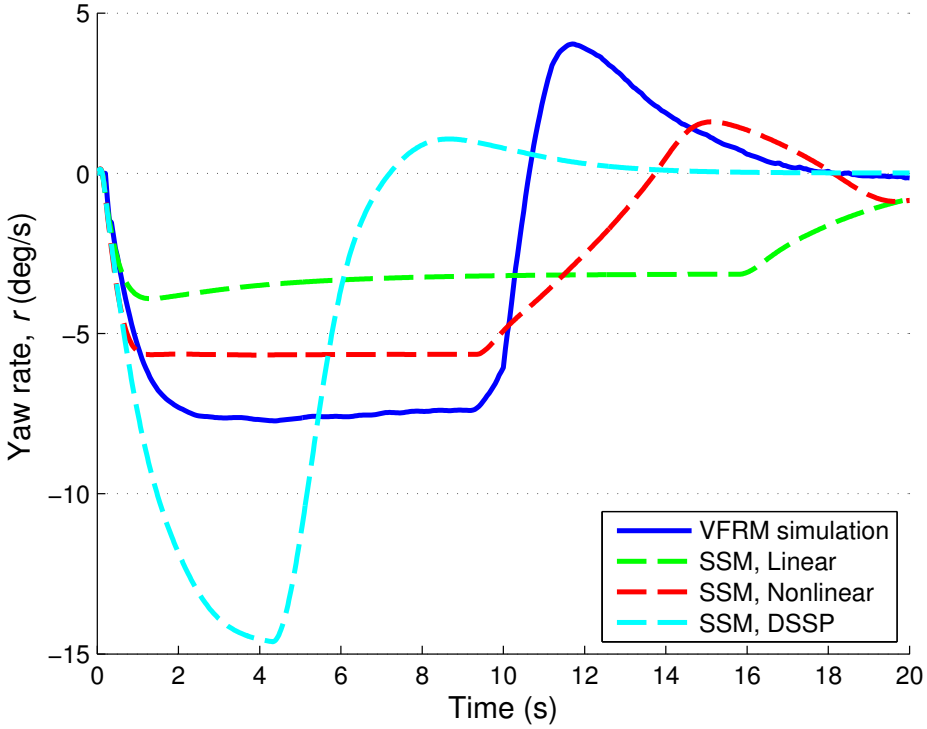
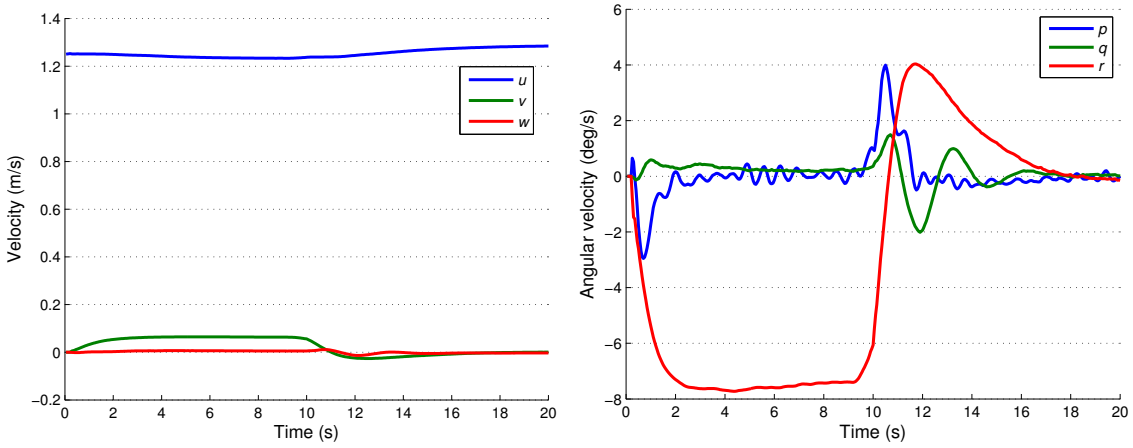
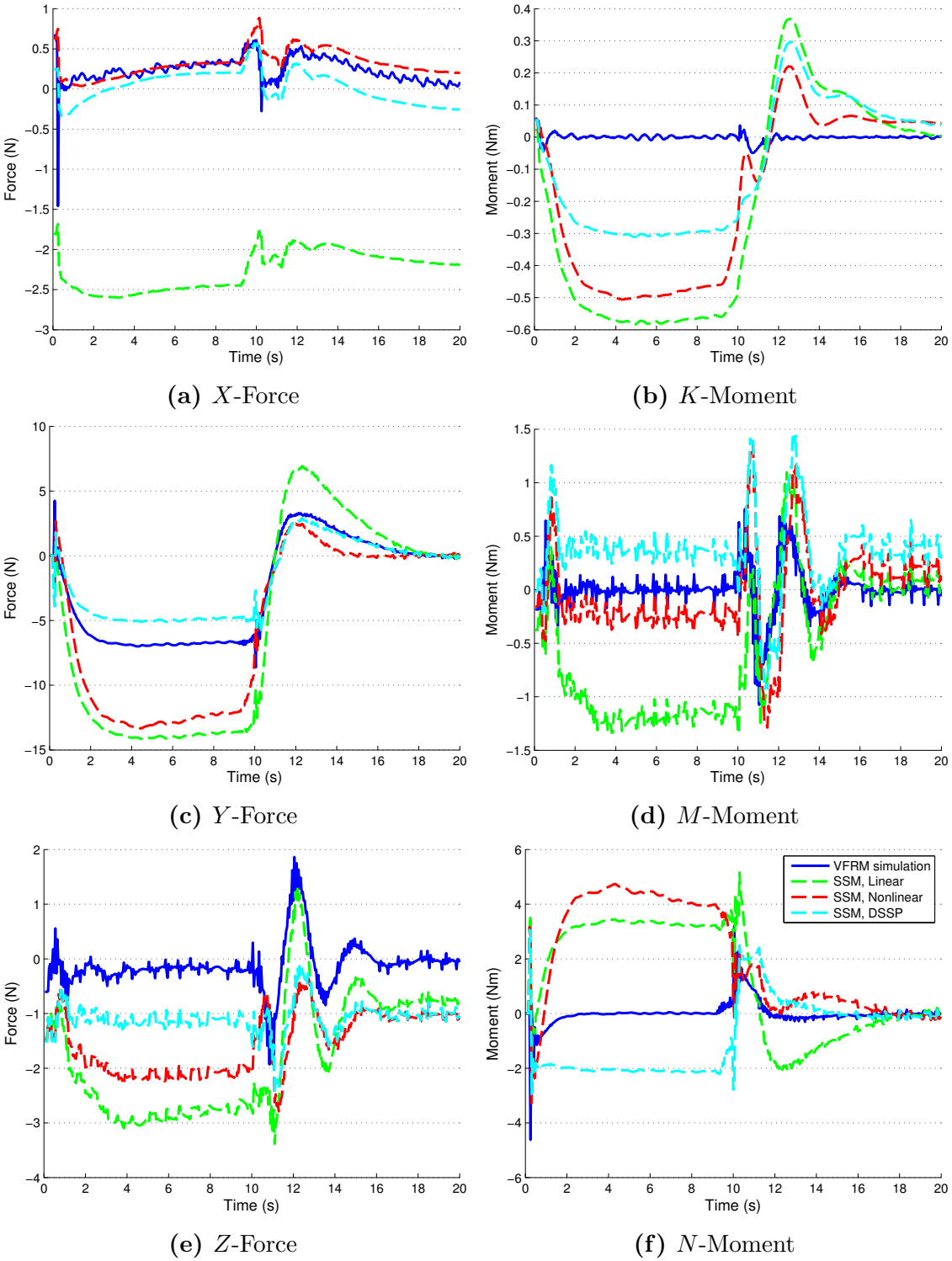


Figure 4.28: Yaw velocity, r, predictions during 35° step-turn maneuver.



(a) Linear velocities **(b)** Angular velocities

Figure 4.29: Vehicle velocities predicted in VFRM simulation of 60° step-turn maneuver and used as input to state-space models in prescribed motion comparison.



(a) X-Force **(b)** K-Moment

(c) Y-Force **(d)** M-Moment

(e) Z-Force **(f)** N-Moment

Figure 4.30: Forces and moments on GPAUV during prescribed motion 60° step-turn maneuver. (Legend shown in (f) applies to all other sub-figures.)

4.3.7.4 0° to -20° Pseudo Zig-Zag

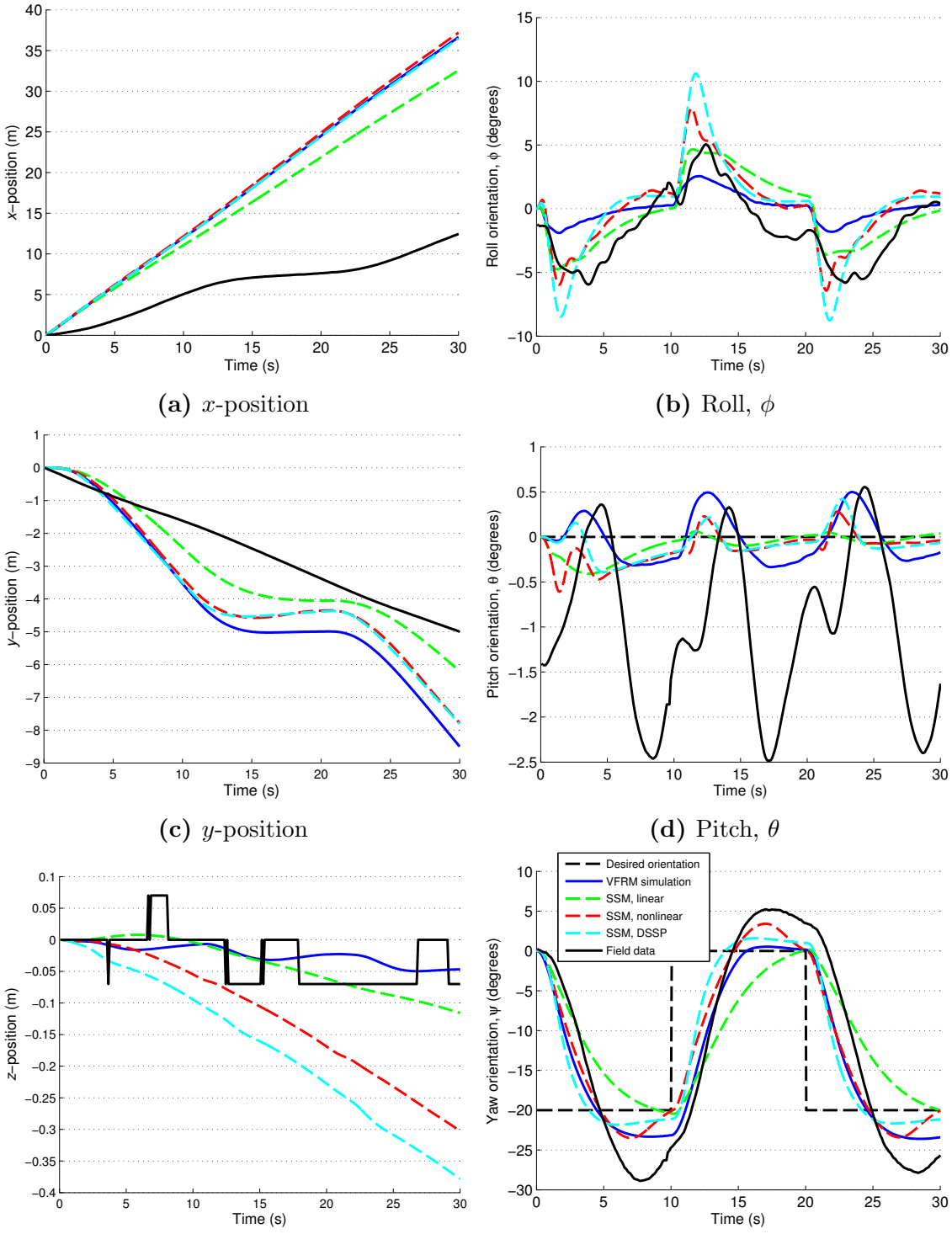
Prescribed Path Comparison Figure 4.31 shows a comparison of the path measured in field experiments for a 20° pseudo zig-zag maneuver with the paths predicted by a VFRM simulation and the three state-space models. A dashed black line is plotted for the pitch and yaw headings to show the reference path. The vehicle is commanded back and forth between yaw headings of $\psi = -20°$ and 0°. The predicted vehicle positions (x, y and z) and orientation (ϕ, θ, ψ) are plotted. Note that the z-position is measured from by the GPAUV using a pressure sensor that operates with a precision of roughly 7 cm. While the sensors operate at the same 10 Hz frequency as the controller, the pressure is recorded at 3.33 Hz.

The orientation predictions, shown in Figures 4.31b, 4.31d and 4.31f, show mixed agreement. The numerical methods capture the vehicle roll relatively well (see Figure 4.31b), but grossly under-predict the magnitude of pitch disturbances (see Figure 4.31d). The yaw heading predictions, plotted in Figure 4.31f, are in fair agreement with the experimental measurements. The VFRM simulation and Nonlinear state-space model predict the time at which the vehicle crosses past -20° to within 5% of the experimental trial, but under predict the overshoot angle.

The position plots (Figures 4.31a, 4.31c and 4.31e) show large discrepancies between the numerical predictions and experimental measurements. This is not surprising as any small difference in the vehicle's forward speed tends to create large differences in position over time. As the GPAUV traveled slower in the experimental trial than the in numerical models ($u \approx 1.11$ m/s experimentally versus $u \approx 1.27$ m/s in the VFRM simulation), the plots for x, and y are quite different.[1] The z-position of the GPAUV is less dependent on this difference in forward speeds; since the vehicle's pitch angle remained relatively small, changes in z were influenced somewhat more equally by u and w. Note however, that as the state-space models do not achieve as much of a nose-down pitch as the experimental trial ($\theta < 0$ in Figure 4.31d), they tend to loose depth over the course of the maneuver ($z > 0$ in Figure 4.31e). The linear and rotational velocities measured in field experiments for the pseudo zig-zag maneuver along with those predicted by the VFRM simulation and state-space models are shown in Figure 4.32. Note that high-frequency oscillations in the linear velocity field data are likely an artifact of measurement from the DVL.

The results shown in Figure 4.32a are somewhat surprising, as the Linear state-space model, which would be expected to perform poorly in predicting axial drag (and therefore forward speed), instead predicts a state history similar to the experimental data. In light of the issues with the experimental trial discussed in Section 4.3.8, this is likely a coincidence. Lower frequency fluctuations in the sway velocities (Figure 4.32c) predicted by the numerical models appear to be of the same order of magnitude as those measured during field trial. The DSSP and Nonlinear state-space models generally show the best agreement with the VFRM simulation's linear velocity predictions. Heave velocity magnitudes (Figure 4.32e) remain

[1] Potential causes of this speed difference are discussed in Section 4.3.8

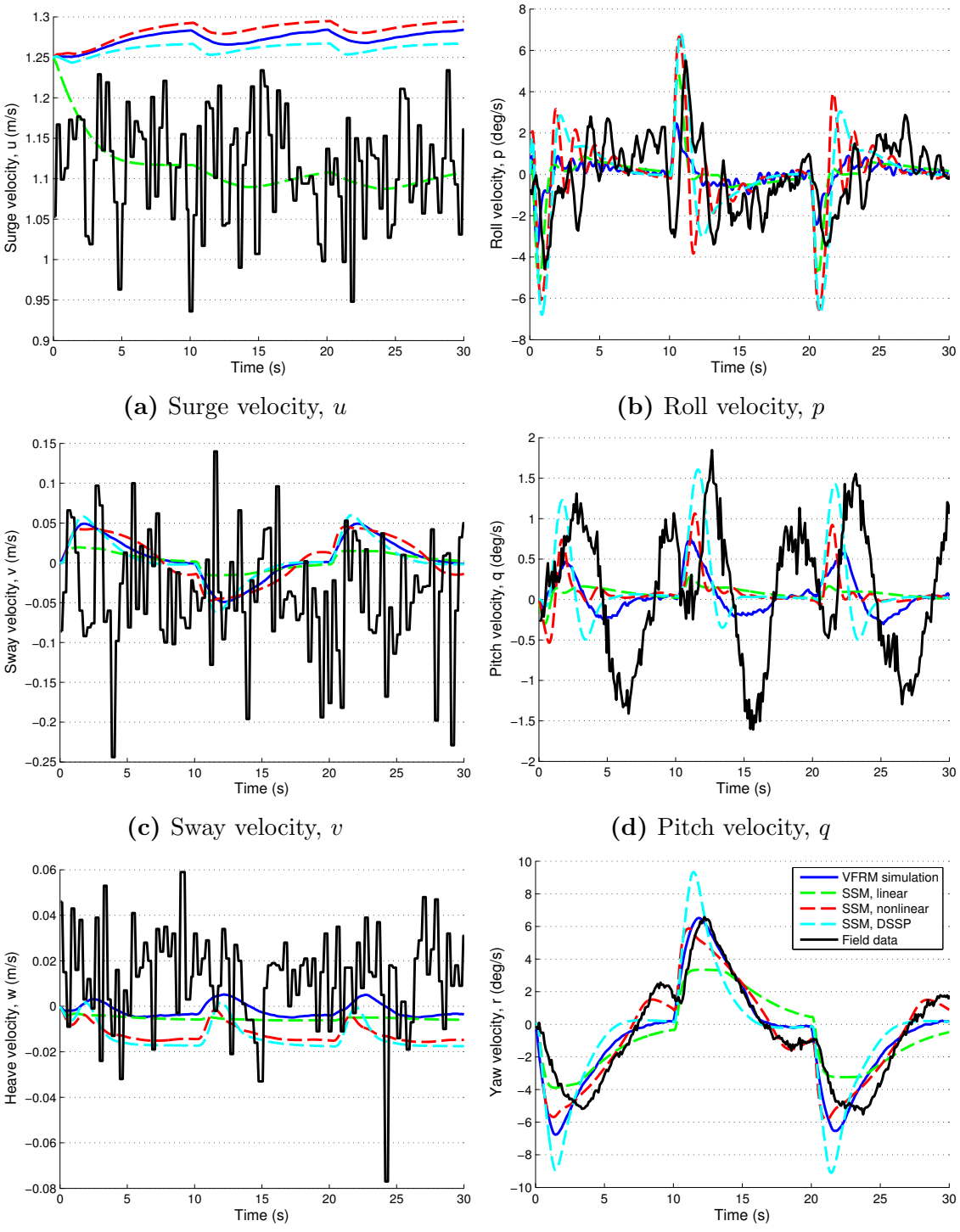


(a) x-position **(b)** Roll, ϕ

(c) y-position **(d)** Pitch, θ

(e) z-position **(f)** Yaw, ψ

Figure 4.31: GPAUV position and orientation during 0° to -20° pseudo zig-zag maneuver from VFRM simulation and state-space models. (Legend shown in (f) applies to all other sub-figures.)



(a) Surge velocity, u

(b) Roll velocity, p

(c) Sway velocity, v

(d) Pitch velocity, q

(e) Heave velocity, w

(f) Yaw velocity, r

Figure 4.32: GPAUV rotational velocities during 0° to -20° pseudo zig-zag maneuver. (Legend shown in (f) applies to all other sub-figures.)

quite small during this maneuver, although the numerical models appear to capture the low frequency oscillation also seen in the experimental data.

Numerical predictions of the vehicle's rotational velocities (see Figures 4.32b, 4.32d and 4.32f) show somewhat better agreement than the linear velocities. Figure 4.32b shows that the numerical methods are able to capture the low frequency trends in roll. The VFRM simulation also exhibits high frequency roll oscillations similar to those seen in the experimental data. The pitch velocity trends measured in the experimental trial are most closely matched by the DSSP and Nonlinear state-space models (see Figure 4.32d), although none of the numerical methods capture the negative pitch motions. The Nonlinear state-space model and VFRM simulation have very good agreement with the experimental data for yaw velocity (see Figure 4.32f). As with any prescribed path comparison, it is important to note that the phase differences in Figure 4.32 are related to factors that compound over time.

Prescribed Motion Comparison A prescribed motion comparison was conducted for the 20° pseudo zig-zag maneuver. The path predicted by the VFRM simulation for this maneuver (see Figure 4.31 and 4.32) was fed into the state-space models to obtain predicted force and moment histories. Figure 4.33 shows these forces and moments.

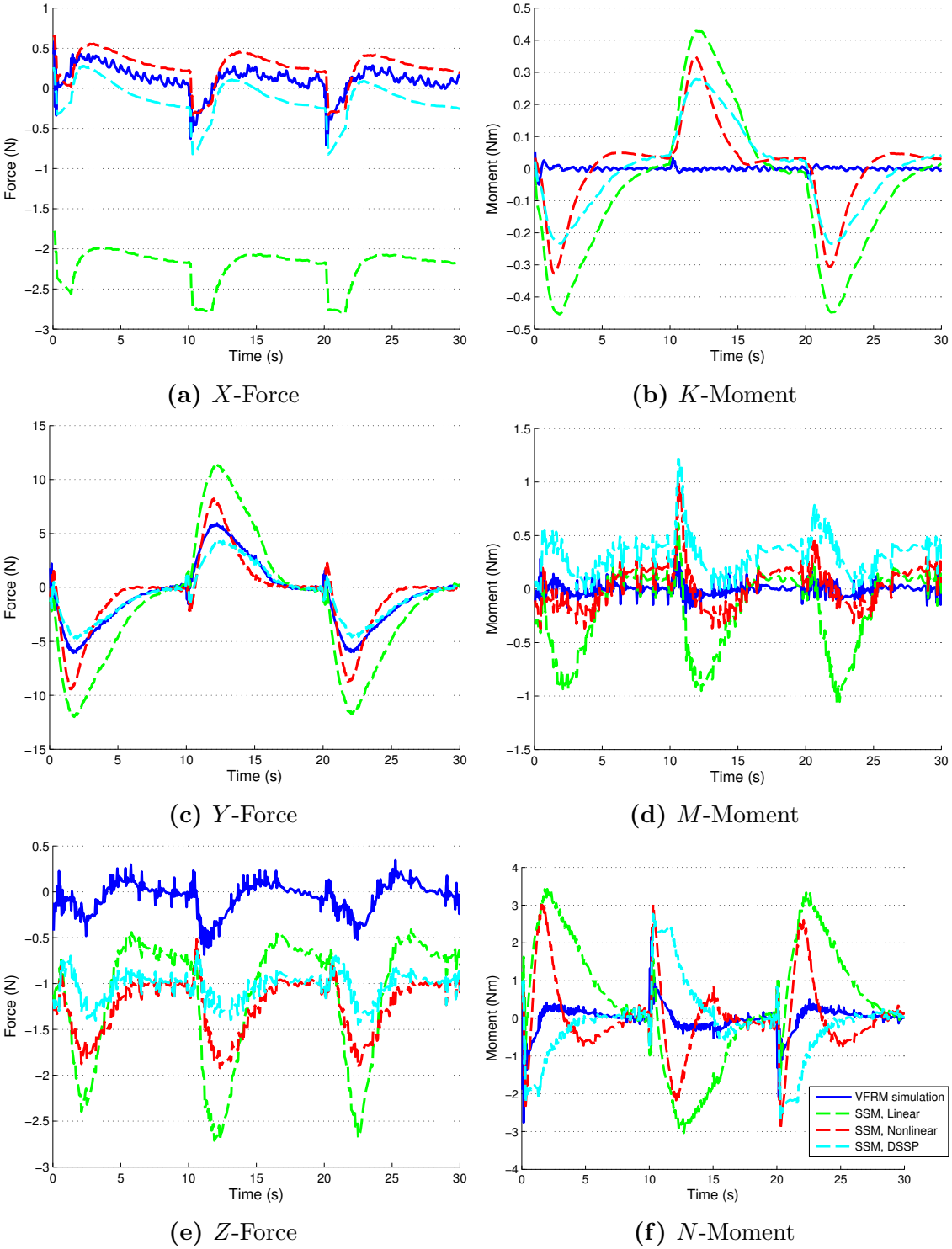
As with the step-turn maneuvers, discrepancies between the different methods are somewhat more pronounced with this prescribed motion mode of comparison than the prescribed path comparison shown in Figures 4.31 and 4.32. Aside from the roll and yaw moments, shown in Figures 4.33b and 4.33f respectively, the state-space models do fairly well at capturing the overall trends from the VFRM simulation. While constant offsets, such as that seen in Z-force in Figure 4.33e, are not insignificant, they are indicative of a poorly defined coefficients in a well formulated model.

4.3.7.5 60° to -60° Pseudo Zig-Zag

Prescribed Path Comparison Figure 4.34 shows the paths predicted by a VFRM simulation and the three state-space models for a 60° to -60° pseudo zig-zag maneuver. A dashed black line is used to show the reference pitch and yaw headings in Figures 4.34d and 4.34f.

This aggressive maneuver produces divergent behavior from the DSSP state-space model. While the origin of this behavior is currently unknown, the issue may be related to an under prediction in yaw damping as discussed in Section 4.3.7.3. From Figure 4.34f, it is evident that the model was predicting unrealistically large yaw velocities and accelerations.

Considering the tendency of errors to compound, the x and y-position predictions, shown in Figures 4.34a and 4.34c respectively, are in fairly good agreement. Both the Nonlinear and Linear state-space models predict larger decreases in depth ($z < 0$) than the VFRM simulation. The Nonlinear model predicts a depth change of roughly 0.5 m over the course of the 30 s maneuver.

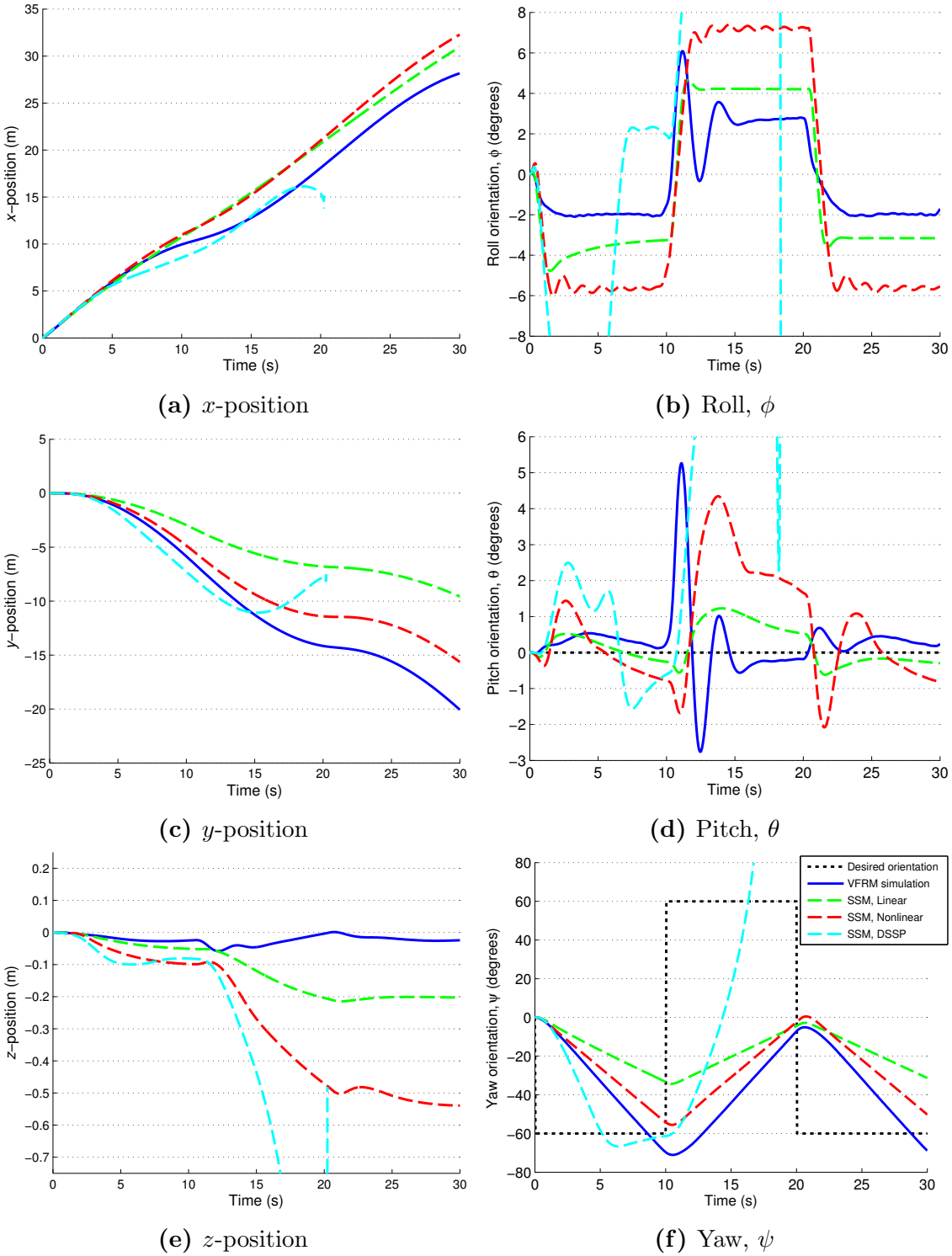


(a) X-Force **(b)** K-Moment

(c) Y-Force **(d)** M-Moment

(e) Z-Force **(f)** N-Moment

Figure 4.33: Forces and moments on GPAUV during 0° to -20° pseudo zig-zag maneuver. (Legend shown in (4.33f) applies to all other sub-figures.)



(a) x-position

(b) Roll, ϕ

(c) y-position

(d) Pitch, θ

(e) z-position

(f) Yaw, ψ

Figure 4.34: GPAUV position and orientation during 60° to -60° pseudo zig-zag maneuver from VFRM simulation and state-space models. (Legend shown in (f) applies to all other sub-figures.)

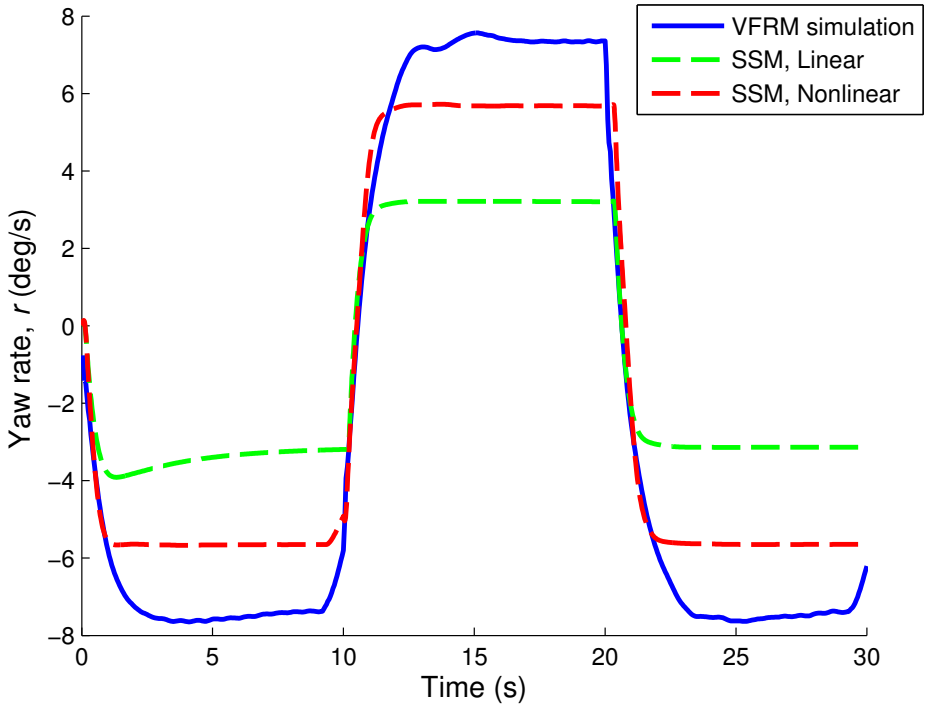


Figure 4.35: GPAUV yaw velocity, r, during 60° to -60° pseudo zig-zag maneuver from VFRM simulation and state-space models (the DSSP state-space model diverges for this maneuver and is omitted in this plot).

As with the 60° step-turn maneuver, the state-space models predict slightly larger roll motions for this maneuver than the VFRM simulation (see Figure 4.34b), although agreement between the VFRM simulation and Linear state-space model is quite close. In the pitch orientation predictions shown in Figure 4.34d, the three methods return a variety of trends. The VFRM simulation shows the same pitch disturbance and ringing at $10 < t < 15$ s as seen in the 60° step-turn maneuver. This is a result of the controller state reset issue discussed in Section 4.3.7.1. In Figure 4.34f, the Linear and Nonlinear state-space models predict yaw heading trends similar to the VFRM simulation, but find different maximum heading change rates. The yaw rate, r, predicted during the 60° to -60° pseudo zig-zag maneuver by each method is shown in Figure 4.35.

Figure 4.36 which shows the local flow velocity on transverse planes near the vehicle 20 s into the 60° pseudo zig-zag maneuver. At this point in the maneuver, the vehicle is traveling forward a 1.23 m/s and yawing at a rate of 7.36 deg/s. That yaw velocity is decreasing at a rate of 12.5 deg/s^2. Figure 4.36 clearly shows asymmetric wake that encompasses the vehicle's control surfaces at this point in the simulation.

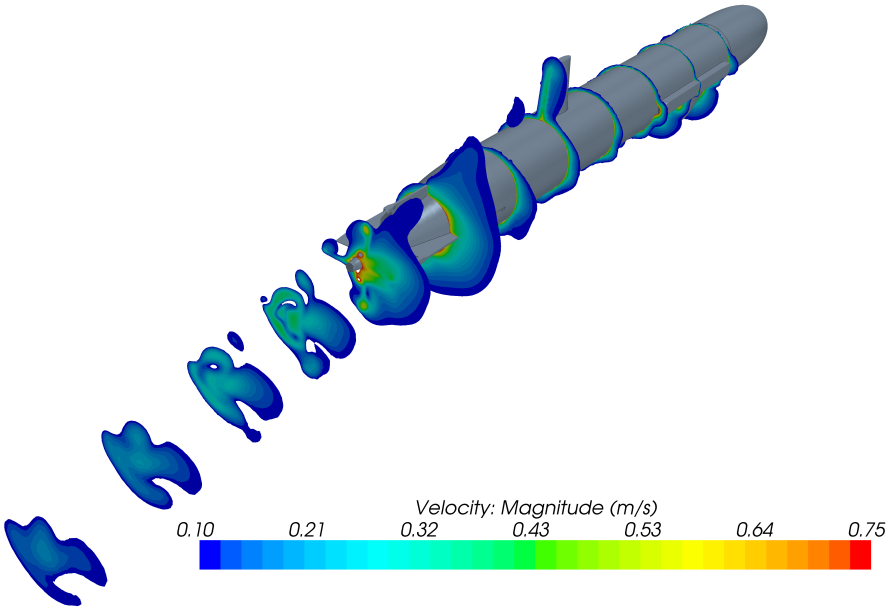
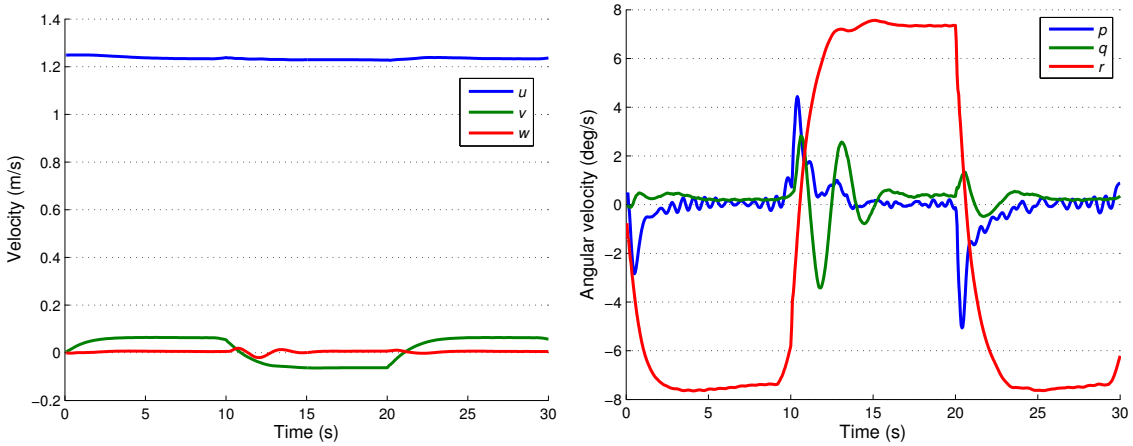


Figure 4.36: VFRM simulation of 60° pseudo zig-zag maneuver at $t = 20\,\mathrm{s}$ showing velocity magnitude on transverse planes.

Prescribed Motion Comparison Figure 4.37 shows the velocities from the prescribed path VFRM simulation of the 60° to -60° pseudo zig-zag maneuver which served as input for the state-space models in this prescribed motion comparison. The forces and moments predicted by the VFRM simulation and state-space models for this motion are shown in Figure 4.38. As with the other maneuvers analyzed in this chapter, discrepancies that are evident in the prescribed path comparison are even more magnified in the corresponding prescribed motion comparison. Many of the same conclusions drawn from the 60° step-turn (see Section 4.3.7.3) can be made for the results shown in Figure 4.38.



(a) Linear velocities **(b)** Angular velocities

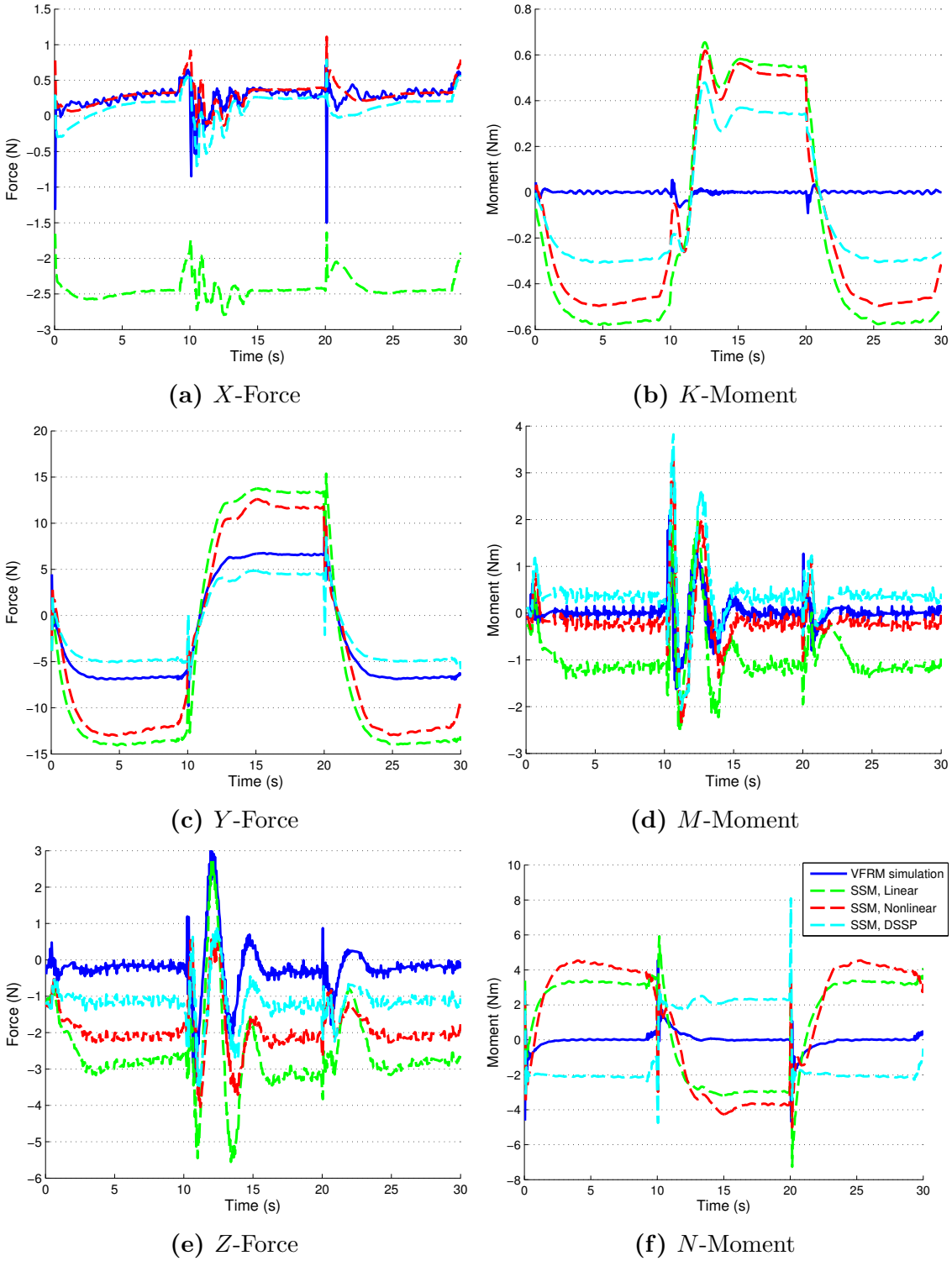
Figure 4.37: Vehicle velocities predicted in VFRM simulation of a 60° to -60° pseudo zig-zag maneuver and used as input to state-space models in prescribed motion comparison.

For the most part, the trends seen in the 60° step-turn maneuver shown in Figure 4.30 are present and repeated for this pseudo zig-zag maneuver. The X-force prediction from the VFRM simulation shows very large force spikes that coincide with the orientation changes at $t = 10$ and 20 s (see Figure 4.38a). These may be linked to the dynamic motion of the GPAUV's rudders. The rapid change in the vehicle side-slip angle shown in Figure 4.39 are also likely to have created large changes in the vehicle's wake structure and therefore the drag force.

4.3.8 Issues with the Experimental Trial Data

One major issue present only in the experimental trials, was an unexplained static offset of the GPAUV's rudders during operation. Figure 4.41 shows the commanded rudder deflection during a steady flight leading up to the pseudo zig-zag maneuver ($-100 \leq t \leq 0$ s). While the top-bottom asymmetry and positive buoyancy of the GPAUV necessitates a nonzero offset of the vehicle's dive planes, the offset of the vehicle's rudders is concerning. Potential causes for this offset include small asymmetries in the installation of appendages.

To facilitate deployment and recovery of the GPAUV during testing, the D-rings shown in Figure 4.40 were affixed to the body. These rings were not modeled in any of the numerical analyses. Although these D-rings are relatively small, there is some concern that they may have contributed to an asymmetric drag on the vehicle. Closer examination of the field trial's rudder command history during the zig-zag maneuver, also in Figure 4.41 ($0 \leq t \leq 40$ s), shows an asymmetry in rudder deflections as well. The GPAUV must work harder, keeping the rudder near its maximum deflection of 18° for a longer period of time, to achieve a negative yaw rate than it does to achieve a positive rate.



(a) X-Force **(b)** K-Moment

(c) Y-Force **(d)** M-Moment

(e) Z-Force **(f)** N-Moment

Figure 4.38: Forces and moments on GPAUV during prescribed motion 60° to -60° pseudo zig-zag maneuver. (Legend shown in (f) applies to all other sub-figures.)

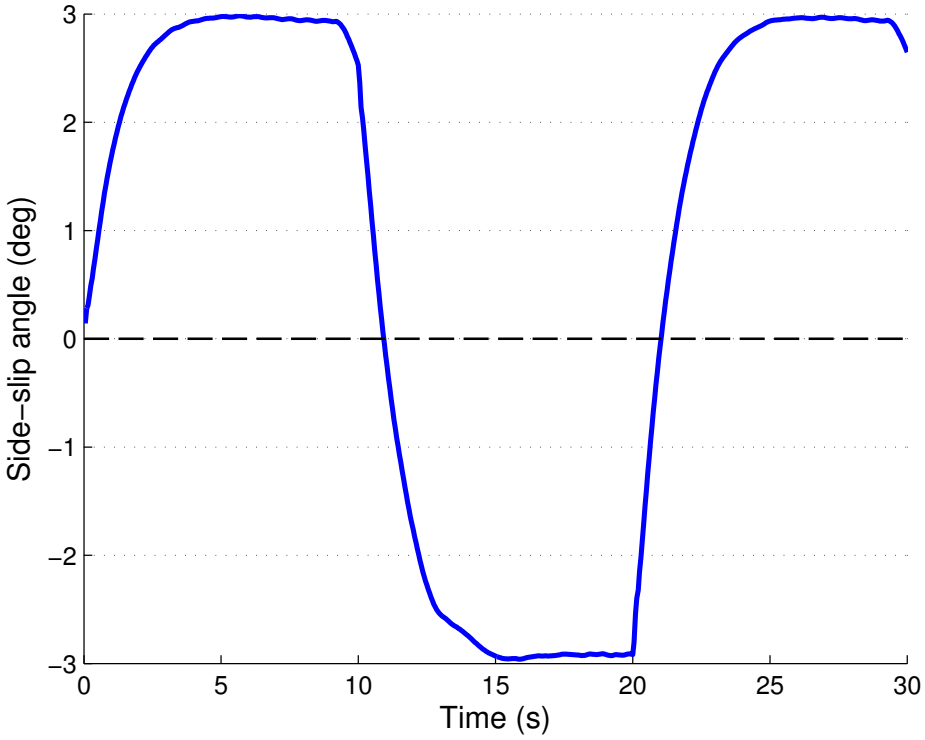


Figure 4.39: GPAUV side-slip angle ($\tan^{-1}(v/u)$) during VFRM simulation of 60° pseudo zig-zag maneuver.

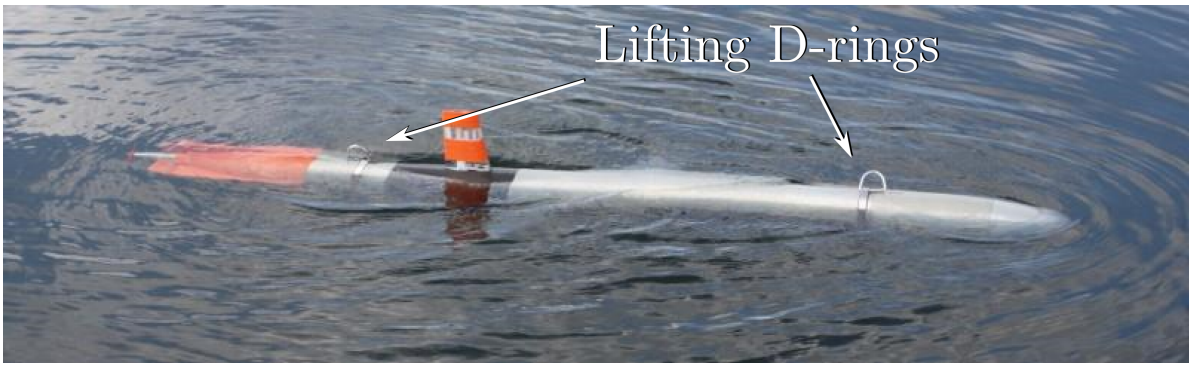


Figure 4.40: GPAUV during test in Claytor Lake, VA.

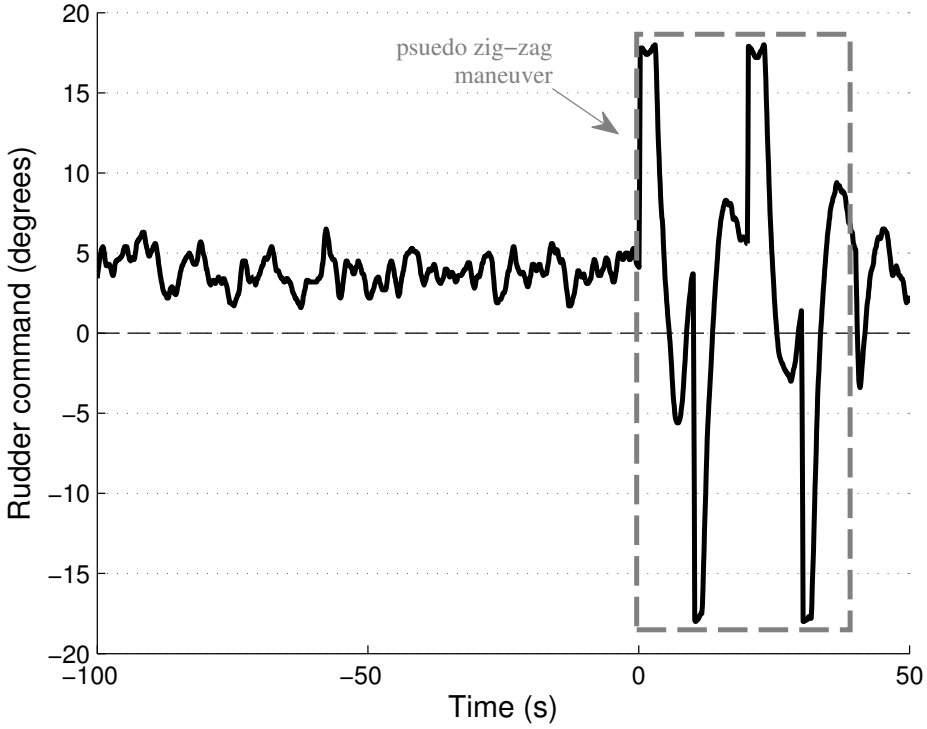


Figure 4.41: Field GPAUV rudder command during steady operation showing constant offset of approximately 4°.

Uncertainty in the GPAUV's rigid-body inertial properties used in VFRM simulations and the state-space models was initially considered to be a potential source for the discrepancies seen between the numerical predictions and experimental measurements. The rigid-body inertial parameters were estimated from a weights spreadsheet, but were never measured directly from the completed GPAUV. A qualitative analysis of the paths shown in Figures 4.22 and 4.31 seems support the hypothesis that the inertial properties of the actual GPAUV may be larger than the values used in the numerical models. Subsequent analysis of the GPAUV's internal configuration has revealed that the yaw moment of inertial (I_{zz} in (A.2)) is likely to have been underestimated by 6%.

To better understand the sensitivity of the maneuvering predictions presented in this study to rigid-body inertial properties, the 35° step-turn maneuver was rerun with the Nonlinear state-space model, using $\mathbf{I}_{RB} = \mathbf{I}_{RB_0} \pm 0.5\mathbf{I}_{RB_0}$. The yaw paths from this analysis are compared with those from the experimental trial are shown in Figure 4.42. It is clear that the path predicted by the model is not particularly sensitive to the rigid-body inertial properties. Although this result is somewhat disappointing, as it rules out a relatively simple solution for the discrepancies seen between the numerical predictions and experimental data, it actually bodes well for the utility of VFRM simulations in general. As accurately defining a vehicle's rigid-body inertial properties is not feasible early in the design process, a high sensitivity to these parameters would make any superiority in a VFRM simulation's ability to capture complex fluid dynamic phenomena moot.

4.3.9 Conclusion

Although agreement of the paths predicted during VFRM simulations, state-space models and experimental field trials is not as good as might be expected, these relatively large discrepancies may be due to problems with the experimental trial (see Section 4.3.8). Unfortunately no additional experimental data for the GPAUV is currently available. Additional testing and comparisons will be necessary to provide more conclusive answers about the relative accuracy of these different methods.

This lack of experimental data for the GPAUV limits the degree to which the accuracy of the three state-space models can be judged. However, if VFRM simulations are considered to be a "nearer-to-reality" comparison point, the Nonlinear model would rank above the DSSP and Linear models in accuracy. The Nonlinear model's formulation was conceived during the course of this study based on a conceptual analysis of vehicle symmetries (see Section 3.3.4.1 for an in-depth explanation). The success of a model that uses a nonlinear representation of viscous damping over a model that uses a linear representation of viscous damping is not particularly surprising. In fact, the Linear model performs better than might be expected.

Aside from the divergence issues shown in the 60° pseudo zig-zag maneuver (see Section 4.3.7.5), the DSSP model performed quite well, especially when considering that the computational time required to produce it is orders of magnitude less than the CFD-based

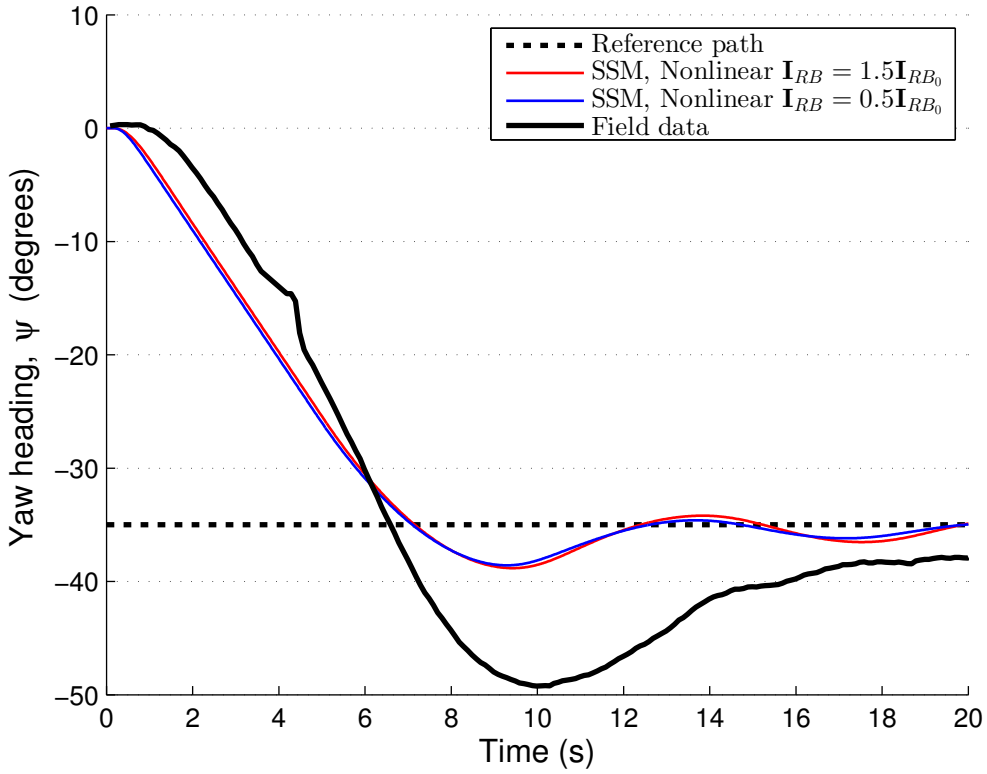


Figure 4.42: GPAUV heading during 35° step-turn maneuver from the Nonlinear state-space model with $\mathbf{I}_{RB} = \mathbf{I}_{RB_0} \pm 0.5\mathbf{I}_{RB_0}$ and experimental trial.

models (seconds for DSSP, hours for the CFD-based models). The DSSP model's divergence seems to be at least partially related to its prediction of unrealistically large yaw velocities and accelerations. This behavior may be linked to the way in which the GPAUV's external appendages were defined within DSSP. As discussed in Section 3.6.2, all of the vehicle's external appendages were defined as lifting surfaces. While the GPAUV's sail and fixed strakes fit well within the classification of lifting surfaces, this representation of the appendage fairing and side scan sonars is not quite as accurate. DSSP may be modeling these appendages as lifting surfaces when in reality they act more like bluff bodies. Flow regime scaling may be another issue, as DSSP is designed for analysis of full-scale submarines ($1 \times 10^7 \lesssim Re \lesssim 1 \times 10^9$) but is used in this study to analyze an AUV with $Re \approx 4.5 \times 10^6$.

Table 4.10 shows the mean values and standard deviations (in parentheses) of the percentage overshoot and the damping ratio for each method. The mean and standard deviation are taken over the results from the three step-turn maneuvers (20°, 35° and 60°). From the values in Table 4.10, the Nonlinear model is the most similar to the VFRM simulation, followed by the DSSP and Linear models. However, the Nonlinear model's parameters showed a large degree of variance between the three step-turn maneuvers.

Table 4.10: Mean and standard deviation for the percentage overshoot and damping ratio values from 20°, 35° and 60° step-turn maneuvers.

Method	**Percentage overshoot,** PO	**Damping ratio,** ζ
VFRM simulation	18.9 (2.48)	0.487 (0.009)
Linear SSM	3.42 (1.80)	0.737 (0.053)
Nonlinear SSM	11.5 (5.75)	0.576 (0.090)
DSSP SSM	8.06 (1.32)	0.626 (0.025)

An interesting comparison can be made between the agreement in force predictions for the prolate spheroid and the GPAUV. The state-space model force predictions for the spheroid match much better for the spheroid than for the GPAUV. This suggests that the more complex geometry of the GPAUV, with its more angular surfaces that tend to induce flow separation, as well as the addition of actual control surfaces, makes creating a state-space model more challenging. Another factor that likely contributes to this difference is the aggressiveness of the maneuvers. While a 4° step-turn maneuver was analyzed for the spheroid, 20°, 35° and 60° maneuvers were considered for the GPAUV.

Chapter 5

Discussion

In this dissertation, research has been presented on many of the components essential to the process of modeling and analyzing the maneuvering performance of fluid-based vehicles. A survey of the state-space models traditionally used in this practice, as well as the underlying theories on which they are based, informed the creation of a semi-novel nonlinear formulation to model viscous damping. This formulation and other similar formulations were used to create state-space models to predict the maneuvering performance of the GPAUV. Two approaches were employed to determine the coefficients that comprise these state-space models: (1) a semi-empirical database regression and (2) a system identification process that is based on the results of CFD simulations. The later of these methods, while orders of magnitude more computationally expensive than the former, is applicable to novel vehicle designs that are not well represented by the available databases. Although other researchers have used CFD-based approaches to determine the parameters for a state-space model, few have performed the in-depth investigation of the factors important to this process that was completed for this dissertation.

VFRM simulations, meant to serve as a high-fidelity maneuvering analysis approach, were developed by coupling unsteady CFD simulations with a rigid-body kinematic model and a vehicle control algorithm. The CFD methodologies employed in these simulations are supported by a series of validation studies with comparison to experimental measurements. An overset mesh approach was used to allow for dynamic deflection of an AUV's control surfaces during a maneuvering. To assess the relative performance of each state-space model and of the VFRM simulations, predictions for a series of maneuvers were computed and compared. A subset of these maneuvering predictions were also compared with data from experimental trials conducted with the GPAUV.

5.1 Suggestions for Future Work

5.1.1 Improvements to VFRM Simulations

5.1.1.1 Adaptive Time-stepping

The computational expense of a VFRM simulation is directly proportional to the time-step size used for the CFD simulation. This value is partially limited by the numerical stability of the fluid dynamic solution, but also by the need to provide adequate temporal resolution of dynamic phenomena (fluid and rigid-body). By allowing the time-step size of a VFRM simulation to vary dynamically, based on some criteria set, a VFRM simulation could lead to major computational savings. The criteria set, which would likely be tunned to the operating conditions of a vehicle, might include rigid body acceleration, change in acceleration, local flow velocity, local flow acceleration or the solution's iterative residuals. One complication to the implementation of adaptive time-stepping is the need to maintain consistency of the virtual actuator's performance and the control algorithm's operating frequency (these issues are discussed in Section 4.1.1).

5.1.1.2 Overset Mesh Operations

The single most computationally expensive aspect of the VFRM simulations presented in this study is the use of an overset mesh configuration to allow for the dynamic deflection of the GPAUV's control surfaces. The need to use very high mesh refinement (much higher than would be necessary to capture local gradients) in the gaps between the control surfaces and adjacent geometry is not ideal. In addition to dramatically increasing the total number of finite volume cells in the simulation's computational domain, the overset region enclosing each control surface requires additional operations for hole-cutting and interpolations.

Unnecessary Operations at Inner Iterations One issue for this research project, which is likely to be resolved in the near future, was STAR-CCM+'s solution process in which these overset mesh operations are carried out for every inner-iteration. While this would be necessary to support an implicit solution process if the motion of the control surfaces were dictated by fluid dynamic forces, it is completely unnecessary for this application. Representative's from CD-adapco have been made aware of this issue and are currently working to correct it. Although no precise quantification is available for the solution speed increase that will result when this error is corrected, it will likely be on the order 50%.

Partitioning Another issue with any multi-region CFD simulation is that of partitioning and scalability. For a computational task to scale well, and run efficiently on an increasing

number of processors, the computational workload must be well distributed amongst the processors (i. e., if one set of processors must "wait" for a second set of processors to finish their work, the system will be unbalanced, and therefore inefficient). For parallel CFD simulations, groups of cells within the domain must be distributed to different processors. Workload balancing issues can occur if certain groups of cells require more computation than others.

As previously discussed in Sections 2.8 and 4.3.3, simulations using overset mesh regions require computational operations specific to the overset mesh procedure. The need to perform these overset mesh operations create an increased workload on small groups of cells within the domain. If this spatially uneven workload is not accounted when the domain is partitioned, a poorly balanced system can result. As STAR-CCM+ does not currently allow for user configuration of domain partitioning, an extended analysis of this issue was not conducted for this study.

5.1.1.3 Simulated Sensor Noise

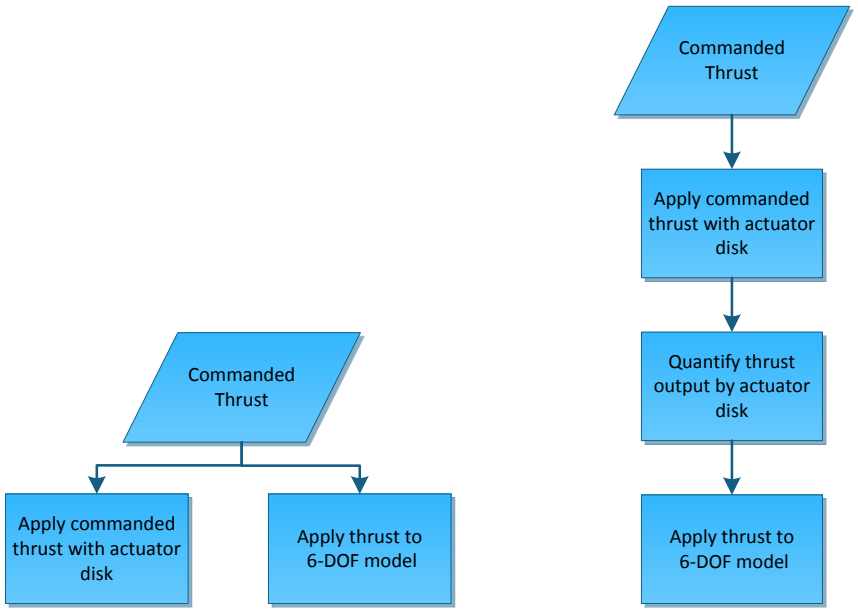
While not employed for models in this study, some state-space models include sub-models to simulate sensor noise so that its effect on a control system can be analyzed [122, 123]. Gaussian noise distributions are added to "measurements" of the vehicle's state to create more realistic simulations. In response to this noise, a state estimation module may be added as well. Noise and state estimation modules could be added to VFRM simulations in this same manner.

5.1.1.4 Self-monitoring Actuator Disk

The VFRM simulations in this study used a split approach to modeling the propulsion system of the GPAUV (see Figure 5.1a). A thrust and torque determined from propeller performance curves were applied separately to the rigid-body kinematic model and the actuator disk. A better estimate of the instantaneous thrust produced by the propeller could be obtained using a method similar to that used in Section 2.7.2.6. In this approach, depicted in Figure 5.1b, the control volume method described in Section 2.7.2.6 would be used to monitor the thrust output by the actuator disk, and apply that value to the rigid-body kinematic model.

5.1.1.5 Equation of Motion for Control Surfaces

A relatively simple differential equation to govern the motion of a vehicle's control surfaces could be introduced to reduce disturbances created by unrealistically large accelerations (see Section 4.3.7.1). Watt presents a second order representation of control surface motion for use in state-space models [124]. Watt's model is based on a sub-critical control system damping and control system response frequency. While providing a model for the motion of an AUV's control surfaces may have little effect on path predictions, implementing such a model would



(a) Current implementation **(b)** Improved implementation

Figure 5.1: Propeller actuator disk and rigid-body implementation schemes.

be particularly important when modeling a full-scale submarine. While the control surfaces of an AUV are generally capable of relatively high accelerations, this is not the case for larger vehicles with large, heavy control surfaces. Another factor that determines the importance of proper modeling of control surface motion is the size of the control surfaces relative to the vehicle as a whole.

5.1.1.6 Outer-Loop Depth Controller

As discussed in Section 4.3.4, the VFRM simulations and state-space models presented in this dissertation lack the outer-loop PI depth controller employed on the actual GPAUV. While, implementing this additional level of control in the VFRM macro and state-space models is relatively trivial, insuring that the vehicle is in a stable operating condition at the beginning of a maneuver requires more consideration. Depth-controlled maneuvering runs with the state-space models indicate that the GPAUV takes approximately 120 s to stabilize in the vertical plane (this result is in good agreement with field experiments). To avoid the need to precede every maneuver by a 120 s stabilization period, the GPAUV's stable condition (which includes the vehicle and controller states) can be found and used to initialize maneuvers.

5.1.2 Analysis of Turning Circle Maneuvers

All of the maneuvers considered for this study involved discrete changes in yaw reference heading. In a number of these maneuvers, the maximum yaw rate predicted by the various methods differed greatly. Turning circle maneuvers, in which the vehicle is commanded to deflect its rudders to some angle and make a steady turn, may provide additional insight into the sources of discrepancies between results. In future work, additional attention should be paid to: (1) determining the coefficients in state-space models that dominate this motion and (2) analyzing turning circle maneuver predictions amongst different methods and models.

5.1.3 VFRM Simulations to Inform System Identification (SI)

The approach employed in this study of using captive tests as the input for an SI process is somewhat atypical. A more common approach is to used data from a free-running maneuver to inform the SI process. If a physical model is used, an optimization can be applied to an objective function based on vehicle state to create a model. This concept is discussed in Section 3.5. A similar approach could be used with a VFRM simulation in place of the physical model. When using a VFRM simulation as the input, the optimization function could be based on vehicle state or the force and moment predictions. For the latter option, the prescribed motion comparisons presented in Section 4.3.7 could be used. The optimization process would be tasked with manipulating the model coefficients to make the state-space model force and moment predictions line up with those from the VFRM simulation. The success of this approach is dependent on a VFRM simulation being a close approximation of reality. As discussed in Section 4.3.9, this has yet to be shown conclusively.

One issue that must be considered when applying this mode of SI to obtain a state-space model is that the time length of a maneuver is effectively equivalent to a weighting factor within the optimization process. Given a 5 s maneuver in the vertical plane and 20 s maneuver in horizontal plane, the optimizer will likely settle on a model that better satisfies the objective function for the horizontal plane. This effect should be taken into consideration when designing a set of maneuvers with which to populate a state-space model.

5.1.4 Biomimetic Vehicles

Biomimetic vehicles, with external geometries, propulsion and control methods based on examples from the natural world, are currently an area of increasing interest [92–95]. For a number of reasons, these vehicles are not easily represented using state-space models. Population of a model for a biomimetic vehicle is made challenging by the lack of similar vehicles from which to draw guidance. Without past experience with a similar vehicle, formulation of a well-tailored state-space model becomes problematic. A related issue is the lack of empirical

databases with similar hull-forms from which model parameters can be interpolated. As mentioned in Section 3.3.6, these vehicles tend to have a large net change in the geometry during propulsion and control surface usage. CFD simulations can be used in these applications both as a means of populating state-space models and as an alternative to state-space models (i. e., VFRM simulations).

5.1.5 Robust Validation

A more robust validation of VFRM simulations could be realized via a comparison with experiments employing visual motion-tracking measurements. These methods offer far more accuracy than the navigational sensors on most autonomous vehicles. A number of experiments have been performed with small unmanned aircraft within specialized motion-tracking laboratories in which estimates for the forces experienced by the aircraft are extracted along with trajectory [125–127]. The tested aircraft are often designed for unsteady flight and high maneuverability.

5.2 Conclusion

One can see from Figure 4.11 that the VFRM simulations are computationally expensive. Maneuvers lasting only thirty seconds may take roughly 60 hours to solve on a medium-sized high-performance computer (HPC). Conversely, the same maneuver can be run with a state-space model in less than one minute on the average desktop. While this constraint will limit the use of VFRM simulations to only the most appropriate situations, if the past is any indication, computing hardware and software are likely to yield vast improvements. If hardware computing efficiency continues to progress according to Moore's law in years to come, a simulation requiring 24 hours of computation today will take only 1.5 hours 10 years from now.

The state-space models created for the GPAUV using an SI approach to process the results of CFD simulations compared well with a model created from a semi-empirical database. This result can support the use of similar processes to obtain models for future vehicles that may not be well characterized by existing semi-empirical databases. Additionally, comparison with experimental measurements and results from VFRM simulations has highlighted areas where additional attention may generate substantial improvements in model performance. A new state-space model formulation was created for a port-starboard symmetric AUV. Of the three state-space models considered in the study, this new formulation showed the best agreement with the VFRM simulation.

Conclusive evidence for the accuracy of VFRM simulations was not obtained from this study. The VFRM simulations were not able to match the experimental measurements with any higher degree of accuracy than the state-space models. While this may be due to the methodology on which these simulations are based, it is equally likely to be linked to major issues with the limited experimental trials available for comparison. As VFRM simulations are more directly based on the actual physical system that governs the dynamics of a vehicle, their ability to return realistic results should be only a matter of proper use of the component modeling tools. Additional experimental data is necessary to provide a more decisive understanding of this analysis approach.

References

[1] Ryan G. Coe and Wayne L. Neu. Vehicle control in a cfd environment. In *2011 Grand Challenges on Modeling and Simulation Conference (GCMS '11)*, The Hague, Netherlands, 2011.

[2] Brian McCarter, Robert Briggs, Stephen Portner, Dan Stilwell, Wayne Neu, Ryan Coe, Richard Duelley, Dexter Malley, and Jason Mims. Design and testing of a self-mooring AUV. In *OCEANS*, Virginia Beach, VA, 2012. MTS/IEEE.

[3] Morton Gertler and Grant. R. Hagen. Standard equations of motion for submarine simulation. Technical Report DTMB 2510, Naval Ship Research and Development Center, 1967.

[4] J. Feldman. Dtnsrdc revised standard submarine equations of motion. Technical Report DTNSRDC/SPD-0393-09, David W. Taylor Naval Ship Research and Development Center, 1979.

[5] A. Goodman. Experimental techniques and methods of analysis used in submerged body research. In *3rd Symposium on Naval Hydromechanics*. Office of Naval Research, 1960.

[6] Morton Gertler. The DTMB planar-motion-mechanism system. Technical Report 2523, 1967.

[7] R. E. D. Bishop and A. G. Parkinson. On the planar motion mechanism used in ship model testing. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 266(1171):35–61, 1970.

[8] R. E. D. Bishop, R. K. Burcher, and W. G. Price. Application of functional analysis to oscillatory ship model testing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 332(1588):37–49, 1973.

[9] R. E. D. Bishop, A. G. Parkinson, and W. G. Price. On the nature of slow motion derivatives. *Journal of Sound and Vibration*, 51(1):111–116, 1977.

[10] J.R. Morison, M.P. O'Brien, J.W. Johnson, and S.A. Schaaf. The force exerted by surface waves on piles. *Journal of Petroleum Technology*, 2(5):149–154, 1949.

[11] Martin A. Abkowitz. *Stability and motion control of ocean vehicles; organization, development, and initial notes of a course of instruction in the subject*. M.I.T. Press, Cambridge, 1969.

[12] J. N. Newman. *Marine hydrodynamics*. MIT Press, Cambridge, Mass., 1977.

[13] J. Strom-Tejsen. A digital computer technique for prediction of standard maneuvers of surface ships. Technical Report 2130, David Taylor Model Basin, 1965.

[14] Thor Fossen. *Nonlinear Modelling and Control of Underwater Vehicles*. PhD thesis, Norwegian Institute of Technology, 1991.

[15] Thor I. Fossen. *Guidance and control of ocean vehicles*. Wiley, Chichester; New York, 1994.

[16] Thor I. Fossen. *Marine control systems : guidance, navigation and control of ships, rigs and underwater vehicles*. Marine Cybernetics, Trondheim, Norway, 2002.

[17] Thor I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. Wiley, Chichester, West Sussex, U.K.; Hoboken N.J., 2011.

[18] H. McDonald and D. Whitfield. Self-propelled maneuvering underwater vehicles. In *21st Symposium on Naval Hydrodynamics*, pages 478–489, Trondheim, Norway, 1997. National Academy Press.

[19] W. C. Zierke, D. A. Boger, F. Davoudzadeh, J. J. Dreyer, H. McDonald, C. G. Schott, w. C. Zierke, A. Arabshahi, W. R. Briley, J. A. Busby, J. P. Chen, M. Y. Jiang, R. Jonnalagadda, J. McGinley, R. Panajakshan, C. Sheng, M. L. Stokes, L. K. Taylor, and D. L. Whitfield. A physics-based means of computing the flow around a maneuvering underwater vehicle. Technical Report TR 97-002, Office of Naval Research, 1997.

[20] L. K. Taylor, R. Pankajakshan, M. Jiang, C. Sheng, W. R. Briley, D. L. Whitfield, F. Davoudzadeh, D. A. Boger, H. J. Gibeling, J. Gorski, H. Haussling, R. Coleman, and G. Buley. Large-scale simulations for maneuvering submarines and propellers. In *29th Plasmadynamics and Lasers Conference*, Albuquerque, NM, 1998. AIAA.

[21] Mark C. Bettle, Andrew G. Gerber, and George D. Watt. Unsteady analysis of the six DOF motion of a buoyantly rising submarine. *Computers & Fluids*, 38(9):1833–1849, 2009.

[22] James J. Dreyer and David A. Boger. Validation of a free-swimming, guided multi-body urans simulation tool. In *28th Symposium on Naval Hydrodynamics*, Pasadena, California, 2010. National Academy Press.

[23] N. Alin, C. Fureby, S.U. Svennberg, W.C. Sandberg, R. Ramamurti, and R.E. Bensow. Large eddy simulation of the transient flow around a submarine during maneuver. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2007. AIAA.

[24] John E. Poremba. *Hydrodynamics and Maneuvering Simulations of a Non-Body-of-Revolution Submarine*. PhD thesis, The Pennsylvania State University, 2009.

[25] R. Pankajakshan, L. K. Taylor, M. Jiang, M. G. Remotigue, W. R. Briley, and D. L. Whitfield. Parallel simulations for control-surface induced submarine maneuvers. In *38th Aerospace Sciences Meeting Conference and Exhibit*, Reno, Nevada, 2000. AIAA.

[26] R. Pankajakshan, L. K. Taylor, C. Sheng, M. Jiang, W. R. Briley, and D. L. Whitfield. Parallel efficiency in implicit multiblock, multigrid simulations, with application to submarine maneuvering. In *39th Aerospace Sciences Meeting Conference and Exhibit*, Reno, NV, 2001. AIAA.

[27] R. Pankajakshan, M. G. Remotigue, L. K. Talyor, M. Jiang, W. R. Briley, and D. L. Whitfield. Validation of control-surface induced submarine maneuvering simulations. In *24th Symposium on Naval Hydrodynamics*, pages 624–639, Fukuoka, Japan, 2002. The National Academies Press.

[28] Pablo M. Carrica, Kwang-Jun Paik, Hamid S. Hosseini, and Frederick Stern. Urans analysis of a broaching event in irregular quartering seas. *Journal of Marine Science and Technology*, 13(4):395–407, 2008.

[29] Shanti Bhushan, Tao Xing, Pablo Carrica, and Frederick Stern. Model- and full-scale urans simulations of athena resistance, powering, seakeeping, and 5415 maneuvering. *Journal of Ship Research*, 53(4):179–198, 2009.

[30] Pablo M. Carrica, Hamid Sadat-Hosseini, and Fred Stern. CFD analysis of broaching for a model surface combatant with explicit simulation of moving rudders and rotating propellers. *Computers & Fluids*, 2011.

[31] Hamid Sadat-Hosseini, Pablo Carrica, Frederick Stern, Naoya Umeda, Hirotada Hashimoto, Shinya Yamamura, and Akihiko Mastuda. Cfd, system-based and efd study of ship dynamic instability events: Surf-riding, periodic motion, and broaching. *Ocean Engineering*, 38(1):88–110, 2011.

[32] Pablo M. Carrica, Farzad Ismail, Mark Hyman, Shanti Bhushan, and Frederick Stern. Turn and zigzag maneuvers of a surface combatant using a URANS approach with dynamic overset grids. *Journal of Marine Science and Technology*, 18(2):166–181, 2012.

[33] F. Stern, K. Agdrup, S.Y. Kim, A.C. Hochbaum, K.P. Rhee, F. Quadvlieg, P. Perdon, T. Hino, R. Broglia, and J. Gorski. Experience from simman 2008the first workshop on verification and validation of ship maneuvering simulation methods. *Journal of Ship Research*, 55(2):135–147, 2011.

[34] CD-adapco. STAR-CCM+ v8 help manual, 2013.

[35] Joel H. Ferziger and Milovan Peric. *Computational methods for fluid dynamics*. Springer, Berlin, 2002.

[36] David C Wilcox. *Turbulence modeling for CFD*, volume 3. DCW industries La Caada, CA, 2006.

[37] S. V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, 1972.

[38] J. D. van Manen and P. van Oossanen. *Principles of Naval Architecture*, volume 2, chapter Propulsion, pages 131–135. The Society of Naval Architects and Marine Engineers, Jersey City, NJ, 1988.

[39] Wayne L. Neu. AOE 3264 resistance and propulsion of ships course notes. 2009.

[40] L. Larsson, B. Regnstrm, L. Broberg, D.-Q.Li, and C.-E.Janson. Failures, fantasies, and feats in the theoretical/numerical prediction of ship performance. In *Twenty-Second Symposium on Naval Hydrodynamics*, Washington, D.C., 1998. Office of Naval Research.

[41] Jens Nrkr Srensen and Wen Zhong Shen. Computation of wind turbine wakes using combined navier-stokes/actuator-line methodology. In *Proceedings of European Wind Energy Conference EWEC '99*, pages 156–159, Nice, France, 1999.

[42] Robert Flemming Mikkelsen, Jens Nrkr Srensen, and Niels Troldborg. Prescribed wind shear modelling with the actuator line technique. In *EWEC*, Milan, Italy, 2007.

[43] Niels Troldborg, Jens N. Srensen, and Robert Mikkelsen. Actuator line simulation of wake of wind turbine operating in turbulent inflow. *Journal of Physics: Conference Series*, 75:012063, 2007.

[44] Stefan S. A. Ivanell. Numerical computations of wind turbine wakes. Technical report, Royal Institute of Technology, 2009.

[45] Richard S. Duelley. *Autonomous Underwater Vehicle Propulsion Design*. PhD thesis, Virginia Polytechnic Institute and State University, 2010.

[46] H. Hadzic. *Development and Application of Finite Volume Method for the Computation of Flows Around Moving Bodies on Unstructured, Overlapping Grids*. PhD thesis, Technical University Hamburg-Harburg, 2005.

[47] Ryan G. Coe and Wayne L. Neu. Use of overset mesh to allow dynamic deflection of tight-fitting control surfaces in CFD simulations. In *32nd International Conference on Ocean, Offshore and Arctic Engineering, OMAE2013*, Nantes, France, 2013. ASME.

[48] C. L. Nickell, C. A. Woolsey, and D. J. Stilwell. A low-speed control module for a streamlined AUV. In *OCEANS, 2005. Proceedings of MTS/IEEE*, pages 1680–1685 Vol. 2, 2005.

[49] Thomas T. Huang, Han-Lieh Liu, and Nancy C. Groves. Experiments of the DARPA SUBOFF program. Technical Report 1298-02, David Taylor Research Center, 1989.

[50] Nancy C. Groves, Thomas T. Huang, and Ming S. Chang. Geometric characteristics of DARPA SUBOFF models (DTRC model numbers 5470 and 5471). Technical Report 1298-01, David Taylor Research Center, 1989.

[51] Han-Lieh Liu and Thomas T. Huang. Summary of DARPA SUBOFF experimental program data. Technical Report 1298-11, Naval Surface Warfare Center, 1998.

[52] L. Bruce Crook. Resistance for DARPA SUBOFF as represented by Model 5470. Technical Report DTRC/SHD-1298-07, David Taylor Research Center, 1990.

[53] Patrick J. Roache. *Verification and validation in computational science and engineering*. Hermosa, Albuquerque, NM, 1998.

[54] C. J. Steffen, D. Reddy, and K. Zaman. Analysis of flowfield from a rectangular nozzle with delta tabs. In *26th AIAA Fluid Dynamics Conference*, Sand Diego, CA, 1995. AIAA.

[55] Taeyoung Han and V. C. Patel. Flow separation on a spheroid at incidence. *J. Fluid Mech.*, 92(4):643–657, 1979.

[56] Seungki Ahn. *An experimental study of flow over a 6 to 1 prolate spheroid at incidence*. PhD thesis, Virginia Polytechnic Institute and State University, 1992.

[57] C. J. Chesnakas and R. L. Simpson. Full three-dimensional measurements a 6:1 prolate spheroid. *Experiments in Fluids*, 17:68–74, 1994.

[58] Ngoc T. Hoang, Todd G. Wetzel, and Roger L. Simpson. Surface pressure measurements over a 6:1 prolate spheroid undergoing time-dependent maneuvers. In *12th AIAA Applied Aerodynamics Conference*, pages 751–763, Colorado Springs, CO, 1994.

[59] Hoang Ngoc, Wetzel Todd, and Simpson Roger. Unsteady measurements over a 6:1 prolate spheroid undergoing a pitch-up maneuver. In *32nd Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 1994.

[60] Todd G. Wetzel. *Unsteady Flow Over a 6:1 Prolate Spheroid*. PhD thesis, Virginia Polytechnic Institute and State University, 1996.

[61] Christopher J. Chesnakas and Roger L. Simpson. Detailed investigation of the three-dimensional separation about a 6:1 prolate spheroid. *AIAA Journal*, 35(6):990–999, 1997.

[62] Todd G. Wetzel and Roger L. Simpson. Unsteady flow over a 6:1 prolate spheroid. Technical Report VPI-AOE-232, Advanced Research Projects Agency, 1997.

[63] Todd G. Wetzel and Roger L. Simpson. Unsteady crossflow separation location measurements on a maneuvering 6:1 prolate spheroid. *AIAA Journal*, 36(11):2063–2071, 1998.

[64] Todd G. Wetzel, Roger L. Simpson, and Christopher J. Chesnakas. Measurement of three-dimensional crossflow separation. *AIAA Journal*, 36(4):557–564, 1998.

[65] Michael C. Goody, Roger L. Simpson, and Christopher J. Chesnakas. Separated flow surface pressure fluctuations and pressure-velocity correlations on prolate spheroid. *AIAA Journal*, 38(2):266–274, 2000.

[66] K.C. Wang. Three-dimensional boundary layer near the plane of symmetry of a spheroid at incidence. *J. Fluid Mech.*, 43(1):187–209, 1970.

[67] Veer N. Vatsa, James L. Thomas, and Bruce W. Wedan. Navier-stokes computations of a prolate spheroid at angle of attack. *Journal of Aircraft*, 26(11):986–993, 1989.

[68] K. E. N. Gee, Russell M. Cummings, and Lewis B. Schiff. Turbulence model effects on separated flow about a prolate spheroid. *AIAA Journal*, 30(3):655–664, 1992.

[69] L.K. Taylor, A. Arabshahi, and D.L. Whitfield. Unsteady three-dimensional incompressible navier-stokes computations for a prolate spheroid undergoing time-dependent maneuvers. In *33rd Aerospace Sciences Meeting and Exhibit*, Reno, NV, 1995. AIAA.

[70] George S. Constantinescu, Hugo Pasinato, You-Qin Wang, James R. Forsythe, and Kyle D. Squires. Numerical investigation of flow past a prolate spheroid. *Journal of Fluids Engineering*, 124(4):904, 2002.

[71] Sung-Eun Kim, Shin H. Rhee, and Davor Cokljat. The prolate spheroid separates turbulence models. Technical report, Fluent, 2005.

[72] Rupesh B. Kotapati-Apparao and Kyle D. Squires. Prediction of a prolate spheroid undergoing a pitchup maneuver. In *41st Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 2003. AIAA.

[73] Zhixiang Xiao, Yufei Zhang, Jingbo Huang, Haixin Chen, and Song Fu. Prediction of separation flows around a 6:1 prolate spheroid using rans/les hybrid approaches. *Acta Mechanica Sinica*, 23(4):369–382, 2007.

[74] N. Alin, M. Berglund, and C. Fureby. A VLES approach applied to flows around complex underwater vehicle hulls. In *38th Aerospace Sciences Meeting & Exhibit*, Reno, NV, 2000. AIAA.

[75] P.O. Hedin, M. Berglund, N. Alin, and C. Fureby. Large eddy simulation of the flow around an inclined prolate spheroid. In *39th AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, 2001. AIAA.

[76] A. Karlsson and C. Fureby. LES of the flow past a 6:1 prolate spheroid. In *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, Orlando, FL, 2009. AIAA.

[77] Nathan W. Scott and Earl P.N. Duque. Unsteady Reynolds-averaged Navier-Stokes predictions of the flow around a prolate spheroid. In *42nd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2004. AIAA.

[78] S.-E. Kim, S. H. Rhee, and Davor Cokljat. Application of modern turbulence models to vortical flow around a 6:1 prolate spheroid at incidence. In *41st Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2003. AIAA.

[79] Todd G. Wetzel and Roger L. Simpson. Unsteady three-dimensional crossflow separation measurements on a prolate spheroid undergoing time-dependent maneuvers. In *35th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 1997. AIAA.

[80] Shin Hyung Rhee and Takanori Hino. Numerical simulation of unsteady turbulent flow around maneuvering prolate spheroid. *AIAA Journal*, 40(10), 2002.

[81] Thor I. Fossen, Karl-Petter Lindegaard, and Roger Skjetne. Inertia shaping techniques for marine vessels using acceleration feedback. In *15th Triennial World Congress*, Barcelona, Spain, 2002. IFAC.

[82] Horace Lamb. *Hydrodynamics*. Cambridge University Press, Cambridge; New York, 1932.

[83] Krishnamurty Karamcheti. *Principles of ideal-fluid aerodynamics*. Wiley, New York, 1966.

[84] Kurt Wendel. Hydrodynamic masses and hydrodynamic moments of inertia. Technical Report Translation 260, David W. Taylor Model Basin, 1956.

[85] Nomenclatlure for treating the motion of a submerged body through a fluid. Technical Report 1-5, The Society of Naval Architects and Marine Engineers, 1950.

[86] L. M. Milne-Thomson. *Theoretical hydrodynamics*. Macmillan, New York, 1968.

[87] T. Miloh and L. Landweber. Generalization of the kelvinkirchhoff equations for the motion of a body through a fluid. *Physics of Fluids*, 24(1):6, 1981.

[88] Frederick H. Imlay. The complete expressions for "added mass" of a rigid body moving in an ideal fluid. Technical report, David Taylor Model Basin, 1961.

[89] D. E. Humphreys and K. W. Watkinson. Prediction of acceleration hydrodynamic coefficients for underwater vehicles from geometric parameters. Technical Report NCSL-TR-327-78, Naval Coastal Systems Laboratory, 1978.

[90] C. L. Crane, H. Eda, and A. Landsburg. *Controlllability*, volume 3, chapter IX, pages 206–207. The Society of Naval Architects and Marine Engineers, Jersy City, NJ, 1989.

[91] M. Mackay. A review of submarine out-of-plane normal force and pitching moment. Technical Report TM 2004-135, DRDC Atlantic, 2004.

[92] D. Barrett, M. Grosenbaugh, and M. Triantafyllou. The optimal control of a flexible hull robotic undersea vehicle propelled by an oscillating foil. In *Autonomous Underwater Vehicle Technology, 1996. AUV '96., Proceedings of the 1996 Symposium on*, pages 1–9, 1996.

[93] Afzal Suleman and Curran Crawford. Design and testing of a biomimetic tuna using shape memory alloy induced propulsion. *Computers & Structures*, 86(35):491–499, 2008.

[94] Jenhwa Guo. Maneuvering and control of a biomimetic autonomous underwater vehicle. *Autonomous Robots*, 26(4):241–249, 2009.

[95] Wang Zhenlong, Wang Yangwei, Li Jian, and Hang Guanrong. A micro biomimetic manta ray robot fish actuated by SMA. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pages 1809–1813, 2009.

[96] Timothy Prestero. *Verification of a Six-Degree of Freedom Simulation Model for the REMUS Autonomous Underwater Vehicle*. PhD thesis, Massachusetts Institute of Technology, 2001.

[97] M. Mackay. DSSP50 build 090910 documentation, part 1: Guide and tutorial. Technical Report TM 2009-226, DRDC Atlantic, 2009.

[98] M. Mackay. DSSP50 build 090910 documentation, part 2: Input reference. Technical Report TM 2009-237, DRDC Atlantic, 2009.

[99] M. Mackay. DSSP50 build 090910 documentation, part 3: Algorithm description. Technical Report TM 2009-228, DRDC Atlantic, 2009.

[100] Jerome P. Feldman. Method of performing captive-model experiments to predict the stability and control characteristics of submarines. Technical Report CRDKNSWC-HD-0393-25, Naval Surface Warfare Center, Carderock Div, 1995.

[101] Serge L. Toxopeus. Deriving mathematical manoeuvring models for bare ship hulls using viscous flow calculations. *Journal of Marine Science and Technology*, 14(1):30–38, 2009.

[102] S.L. Toxopeus. *Practical application of viscous-flow calculations for the simulation of manoeuvring ships*. PhD thesis, Delft University of Technology, 2011.

[103] R. Broglia, R. Muscari, and A. Di Mascio. Numerical analysis of blockage effects in PMM tests. In *26th Symposium on Naval Hydrodynamics*, Rome, Italy, 2006.

[104] A. C. Hochbaum. Virtual PMM tests for manoeuvring prediction. In *26th Symposium on Naval Hydrodynamics*, Rome, Italy, 2006.

[105] A. Phillips, M. Furlong, and S.R. Turnock. Virtual planar motion mechanism tests of the autonomous underwater vehicle Autosub. In *STG-Conference / Lectureday "CFD in Ship Design"*, Hamburg, Germany, 2007.

[106] Seong-Keon Lee, Tae-Hwan Joung, Se-Jong Cheon, Taek-Soo Jang, and Jeong-Hee Lee. Evaluation of the added mass for a spheroid-type unmanned underwater vehicle by vertical planar motion mechanism test. *International Journal of Naval Architecture and Ocean Engineering*, 3(3):174–180, 2011.

[107] M. Vantorre. On the influence of the accuracy of planar motion mechanisms on results of captive manoeuvring tests. In *Scientific and Methodological Seminar on Ship Hydrodynamics*, pages 28.1 – 28.8, Varna, 1988.

[108] Ryan G. Coe and Wayne L. Neu. Amplitude effects on virtual PMM tests. In *OCEANS*, Hampton Roads, VA, 2012. IEEE.

[109] Ryan Coe and Wayne Neu. Virtual planar motion mechanism tests in a cfd environment. In *Virginia Space Grant Consortium Student Research Conference*, Williamsburg, VA, 2012.

[110] III Morrison, A. T. and D. R. Yoerger. Determination of the hydrodynamic parameters of an underwater vehicle during small scale, nonuniform, 1-dimensional translation. In *OCEANS*, pages II277–II282 vol.2. MTS/IEEE, 1993.

[111] A. Alessandri, M. Caccia, G. Indiveri, and G. Veruggio. Application of ls and ekf techniques to the identification of underwater vehicles. In *Control Applications*, 1998.

[112] Massimo Caccia, Giovanni Indiveri, and Gianmarco Veruggio. Modeling and identification of open-frame variable configuration unmanned underwater vehicles. *IEEE JOURNAL OF OCEANIC ENGINEERING*, 25(2):227–240, 2000.

[113] M. E. Rentschler, F. S. Hover, and C. Chryssostomidis. System identification of open-loop maneuvers leads to improved AUV flight performance. *IEEE Journal of Oceanic Engineering*, 31(1):200–208, 2006.

[114] T. Binazadeh, M. J. Yazdanpanah, and M. H. Shafiei. Identification of a variable mass underwater vehicle via volterra neural network. *Journal of Dynamic Systems, Measurement, and Control*, 132(2):024501–024501, 2010. 10.1115/1.4000814.

[115] Stephen Portner. Three-dimensional panel method code, 2012.

[116] Ryan G. Coe and Wayne L. Neu. Asymmetrical wake and propulsor effects on control surface effectiveness on auvs. In *OCEANS*, Hampton Roads, VA, 2012. MTS/IEEE.

[117] Robert Ives Hickey. *Submarine Motion Simulation Including Zero Forward Speed and Propeller Race Effects*. PhD thesis, Massachusetts Institute of Technology, 1990.

[118] Brian R. McCarter. State-space models for the gpauv, 2013.

[119] Ryan G. Coe, Brian R. McCarter, and Wayne L. Neu. CFD-based maneuvering simulations for autonomous underwater vehicles. In *Virginia Space Grant Consortium Student Research Conference*, Norfolk, VA, 2013.

[120] Jan Petrich. *Improved Guidance, Navigation, and Control for Autonomous Underwater Vehicles: Theory and Experiment*. PhD thesis, Virginia Polytechnic Institute and State University, 2009.

[121] Jan Petrich and Daniel J. Stilwell. Robust control for an autonomous underwater vehicle that suppresses pitch and yaw coupling. *Ocean Engineering*, 38(1):197–204, 2011.

[122] J.H.A.M. Vervoort. *Modeling and Control of an Unmanned Underwater Vehicle*. PhD thesis, University of Canterbury - University of Technology Eindhoven, 2009.

[123] L. V. Steenson, A. B. Phillips, S. R. Turnock, M. E. Furlong, and E. Rogers. Effect of measurement noise on the performance of a depth and pitch controller using the model predictive control method. In *Autonomous Underwater Vehicles (AUV), 2012 IEEE/OES*, pages 1–8, 2012.

[124] George D. Watt. Modelling and simulating unsteady six degrees-of-freedom submarine rising maneuvers. Technical Report TR 2007-008, DRDC, 2007.

[125] B. F. Mettler. Extracting micro air vehicles aerodynamic forces and coefficients in free flight using visual motion tracking techniques. *Experiments in Fluids*, 49(3):557–569, 2010.

[126] Daniel V. Uhlig, Agrim Sareen, Pritam Sukumar, Arjun H. Rao, and Michael S. Selig. Determining aerodynamic characteristics of a micro air vehicle using motion tracking. In *AIAA GNC/AFM/MST/ASC/ASE 2010 Conference*, Toronto, Ontario, Canada, 2010. AIAA.

[127] Daniel V. Uhlig and Michael S. Selig. Determining aerodynamic characteristics of a micro air vehicle using motion tracking. *Journal of Aircraft*, pages 1–10, 2013.

Appendix A

State-Space Models

A.1 State-Space Model Formulations

A.1.1 Rigid-body Kinematic Formulation

The rigid kinematic formulations employed in the dynamic CFD simulations and state-space models are a form of Netwon's second law.

$$\vec{\tau} = \mathbf{M}_{RB}\dot{n\vec{u}} + \mathbf{C}_{RB}\left(n\vec{u}\right)n\vec{u} \tag{A.1}$$

The terms in (A.1) are defined in Section 3.4.2. If the off-diagnol rigid-body inertial terms (I_{xy}, I_{xz},...) are assumed to be zero and the body is assumed to be port-starboard symmetric ($y_G = 0$), the kinematic formulation can be written out in component form as follows.

$$\begin{aligned}
X_{\text{external}} &= m\left[\dot{u} + qw - rv + x_G\left(q^2 + r^2\right) + z_G\left(pr + \dot{q}\right)\right] \\
Y_{\text{external}} &= m\left[\dot{v} - wp + ur + x_G\left(qp + \dot{r}\right) + z_G\left(qr - \dot{p}\right)\right] \\
Z_{\text{external}} &= m\left[\dot{w} - uq + vp + x_G\left(rp - \dot{q}\right) - z_G\left(p^2 + q^2\right)\right] \\
K_{\text{external}} &= I_{xx}\dot{p} + \left(I_{zz} - I_{yy}\right)rq - mz_G\left(\dot{v} - wp + ur\right) \\
M_{\text{external}} &= I_{qq}\dot{q} + \left(I_{xx} - I_{zz}\right)rp + m\left[z_G\left(\dot{u} - vr + wq\right) - x_G\left(\dot{w} - uq + vp\right)\right] \\
N_{\text{external}} &= I_{zz}\dot{r} + \left(I_{yy} - I_{xx}\right)pq - mx_G\left(\dot{v} - wp + ur\right)
\end{aligned} \tag{A.2}$$

A.1.2 Gertler-Hagen Submarine Equations of Motion

Gertler and Hagen's "Standard Equations of Motion for Submarine Simulation" are reprinted here for convenience. Definitions of the terms and intermediate variables can be found in [3].

Surge

$$
\begin{aligned}
& m\left[\dot{u} - vr + wq - x_g\left(q^2 + r^2\right) + y_g\left(pq - \dot{r}\right) + z_g\left(pr - \dot{q}\right)\right] = \\
& \quad + \frac{\rho}{2}\ell^4\left[{X_{qq}}'q^2 + {X_{rr}}'r^2 + {X_{rp}}'rp\right] \\
& \quad + \frac{\rho}{2}\ell^3\left[{X_{\dot{u}}}'\dot{u} + {X_{vr}}'vr + {X_{wq}}'wq\right] \\
& \quad + \frac{\rho}{2}\ell^2\left[{X_{uu}}'u^2 + {X_{vv}}'v^2 + {X_{ww}}'w^2\right] \\
& \quad + \frac{\rho}{2}\ell^2u^2\left[{X_{\delta_r\delta_r}}'{\delta_r}^2 + {X_{\delta_s\delta_s}}'{\delta_s}^2 + {X_{\delta_b\delta_b}}'{\delta_b}^2\right] \\
& \quad + \rho\ell^2\left[a_iu^2 + b_iuu_c + c_iu_c^2\right] \\
& \quad - (W - B)\sin(\theta) \\
& \quad + \frac{\rho}{2}\ell^2\left[X_{vv\eta}v^2 + X_{ww\eta}w^2 + X_{\delta_r\delta_r\eta}\delta_r^2u^2 + X_{\delta_s\delta_s\eta}\delta_s^2u^2\right](\eta - 1)
\end{aligned}
\tag{A.3}
$$

Sway

$$
\begin{aligned}
& m\left[\dot{v} - wq + ur - y_g\left(r^2 + p^2\right) + z_g\left(pq - \dot{r}\right) + x_g\left(qp + \dot{r}\right)\right] = \\
& \quad + \frac{\rho}{2}\ell^4\left[Y'_{\dot{r}}\dot{r} + Y'_{\dot{p}}\dot{p} + Y_{p|p|}p\left|p\right| + Y'_{pq}pq + Y'_{qr}qr\right] \\
& \quad + \frac{\rho}{2}\ell^3\left[Y'_{\dot{v}}\dot{v} + Y'_{vq}vq + Y'_{wp}wp + Y'_{wr}wr\right] \\
& \quad + \frac{\rho}{2}\ell^2\left[Y'_ur + Y'_pup + Y'_{|r|\delta_r}u\left|r\right|\delta_r + Y'_{v|r|}\frac{v}{|v|}\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right|\left|r\right|\right] \\
& \quad + \frac{\rho}{2}\ell^2\left[Y'_*u^2 + Y'_vuv + Y'_{v|v|}v\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right|\right] \\
& \quad + \frac{\rho}{2}\ell^2\left[Y'_{vw}vw + Y'_{\delta_r}u^2\delta_r\right] \\
& \quad + (W - B)\cos(\theta)\sin(\phi) \\
& \quad + \frac{\rho}{2}\ell^2Y'_{r\eta}ur\,(\eta - 1) \\
& \quad + \frac{\rho}{2}\ell^2\left[Y'_{v\eta}uv + Y_{v|v|\eta}v\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right| + Y'_{\delta_r\eta}\delta_ru^2\right](\eta - 1)
\end{aligned}
\tag{A.4}
$$

Heave

$$
\begin{aligned}
& m\left[\dot{w}-uq+vp-x_G\left(p^2+q^2\right)+x_G\left(rp-\dot{q}\right)+y_G\left(rq+\dot{p}\right)\right]= \\
& \quad+\frac{\rho}{2}\ell^4\left[Z'_{\dot{q}}\dot{q}+Z'_{pp}p^2+Z'_{rr}r^2+Z'_{rp}rp\right] \\
& \quad+\frac{\rho}{2}\ell^3\left[Z'_{\dot{w}}\dot{w}+Z'_{vr}vr+Z'_{vp}vp\right] \\
& \quad+\frac{\rho}{2}\ell^2\left[Z'_{q}uq+Z'_{|q|\delta_s}u\left|q\right|\delta_s+Z'_{w|q|}\frac{w}{|w|}\left|\left(v^2+w^2\right)^{\frac{1}{2}}\right|\left|q\right|\right] \\
& \quad+\frac{\rho}{2}\ell^2\left[Z'_{*}u^2+Z'_{w}uw+Z'_{w|w|}w\left|\left(v^2+w^2\right)^{\frac{1}{2}}\right|\right] \\
& \quad+\frac{\rho}{2}\ell^2\left[Z'_{|w|}u\left|w\right|+Z'_{ww}\left|\left(v^2+w^2\right)^{\frac{1}{2}}\right|\right] \\
& \quad+\frac{\rho}{2}\ell^2\left[Z'_{vv}v^2+Z'_{\delta_s}u^2\delta_s+Z'_{\delta_b}+u^2\delta_b\right] \\
& \quad+\left(W-B\right)\cos\left(\theta\right)\cos\left(\phi\right) \\
& \quad+\frac{\rho}{2}\ell^3Z'_{q\eta}uq\left(\eta-1\right) \\
& \quad+\frac{\rho}{2}\ell^3\left[Z'_{w\eta}uw+Z'_{w|w|\eta}w\left|\left(v^2+w^2\right)^{\frac{1}{2}}\right|+Z'_{\delta_s\eta}\delta_su^2\right]\left(\eta-1\right)
\end{aligned}
\tag{A.5}
$$

Roll

$$
\begin{aligned}
& I_x\dot{p}+\left(I_x-I_y\right)qr-\left(\dot{r}+pq\right)I_{xx}+\left(r^2-q^2\right)I_{yz}+\left(pr-\dot{q}\right)I_{xy} \\
& \quad+m\left[y_g\left(\dot{w}-uq+vp\right)-x_g\left(\dot{v}-wp+ur\right)\right]= \\
& \quad+\frac{\rho}{2}\ell^5\left[K'_{\dot{p}}\dot{p}+K'_{\dot{r}}\dot{r}+K'_{qr}qr+K'_{pq}pq+K'_{p|p|}p\left|p\right|\right] \\
& \quad+\frac{\rho}{2}\ell^4\left[K'_{p}up+K'_{r}ur+K'_{\dot{v}}\dot{v}\right] \\
& \quad+\frac{\rho}{2}\ell^4\left[K'_{vq}vq+K'_{wp}wp+K'_{wr}wr\right] \\
& \quad+\frac{\rho}{2}\ell^3\left[K'_{*}u^2+K'_{v}uv+Z'_{v|v|}v\left|\left(v^2+w^2\right)^{\frac{1}{2}}\right|\right] \\
& \quad+\frac{\rho}{2}\ell^3\left[K'_{vw}vw+K'_{\delta_r}u^2\delta_r\right] \\
& \quad+\left(y_gW-y_BB\right)\cos\left(\theta\right)\cos\left(\phi\right)-\left(x_gW-x_BB\right)\cos\left(\theta\right)\sin\left(\phi\right) \\
& \quad+\frac{\rho}{2}\ell^3K'_{*\eta}u^2\left(\eta-1\right)
\end{aligned}
\tag{A.6}
$$

Pitch

$$
\begin{aligned}
&I_y\dot{q} + (I_x - I_z)\, rp - (\dot{q} + qr)\, I_{xy} + \left(p^2 - r^2\right) I_{zx} + (qp - \dot{r})\, I_{yz} \\
&\quad + m\left[z_g\left(\dot{u} - vr + wq\right) - x_g\left(\dot{w} - uq + vp\right)\right] = \\
&\quad + \frac{\rho}{2}\ell^5\left[M'_{\dot{q}}\dot{q} + M'_{pp}p^2 + M'_{rr}r^2 + M'_{rp}rp + M'_{q|q|}q\,|q|\right] \\
&\quad + \frac{\rho}{2}\ell^4\left[M'_{\dot{w}}\dot{w} + M'_{vr}vr + M'_{vp}vp\right] \\
&\quad + \frac{\rho}{2}\ell^4\left[M'_{q}uq + M'_{|q|\delta_s}u\,|q|\,\delta_s + M'_{|w|q}\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right| q\right] \\
&\quad + \frac{\rho}{2}\ell^3\left[M'_{*}u^2 + M'_{w}uw + M'_{w|w|}w\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right|\right] \\
&\quad + \frac{\rho}{2}\ell^3\left[M'_{|w|}u\,|w| + M'_{ww}\left|w\left(v^2 + w^2\right)^{\frac{1}{2}}\right|\right] \\
&\quad - \left(x_G W - x_B B\right)\cos(\theta)\cos(\phi) - \left(z_G W - z_B B\right)\sin(\theta) \\
&\quad + \frac{\rho}{2}\ell^4 M'_{q\eta}uq\,(\eta - 1) \\
&\quad + \frac{\rho}{2}\ell^3\left[M'_{w\eta}uw + M'_{w|w|\eta}w\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right| + M'_{\delta_s\eta}\delta_s u^2\right](\eta - 1)
\end{aligned}
\tag{A.7}
$$

Yaw

$$
\begin{aligned}
&I_z\dot{r} + (I_y - I_x)\, pq - (\dot{q} + rp)\, I_{yz} + \left(q^2 - p^2\right) I_{xy} + (rq - \dot{p})\, I_{zx} \\
&\quad + m\left[x_G\left(\dot{v} - wp + ur\right) - y_G\left(\dot{u} - vr + wq\right)\right] = \\
&\quad + \frac{\rho}{2}\ell^5\left[N'_{\dot{r}}\dot{r} + N'_{\dot{p}}\dot{p} + N'_{pq}pq + N'_{qr}qr + N'_{r|r|}r\,|r|\right] \\
&\quad + \frac{\rho}{2}\ell^4\left[N'_{\dot{v}}\dot{v} + N'_{wr}wr + N'_{wp}wp + N'_{vq}vq\right] \\
&\quad + \frac{\rho}{2}\ell^4\left[N'_{p}up + N'_{r}ur + N'_{|r|\delta_r}u\,|r|\,\delta_r + N'_{|v|r}\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right| r\right] \\
&\quad + \frac{\rho}{2}\ell^3\left[N'_{*}u^2 + N'_{v}uv + N'_{v|v|}v\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right|\right] \\
&\quad + \frac{\rho}{2}\ell^3\left[N'_{vw} + N_{\delta_r}u^2\delta_r\right] \\
&\quad - \left(x_G W - x_B B\right)\cos(\theta)\sin(\phi) - \left(y_g W - y_B B\right)\sin(\theta) \\
&\quad + \frac{\rho}{2}\ell^4 N'_{r\eta}ur\,(\eta - 1) \\
&\quad + \frac{\rho}{2}\ell^3\left[N'_{v\eta}uv + N'_{v|v|\eta}v\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right| + N'_{\delta_r\eta}\delta_r u^2\right](\eta - 1)
\end{aligned}
\tag{A.8}
$$

A.1.3 Feldman's Revised Submarine Equations of Motion

Feldman's "Revised Standard Submarine Equations of Motions" are reprinted here for convenience. Definitions of the terms and intermediate variables can be found in [4].

Surge

$$\begin{aligned}
&m\left[\dot{u}-vr+wq-x_g\left(q^2+r^2\right)+y_g\left(pq-\dot{r}\right)+z_g\left(pr-\dot{q}\right)\right]= \\
&\quad+\frac{\rho}{2}\ell^4\left[{X_{qq}}'q^2+{X_{rr}}'r^2+{X_{rp}}'rp\right] \\
&\quad+\frac{\rho}{2}\ell^3\left[{X_{\dot{u}}}'\dot{u}+{X_{vr}}'vr+{X_{wq}}'wq\right] \\
&\quad+\frac{\rho}{2}\ell^2\left[{X_{vv}}'v^2+{X_{ww}}'w^2\right] \\
&\quad+\frac{\rho}{2}\ell^2\left[{X_{\delta_r\delta_r}}'u^2{\delta_r}^2+{X_{\delta_s\delta_s}}'u^2{\delta_s}^2+{X_{\delta_b\delta_b}}'u^2{\delta_b}^2\right] \\
&\quad-(W-B)\sin(\theta)+F_{xp}
\end{aligned} \tag{A.9}$$

Sway

$$\begin{aligned}
&m\left[\dot{v}-wq+ur-y_g\left(r^2+p^2\right)+z_g\left(pq-\dot{r}\right)+x_g\left(qp+\dot{r}\right)\right]= \\
&\quad+\frac{\rho}{2}\ell^4\left[{Y_{\dot{r}}}'\dot{r}+{Y_{\dot{p}}}'\dot{p}+{Y_{p|p|}}p\left|p\right|+{Y_{pq}}'pq\right] \\
&\quad+\frac{\rho}{2}\ell^3\left[{Y_r}'ur+{Y_p}'up+{Y_{\dot{v}}}'\dot{v}+{Y_{wp}}'wp\right] \\
&\quad+\frac{\rho}{2}\ell^2\left[Y_*'u^2+Y_v'uv+Y_{v|v|R}'v\left|\left(v^2+w^2\right)^{\frac{1}{2}}\right|\right] \\
&\quad+\frac{\rho}{2}\ell^2\left[{Y_{\delta_r\delta_r}}'u^2\delta_r+{Y_{\delta_r\delta_\eta}}'u^2\delta_r\left(\eta-\frac{1}{C}\right)C\right] \\
&\quad+\frac{\rho}{2}C_d\int_\ell h(x)\,v(x)\left\{w(x)^2+w(x)^2\right\}^{\frac{1}{2}}dx \\
&\quad-\frac{\rho}{2}\ell\bar{C}_l\int_{x_2}^{x_1}w(x)\,\bar{v}_{FW}\left(t-\tau[x]\right)dx \\
&\quad+(W-B)\cos(\theta)\sin(\theta)
\end{aligned} \tag{A.10}$$

Heave

$$
\begin{aligned}
& m\left[\dot{w}-uq+vp-x_G\left(p^2+q^2\right)+x_G\left(rp-\dot{q}\right)+y_G\left(rq+\dot{p}\right)\right]= \\
& \quad+\frac{\rho}{2}\ell^4 Z'_{\dot{q}}\dot{q} \\
& \quad+\frac{\rho}{2}\ell^3\left[Z'_{\dot{w}}\dot{w}+Z'_q uq+Z'_{vp}vp\right] \\
& \quad+\frac{\rho}{2}\ell^2\left[Z'_* u^2+Z'_w uw\right] \\
& \quad+\frac{\rho}{2}\ell^2\left[Z'_{|w|}u\left|w\right|+Z'_{ww}\left|w\left(v^+w^2\right)\right|^{\frac{1}{2}}\right] \\
& \quad+\frac{\rho}{2}\ell^2\left[Z'_{\delta s}u^2\delta_s+Z'_{\delta b}u^2\delta_b+Z'_{\delta s\eta}u^2\delta_s\left(\eta-\frac{1}{C}\right)C\right] \\
& \quad+\frac{\rho}{2}C_d\int_\ell b\left(x\right)w\left(x\right)\left\{w\left(x\right)^2+v\left(x\right)^2\right\}^{\frac{1}{2}}dx \\
& \quad+\frac{\rho}{2}\ell\bar{C}_L\int_{x_2}^{x_1}v\left(x\right)\bar{v}_{FW}\left[t-\tau\left(x\right)\right]dx \\
& \quad+\left(W-B\right)\cos\left(\theta\right)\cos\left(\theta\right)
\end{aligned}
\tag{A.11}
$$

Roll

$$
\begin{aligned}
& I_x\dot{p}+\left(I_x-I_y\right)qr-\left(\dot{r}+pq\right)I_{xx}+\left(r^2-q^2\right)I_{yz}+\left(pr-\dot{q}\right)I_{xy} \\
& \quad+m\left[y_g\left(\dot{w}-uq+vp\right)-x_g\left(\dot{v}-wp+ur\right)\right]= \\
& \quad+\frac{\rho}{2}\ell^5\left[{K_{\dot{p}}}'\dot{p}+K'_{\dot{r}}\dot{r}+{K_{qr}}'qr+{K_{p|p|}}'p\left|p\right|\right] \\
& \quad+\frac{\rho}{2}\ell^4\left[{K_p}'up+{K_r}'ur+{K_{\dot{v}}}'\dot{v}+{K_{wp}}'wp\right] \\
& \quad+\frac{\rho}{2}\ell^3\left[{K_*}'u^2+{K_{vR}}'uv+{K_i}'uv_{FW}\left(t-\tau_T\right)\right] \\
& \quad+\frac{\rho}{2}\ell^3\left[{K_{\delta_r}}'u^2\delta_r+{K_{\delta_{r\eta}}}'u^2\delta_r\left(\eta-\frac{1}{C}\right)C\right] \\
& \quad+\frac{\rho}{2}\ell^3\left(u^2+{v_s}^2+{w_s}^2\right){\beta_s}^2\left[{K_{4s}}'\sin\left(4\phi_s\right)+{K_{8s}}'\sin\left(8\phi_s\right)\right] \\
& \quad+\frac{\rho}{2}\ell^2{x_1}'\bar{C}_L\int_{x_2}^{x_1}w\left(x\right)\bar{v}_{FW}\left(t-\tau\left[x\right]\right)dx \\
& \quad+\left(y_gW-y_BB\right)\cos\left(\theta\right)\cos\left(\phi\right)-\left(x_gW-x_BB\right)\cos\left(\theta\right)\sin\left(\phi\right) \\
& \quad+Q_p
\end{aligned}
\tag{A.12}
$$

Pitch

$$
\begin{aligned}
&I_y\dot{q} + (I_x - I_z)\, rp - (\dot{q} + qr)\, I_{xy} + \left(p^2 - r^2\right) I_{zx} + (qp - \dot{r})\, I_{yz} \\
&\quad + m\left[z_g\left(\dot{u} - vr + wq\right) - x_g\left(\dot{w} - uq + vp\right)\right] = \\
&\quad + \frac{\rho}{2}\ell^5\left[M'_{\dot{q}}\dot{q} + M_{rp}{}'rp\right] \\
&\quad + \frac{\rho}{2}\ell^4\left[M'_{\dot{w}}\dot{w} + M_q{}'uq\right] \\
&\quad + \frac{\rho}{2}\ell^3\left[M'_* u^2 + M'_w uw + M'_{w|w|R} w\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right|\right] \\
&\quad + \frac{\rho}{2}\ell^3\left[M'_{|w|} u\left|w\right| + M'_{ww}\left|w\left(v^2 + w^2\right)^{\frac{1}{2}}\right|\right] \\
&\quad + \frac{\rho}{2}\ell^3\left[M_{\delta_s}{}'u^2\delta_s + M_{\delta b}{}'u^2\delta_b + M_{\delta_{s\eta}}{}'u^2\delta_s\left(\eta - \frac{1}{C}\right)C\right] \\
&\quad + \frac{\rho}{2}C_d\int_\ell xb\left(x\right)w\left(x\right)\left\{w\left(x\right)^2 + v\left(x\right)^2\right\}^{\frac{1}{2}}dx \\
&\quad - \frac{\rho}{2}C_L\int_{x_2}^{x_1} xv\left(x\right)\bar{v}_{FW}\left(t - \tau\left[x\right]\right)dx \\
&\quad - \left(x_G W - x_B B\right)\cos\left(\theta\right)\cos\left(\phi\right) - \left(z_G W - z_B B\right)\sin\left(\theta\right)
\end{aligned}
\tag{A.13}
$$

Yaw

$$
\begin{aligned}
&I_z\dot{r} + (I_y - I_x)\, pq - (\dot{q} + rp)\, I_{yz} + \left(q^2 - p^2\right) I_{xy} + (rq - \dot{p})\, I_{zx} \\
&\quad + m\left[x_G\left(\dot{v} - wp + ur\right) - y_G\left(\dot{u} - vr + wq\right)\right] = \\
&\quad + \frac{\rho}{2}\ell^5\left[N'_{\dot{r}}\dot{r} + N'_{\dot{p}}\dot{p} + N'_{pq}pq\right] \\
&\quad + \frac{\rho}{2}\ell^4\left[N'_p up + N'_r ur + N'_{\dot{v}}\dot{v}\right] \\
&\quad + \frac{\rho}{2}\ell^3\left[N'_* u^2 + N'_v uv + N'_{v|v|R} v\left|\left(v^2 + w^2\right)^{\frac{1}{2}}\right|\right] \\
&\quad + \frac{\rho}{2}\ell^3\left[N'_{\delta_r} u^2\delta_r + N'_{\delta_{r\eta}} u^2\delta_r\left(\eta - \frac{1}{C}\right)C\right] \\
&\quad + \frac{\rho}{2}C_d\int_\ell xh\left(x\right)v\left(x\right)\left\{w\left(x\right)^2 + v\left(x\right)^2\right\}^{\frac{1}{2}}dx \\
&\quad - \frac{\rho}{2}C_L\int_{x_2}^{x_1} xw\left(x\right)\bar{v}_{FW}\left(t - \tau\left[x\right]\right)dx \\
&\quad - \left(x_G W - x_B B\right)\cos\left(\theta\right)\sin\left(\phi\right) - \left(y_g W - y_B B\right)\sin\left(\theta\right)
\end{aligned}
\tag{A.14}
$$

A.1.4 Expansion of Fossen's Quasi-Steady State-Space Model Formulation

Fossen's vectorized state-space equations shown in Section 3.4.2 can be expanded to show contributions forces and moments in each degree-of-freedom. These expansions are helpful when considering the coefficients that can be derived from a certain captive motion test. For instance, if a PMM test induces surge velocity, u, sway velocity, v, and sway acceleration, $\dot{v}$, only coefficients that are multiplied by only these factors and products of these factors can be determined.

$$\tau_{fluid} = M_A\dot{\nu} + C_A(\nu)\nu + D\nu = \begin{bmatrix} X_{fluid} \\ Y_{fluid} \\ Z_{fluid} \\ K_{fluid} \\ M_{fluid} \\ N_{fluid} \end{bmatrix} \tag{A.15}$$

$$\begin{aligned} X'_{fluid} = {} & X'_p p + X'_{\dot{p}}\dot{p} + X'_q q + X'_{\dot{q}}\dot{q} + X'_r r + X'_{\dot{r}}\dot{r} + X'_u u + X'_{\dot{u}}\dot{u} + X'_v v + X'_{\dot{v}}\dot{v} \\ & + X'_w w + X'_{\dot{w}}\dot{w} + Z'_{\dot{q}}q^2 - Y'_{\dot{r}}r^2 - Y'_{\dot{p}}pr + Z'_{\dot{p}}pq - Y'_{\dot{q}}qr + Z'_{\dot{r}}qr + X'_{\dot{w}}qu - X'_{\dot{v}}ru \\ & + Y'_{\dot{w}}qv - Y'_{\dot{v}}rv - Y'_{\dot{w}}rw + Z'_{\dot{w}}qw \end{aligned} \tag{A.16}$$

$$\begin{aligned} Y'_{fluid} = {} & Y'_p p + Y'_{\dot{p}}\dot{p} + Y'_q q + Y'_{\dot{q}}\dot{q} + Y'_r r + Y'_{\dot{r}}\dot{r} + X'_{\dot{v}}\dot{u} + Y'_u u + Y'_v v + Y'_{\dot{v}}\dot{v} \\ & + Y'_w w + Y'_{\dot{w}}\dot{w} - Z'_{\dot{p}}p^2 + X'_{\dot{r}}r^2 + X'_{\dot{p}}pr + X'_{\dot{q}}qr - Z'_{\dot{q}}pq - Z'_{\dot{r}}pr - X'_{\dot{w}}pu + X'_{\dot{u}}ru \\ & - Y'_{\dot{w}}pv + X'_{\dot{v}}rv + X'_{\dot{w}}rw - Z'_{\dot{w}}pw \end{aligned} \tag{A.17}$$

$$\begin{aligned} Z'_{fluid} = {} & Z'_p p + Z'_{\dot{p}}\dot{p} + Z'_q q + Z'_{\dot{q}}\dot{q} + Z'_r r + Z'_{\dot{r}}\dot{r} + X'_{\dot{w}}\dot{u} + Z'_u u + Y'_{\dot{w}}\dot{v} + Z'_v v \\ & + Z'_w w + Z'_{\dot{w}}\dot{w} + Y'_{\dot{p}}p^2 - X'_{\dot{q}}q^2 - X'_{\dot{p}}pq + Y'_{\dot{q}}pq - X'_{\dot{r}}qr + Y'_{\dot{r}}pr + X'_{\dot{v}}pu - X'_{\dot{u}}qu \\ & - X'_{\dot{v}}qv + Y'_{\dot{v}}pv - X'_{\dot{w}}qw + Y'_{\dot{w}}pw \end{aligned} \tag{A.18}$$

$$\begin{aligned} K'_{fluid} = {} & K'_p p + K'_{\dot{p}}\dot{p} + K'_q q + K'_{\dot{q}}\dot{q} + K'_r r + K'_{\dot{r}}\dot{r} + K'_u u + K'_v v + K'_w w + X'_{\dot{p}}\dot{u} \\ & + Y'_{\dot{p}}\dot{v} + Z'_{\dot{p}}\dot{w} + M'_{\dot{q}}q^2 - M'_{\dot{q}}r^2 + Y'_{\dot{w}}v^2 - Y'_{\dot{w}}w^2 + K'_{\dot{r}}pq - K'_{\dot{q}}pr - M'_{\dot{p}}qr \\ & + N'_{\dot{r}}qr + X'_{\dot{r}}qu - X'_{\dot{q}}ru - Y'_{\dot{p}}pw + Z'_{\dot{p}}pv + Y'_{\dot{r}}qv - Y'_{\dot{q}}qw - Y'_{\dot{q}}rv + Z'_{\dot{q}}qv \\ & - Y'_{\dot{r}}rw + Z'_{\dot{r}}qw + Z'_{\dot{r}}rv - Z'_{\dot{q}}rw + X'_{\dot{w}}uv - X'_{\dot{v}}uw - Y'_{\dot{v}}vw + Z'_{\dot{w}}vw \end{aligned} \tag{A.19}$$

$$\begin{aligned} M'_{fluid} = {} & K'_{\dot{q}}\dot{p} + M'_{p}p + M'_{q}q + M'_{\dot{p}}\dot{q} + M'_{r}r + M'_{\dot{q}}\dot{r} + M'_{u}u + M'_{v}v + M'_{w}w + X'_{\dot{q}}\dot{u} \\ & + Y'_{\dot{q}}\dot{v} + Z'_{\dot{q}}\dot{w} - K'_{\dot{r}}p^2 + K'_{\dot{r}}r^2 - X'_{\dot{w}}u^2 + X'_{\dot{w}}w^2 + K'_{\dot{p}}pr + K'_{\dot{q}}qr - M'_{\dot{q}}pq \\ & - N'_{\dot{r}}pr - X'_{\dot{r}}pu + X'_{\dot{p}}pw + X'_{\dot{p}}ru - Z'_{\dot{p}}pu - Y'_{\dot{r}}pv + X'_{\dot{q}}qw - Z'_{\dot{q}}qu + Y'_{\dot{p}}rv \\ & + X'_{\dot{r}}rw - Z'_{\dot{r}}pw - Z'_{\dot{r}}ru + Z'_{\dot{p}}rw + X'_{\dot{u}}uw - Y'_{\dot{w}}uv + X'_{\dot{v}}vw - Z'_{\dot{w}}uw \end{aligned} \tag{A.20}$$

$$\begin{aligned} N'_{fluid} = {} & K'_{\dot{r}}\dot{p} + N'_{p}p + M'_{\dot{q}}\dot{q} + N'_{q}q + N'_{r}r + N'_{\dot{r}}\dot{r} + N'_{u}u + N'_{v}v + N'_{w}w + X'_{\dot{r}}\dot{u} \\ & + Y'_{\dot{r}}\dot{v} + Z'_{\dot{r}}\dot{w} + K'_{\dot{q}}p^2 - K'_{\dot{q}}q^2 + X'_{\dot{v}}u^2 - X'_{\dot{v}}v^2 - K'_{\dot{p}}pq + M'_{\dot{p}}pq - K'_{\dot{r}}qr \\ & + M'_{\dot{q}}pr + X'_{\dot{q}}pu - X'_{\dot{p}}pv - X'_{\dot{p}}qu + Y'_{\dot{p}}pu - X'_{\dot{q}}qv + Y'_{\dot{q}}pv + Y'_{\dot{q}}qu - Y'_{\dot{p}}qv \\ & - X'_{\dot{r}}rv + Y'_{\dot{r}}ru + Z'_{\dot{q}}pw - Z'_{\dot{p}}qw - X'_{\dot{u}}uv + Y'_{\dot{v}}uv - X'_{\dot{w}}vw + Y'_{\dot{w}}uw \end{aligned} \tag{A.21}$$

A.2 GPAUV State-Space Models

A.2.1 Control Sub-model

A parametric control model of the following form was used for all state-space models of the GPAUV considered in this study. An interpolation function was used to approximate the reaction in each degree-of-freedom from each control surface based on the data listed in Tables A.1, A.2, A.3 and A.4.

A.2.2 Semi-Empirical Model

The following section details the state-space model created for the GPAUV using DSSP, as described in Section 3.5.1. The coefficients for the GPAUV are included in the following

Table A.1: Nondimensional data forces and moments created by GPAUV starboard dive plane.

Deflection angle, δ (deg)	X'	Y'	Z'	K'	M'	N'
-24	-2.1×10^{-4}	5.7×10^{-5}	8.6×10^{-4}	1.3×10^{-5}	4.7×10^{-4}	-3.4×10^{-5}
-16	-1.0×10^{-4}	2.1×10^{-5}	5.5×10^{-4}	8.0×10^{-6}	3.3×10^{-4}	2.4×10^{-5}
-8	-1.4×10^{-5}	0	2.3×10^{-4}	4.0×10^{-6}	1.7×10^{-4}	-4.0×10^{-6}
0	0	0	0	0	0	0
8	-1.4×10^{-5}	0	-2.8×10^{-4}	-4.0×10^{-6}	-1.3×10^{-4}	7.0×10^{-6}
16	-1.0×10^{-4}	2.1×10^{-5}	-6.0×10^{-4}	-8.0×10^{-6}	-2.9×10^{-4}	-2.0×10^{-5}
24	-2.1×10^{-4}	5.7×10^{-5}	-9.1×10^{-4}	-1.2×10^{-5}	-4.3×10^{-4}	-3.4×10^{-5}

Table A.2: Nondimensional data forces and moments created by GPAUV port dive plane.

Deflection angle, δ (deg)	X'	Y'	Z'	K'	M'	N'
-24	-1.7×10^{-4}	-2.7×10^{-5}	-8.4×10^{-4}	1.2×10^{-5}	-3.4×10^{-4}	1.2×10^{-5}
-16	-7.2×10^{-5}	-1.1×10^{-5}	-5.6×10^{-4}	8.0×10^{-6}	-2.1×10^{-4}	5.0×10^{-6}
-8	-1.7×10^{-5}	-1.0×10^{-5}	-2.9×10^{-4}	5.0×10^{-6}	-8.0×10^{-5}	1.0×10^{-6}
0	0	0	0	0	0	0
8	-1.7×10^{-5}	-2.0×10^{-6}	2.5×10^{-4}	-4.0×10^{-6}	1.2×10^{-4}	1.0×10^{-6}
16	-7.2×10^{-5}	-1.1×10^{-5}	5.2×10^{-4}	-7.0×10^{-6}	2.4×10^{-4}	5.0×10^{-6}
24	-1.7×10^{-4}	-2.7×10^{-5}	8.0×10^{-4}	-1.1×10^{-5}	3.8×10^{-4}	1.2×10^{-5}

Table A.3: Nondimensional data forces and moments created by GPAUV lower rudder.

Deflection angle, δ (deg)	X'	Y'	Z'	K'	M'	N'
-24	-1.6×10^{-4}	-8.2×10^{-4}	4.3×10^{-5}	1.1×10^{-5}	2.0×10^{-5}	3.8×10^{-4}
-16	-6.8×10^{-5}	-5.5×10^{-4}	2.3×10^{-5}	7.0×10^{-6}	1.1×10^{-5}	2.5×10^{-4}
-8	-1.3×10^{-5}	-2.9×10^{-4}	5.0×10^{-6}	3.0×10^{-6}	2.0×10^{-6}	1.3×10^{-4}
0	0	0	0	0	0	0
8	-1.3×10^{-5}	2.4×10^{-4}	5.0×10^{-6}	-4.0×10^{-6}	2.0×10^{-6}	-1.2×10^{-4}
16	-6.8×10^{-5}	5.1×10^{-4}	2.3×10^{-5}	-8.0×10^{-6}	1.1×10^{-5}	-2.4×10^{-4}
24	-1.6×10^{-4}	7.8×10^{-4}	4.3×10^{-5}	-1.1×10^{-5}	2.0×10^{-5}	-3.7×10^{-4}

Table A.4: Nondimensional data forces and moments created by GPAUV upper rudder.

Deflection angle, δ (deg)	X'	Y'	Z'	K'	M'	N'
-24	-1.9×10^{-4}	8.7×10^{-4}	-5.0×10^{-5}	1.2×10^{-5}	-2.3×10^{-5}	-4.1×10^{-4}
-16	-7.5×10^{-5}	5.7×10^{-4}	-2.6×10^{-5}	8.0×10^{-6}	-1.0×10^{-5}	-2.7×10^{-4}
-8	-1.7×10^{-5}	2.8×10^{-4}	-1.2×10^{-5}	4.0×10^{-6}	-4.0×10^{-6}	-1.3×10^{-4}
0	0	0	0	0	0	0
8	-1.7×10^{-5}	-2.8×10^{-4}	-1.2×10^{-5}	-4.0×10^{-6}	-4.0×10^{-6}	1.3×10^{-4}
16	-7.5×10^{-5}	-5.7×10^{-4}	-2.6×10^{-5}	-8.0×10^{-6}	-1.0×10^{-5}	2.7×10^{-4}
24	-1.9×10^{-4}	-8.8×10^{-4}	-5.0×10^{-5}	-1.1×10^{-5}	-2.3×10^{-5}	4.1×10^{-4}

output from DSSP.

```
1
2  --- DRDC Submarine Simulation Program ---------------- Version 5.0 ----
3  ------------------------------------------------- Build 111121A ----
4  --- (c) Copyright Defence R&D Canada ----------------------------------
5
6  Filename root: gpauv
7
8  STAMP      17-FEB-2012  15:23:46
9
10 Configuration Summary:
11 ----------------------
12
13 Hull Label                        Type         Sections         Weight
14   1  #HULL                        HULFOM           21           1.000000
15
16 Lift Label                        Type         Control          Weight
17   1  #SAIL                        SAIL         NONE             1.000000
18   2  #TOPSTERNPLANE               STERNPLANE   NONE             1.000000
19   3  #BOTTOMSTERNPLANE            STERNPLANE   NONE             1.000000
20   4  #STBDSTERNPLANE              STERNPLANE   NONE             1.000000
21   5  #PORTSTERNPLANE              STERNPLANE   NONE             1.000000
22   6  #FAIRING                     ISOLATED     NONE             1.000000
23   7  #STBDSIDESCAN                ISOLATED     NONE             1.000000
24   8  #PORTSIDESCAN                ISOLATED     NONE             1.000000
25
26
27 Plane reversal correction data (in Lumps as PRev-Bow/Stern)
28
29 None
30
31 Reference Origin        :       -0.925100       0.000000       0.000000
32
33 Reference axes in vehicle coordinates
34                     XRV :        1.000000       0.000000       0.000000
35                     YRV :        0.000000       1.000000       0.000000
36                     ZRV :        0.000000       0.000000       1.000000
37
38 ------------------------------------------------------------------------
39
40 Process Simulation Data:
41
42 ------------------------------------------------------------------------
43
44 001  COEFFICNT Sim.  May be output on sim001.txt
45 ===
46
47 (Input)
48  -----
49
50 UMS defined   :                  2.000000
51
52 (End of Input)
53  ------------
54
55 simnnn output :                      COEF
56
57 RHO is :                       1000.000000
58
59 Ref. length default :            2.030000
60
61 LLSR-type estimation of vehicle coefficients.
62 These quantities, except for <..>, are in the Lump model.
```

```
All Nondimensional x 1000.
(* Coefficients in VW, etc, likely have greater uncertainty.)

1.   Zero incidence
       Xuu, Yuu, Zuu    :       -1.342877       0.000000       0.000000
       Kuu0, Muu, Nuu   :        0.000000      -0.000558       0.000000

2.   Static pitch
       Xww0             :        3.475484
       Zuw0, Zwnu0      :       -8.180822     -47.748994
       Zu1w1, Z1wnu1    :        0.000000       0.000000
       Muw0, Mwnu0      :        5.471428      -5.699074
       Mu1w1, M1wnu1    :        0.000000       0.000000

3.   Static yaw
       Xvv0             :       12.708530
       Yuv0, Yvnu0      :      -28.083307     -56.843908
       Zvv              :       41.483922
       Kuv, Kvnu        :       -0.723857      -0.182070
       Mvv              :       12.426119
       Nuv0, Nvnu0      :       -3.122841       7.134327

4.   Static pitch and yaw (*)
       Yvw, Kvw, Nvw    :       40.704853       0.000000     -12.503542

5.   Steady roll rate
       Yup, Yp1p1       :       -1.062333      -0.000056
       Zpp              :        0.000000
       Kup, Kp1p1       :       -0.072244      -0.000008
       Mpp              :        0.000039
       Nup              :        0.209408

6.   Steady pitch rate
       Xqq              :        0.133712
       Zuq0, <Zq1q1>    :       -2.100462      -1.876485
       Muq0, Mq1q1      :       -0.959698      -0.785233

7.   Steady yaw rate
       Xrr              :        0.402961
       Yur0, <Yr1r1>    :        5.074825       1.879752
       Zrr, Kur, Mrr    :        2.027031       0.169142       0.660870
       Nur0, Nr1r1      :       -1.543157      -0.786062

8.   Yaw and roll rate (*)
       Zvp, Mvp         :        2.728135       0.798258

9.   Yaw and pitch rate (*)
       Yvq, Kvq, Nvq    :       13.149495       0.041196      -4.243197

10.  Yaw and yaw rate (*)
       Xvr              :       -2.655751
       Yvnu1r1v1        :       -4.394703
       Zvr, Mvr         :      -20.398445      -6.404262
       Nrnu             :       -1.563715

11.  Pitch and roll rate (*)
       Ywp, Kwp, Nwp    :        2.349796       0.000000      -0.723634

12.  Pitch and pitch rate (*)
       Xwq              :        0.229489
       Zwnu1q1w1        :       -2.218115
       Mqnu             :       -1.171838

13.  Pitch and yaw rate (*)
```

```
      Ywr, Kwr, Nwr    :       -6.598896       0.001828       2.027021

14.  Roll rate and pitch rate (*)
      Ypq, Kpq, Npq    :        0.782009       0.002213      -0.246308

15.  Roll rate and yaw rate (*)
      Xpr, Zpr, Mpr    :       -0.116739      -0.717532      -0.233359

16.  Pitch rate and yaw rate (*)
      Yqr, Kqr, Nqr    :       -2.184630      -0.005767       0.687952

Vehicle coefficients x1000 output on .spr

17.  Bowplanes, ahead
      Xuudbdb          :        0.000000
      Yuudb, Zuudb     :        0.000000       0.000000
      Kuudb, Muudb     :        0.000000       0.000000
      Nuudbdb          :        0.000000

18.  Rudders, ahead
      Xuudrdr0         :        0.000000
      Yuudr0           :        0.000000
      Kuudr0, Muudrdr0 :        0.000000       0.000000
      Nuudr0           :        0.000000

19.  Sternplanes, ahead
      Xuudsds0         :        0.000000
      Yuuds, Zuuds0    :        0.000000       0.000000
      Kuuds0, Muuds0   :        0.000000       0.000000
      Nuudsds0         :        0.000000

20.  Rudders and yaw rate (*)
      Yu1r1dr          :        0.000000
      Nu1r1dr          :        0.000000

21.  Sternplanes and pitch rate (*)
      Zu1q1ds          :        0.000000
      Mu1q1ds          :        0.000000

Control coefficients x1000 output on .spr

Inertial coefficient matrix (FORM).  All Nondimensional x 1000.

Given as [ IM-A IM-B ]
         [ IM-C IM-D ]
                                  udot           vdot           wdot
IM-A  :                  X      11.052143       0.000000       0.000000
                         Y       0.000000      11.052143       0.000000
                         Z       0.000000       0.000000      11.052143
                                  pdot           qdot           rdot
IM-B  :                  X       0.000000       0.049172       0.000000
                         Y      -0.049172       0.000000       0.000000
                         Z       0.000000       0.000000       0.000000
                                  udot           vdot           wdot
IM-C  :                  K       0.000000      -0.049172       0.000000
                         M       0.049172       0.000000       0.000000
                         N       0.000000       0.000000       0.000000
                                  pdot           qdot           rdot
IM-D  :                  K       0.011433       0.000000      -0.002944
                         M       0.000000       0.653287       0.000000
                         N      -0.002944       0.000000       0.653438

Added mass coefficient matrix (ESAM).  All Nondimensional x 1000.
```

```
 Given as [ AM-A AM-B ]
          [ AM-C AM-D ]
                                        udot           vdot           wdot
 AM-A  :                    X       -0.181537       0.000000       0.000000
                            Y        0.000000     -10.717027       0.000000
                            Z        0.000000       0.000000     -10.160124
                                        pdot           qdot           rdot
 AM-B  :                    X        0.000000      -0.000298       0.000000
                            Y        0.000708       0.000000       0.165910
                            Z        0.000000      -0.208411       0.000000
                                        udot           vdot           wdot
 AM-C  :                    K        0.000000       0.000708       0.000000
                            M       -0.000298       0.000000      -0.208411
                            N        0.000000       0.165910       0.000000
                                        pdot           qdot           rdot
 AM-D  :                    K       -0.001616       0.000000       0.002813
                            M        0.000000      -0.590158       0.000000
                            N        0.002813       0.000000      -0.603134

 Routh-Hurwitz Horizontal Plane Stability

 Char. Eqn. A,B,C         :        0.000027       0.000070       0.000025
   B/A  :                          2.575905        B/A > 0
   C/A  :                          0.902798        C/A > 0         STABLE
 and stability index G_h  :                       0.569277
     x'_r, x'_v, and s'_h :        0.258169       0.111199       0.146970

 Routh-Hurwitz Vertical Plane Stability

 Hi-U Char. Eqn. A,B,C    :        0.000026       0.000034      -0.000041
   B/A  :                          1.273532        B/A > 0
   C/A  :                         -1.561828        C/A < 0       UNSTABLE
 and stability index G_v  :                      -5.238405
     x'_q, x'_w, and s'_v :        0.107209       0.668812      -0.561603
 Critical speed U_c, m/s  :                       0.593286
 Critical speed U_c, kt   :                       1.153255

 (Sim complete)

 ------------------------------------------------------------------------
```

A.2.2.1 Input Geometry

The following listing serves to define the geometry and method to be used in producing a state-space model for the GPAUV with DSSP.

```
Plot
        Matlab

AddedMass ESAM          ! ESAM or Simple

CB      -0.9251 0.0000  0.0000
Reference       CB

Hull
        Label   #Hull
        Nose    0
        Tail    -2.03
        Station 21,     Irregular
                0.0000  0.0000
```

```
                -0.0030 0.0340
                -0.0060 0.0460
                -0.0130 0.0620
                -0.0250 0.0840
                -0.0510 0.1140
                -0.1030 0.1440
                -0.1540 0.1600
                -0.2050 0.1700
                -0.3080 0.1740
                -0.4100 0.1760
                -1.5900 0.1760
                -1.6410 0.1720
                -1.6920 0.1640
                -1.7440 0.1480
                -1.7950 0.1280
                -1.8460 0.1040
                -1.8970 0.0800
                -1.9490 0.0540
                -2.0000 0.0300
                -2.0330 0.0200

Lift
        Label   #Sail
        Sail    0.0880
        Planform
                -1.2080 0.0000  -0.0880
                -1.2310 0.0000  -0.2180
                -1.3080 0.0000  -0.2180
                -1.3080 0.0000  -0.0880
        ToC     0.1500

Lift
        Label   #TopSternPlane
        Sternplane      0.03200 0.08800
        Planform
                -1.6010 0.0000  -0.0880
                -1.6010 0.0000  -0.0881
                -1.9620 0.0000  -0.0881
                -1.9620 0.0000  -0.0240
        ToC     0.0141
        Save

Lift
        Label   #BottomSternPlane
        Rotate  180

Lift
        Label   #StbdSternPlane
        Sternplane      0.03200 0.08800
        Planform
                -1.6010 -0.0880 0.0000
                -1.6010 -0.0881 0.0000
                -1.9620 -0.0881 0.0000
                -1.9620 -0.0240 0.0000
        ToC     0.0141
        Save

Lift
        Label   #PortSternPlane
        Rotate  180

Lift
        Label   #Fairing
        Planform
```

```
                -0.3880 0.0000  0.0880
                -0.5020 0.0000  0.1450
                -0.5890 0.0000  0.1450
                -0.7800 0.0000  0.0880
        ToC     0.1837

Lift
        Label   #StbdSideScan
        Planform
                -0.3850 0.0880  0.0000
                -0.4390 0.1050  0.0000
                -0.8550 0.1050  0.0000
                -0.9090 0.0880  0.0000
        Save
Lift
        Label   #PortSideScan
        Rotate  180
```

A.2.3 CFD-based Models

The coefficients for the linear and Taylor series damping state-space models populated from the results of CFD simulations are provided in the following section.

A.2.3.1 Linear-Damping Model

A state-space model with linear damping, using following form and coefficients, was obtained for the GPAUV.

$$\left(\mathbf{M}_{RB}+\mathbf{M}_{A}\right)\dot{\vec{\nu}}+\left(\mathbf{C}_{RB}+\mathbf{C}_{A}\left(\vec{\nu}\right)\right)\vec{\nu}+\mathbf{D}\vec{\nu}+\vec{g}\left(\vec{\eta}\right)=\vec{\tau}_{CS} \tag{A.22}$$

$$\mathbf{M}'_{RB}=\begin{bmatrix} 1000.00 & 0 & 0 & 0 & 4.80 & 0.10 \\ 0 & 1000.00 & 0 & -4.80 & 0 & 0.01 \\ 0 & 0 & 1000.00 & -0.10 & -0.01 & 0 \\ 0 & -4.80 & -0.10 & 0.90 & 0 & 0 \\ 4.80 & 0 & -0.01 & 0 & 50.53 & 0 \\ 0.10 & 0.01 & 0 & 0 & 0 & 50.37 \end{bmatrix}\times 10^{-3} \tag{A.23}$$

$$\mathbf{M}'_{A}=\begin{bmatrix} -0.94 & 0 & -0.03 & 0 & 0 & 0 \\ 0 & -11.42 & 0 & 0 & 0 & -0.22 \\ -0.03 & 0 & -9.87 & 0 & 0.29 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.29 & 0 & -0.63 & 0 \\ 0 & -0.22 & 0 & 0 & 0 & -0.66 \end{bmatrix}\times 10^{-3} \tag{A.24}$$

$$\mathbf{D}' = \begin{bmatrix} -2.44 & 0.05 & -0.46 & 0.11 & 0.19 & -0.03 \\ 0.01 & -73.94 & -0.63 & -1.33 & 0.20 & 14.85 \\ 0.03 & 0.15 & -39.92 & -0.03 & -10.10 & -0.10 \\ -0.01 & -1.42 & 0.01 & -0.12 & -0.01 & 0.27 \\ -0.10 & 0.25 & -9.16 & 0.01 & -3.94 & 0.10 \\ 0 & 14.84 & 0.03 & 0.30 & -0.04 & -5.23 \end{bmatrix} \times 10^{-3} \tag{A.25}$$

A.2.3.2 Nonlinear-Damping Model

A state-space model with nonlinear damping, using following form and coefficients, was obtained for the GPAUV.

$$(\mathbf{M}_{RB} + \mathbf{M}_A)\,\dot{\vec{\nu}} + (\mathbf{C}_{RB} + \mathbf{C}_A(\vec{\nu}))\,\vec{\nu} + \mathbf{D}_{\vec{\nu}^2}\vec{\nu}^2 + \mathbf{D}_{\vec{\nu}|\vec{\nu}|}\vec{\nu}\,|\vec{\nu}| + \vec{g}(\vec{\eta}) = \vec{\tau}_{CS} \tag{A.26}$$

$$\mathbf{M}'_{RB} = \begin{bmatrix} 1000 & 0 & 0 & 0 & 4.80 & 0.10 \\ 0 & 1000 & 0 & -4.80 & 0 & 0.01 \\ 0 & 0 & 1000 & -0.10 & -0.01 & 0 \\ 0 & -4.80 & -0.10 & 0.90 & 0 & 0 \\ 4.80 & 0 & -0.01 & 0 & 50.53 & 0 \\ 0.10 & 0.01 & 0 & 0 & 0 & 50.37 \end{bmatrix} \times 10^{-3} \tag{A.27}$$

$$\mathbf{M}'_A = \begin{bmatrix} -0.94 & 0 & -0.03 & 0 & 0 & 0 \\ 0 & -11.42 & 0 & 0 & 0 & -0.22 \\ -0.03 & 0 & -9.87 & 0 & 0.29 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.29 & 0 & -0.63 & 0 \\ 0 & -0.22 & 0 & 0 & 0 & -0.66 \end{bmatrix} \times 10^{-3} \tag{A.28}$$

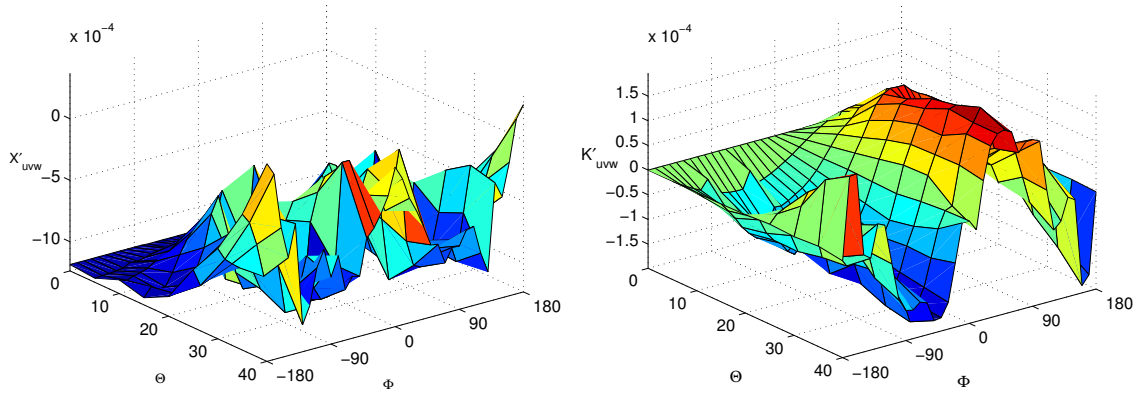
$$\mathbf{D}'_{\nu^2} = \begin{bmatrix} -0.72 & 44.07 & -0.79 & 5.92 & -0.05 & -0.51 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.00 & 11.13 & 1.41 & 5.03 & 1.57 & 1.34 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.03 & 23.55 & 1.28 & -3.90 & -1.33 & -0.73 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times 10^{-3} \tag{A.29}$$

$$\mathbf{D}'_{\nu|\nu|} = \begin{bmatrix} -0.49 & 0 & -3.28 & 0 & 2.23 & 0 \\ 0 & -538.61 & 0 & -15.17 & 0 & 167.46 \\ 0 & 0 & -300.37 & 0 & -114.30 & 0 \\ 0 & -9.94 & 0 & -1.04 & 0 & 3.15 \\ -0.03 & 0 & -68.80 & 0 & -46.06 & 0 \\ 0 & 114.21 & 0 & 3.11 & 0 & -59.81 \end{bmatrix} \times 10^{-3} \tag{A.30}$$

A.2.3.3 Parametric Damping

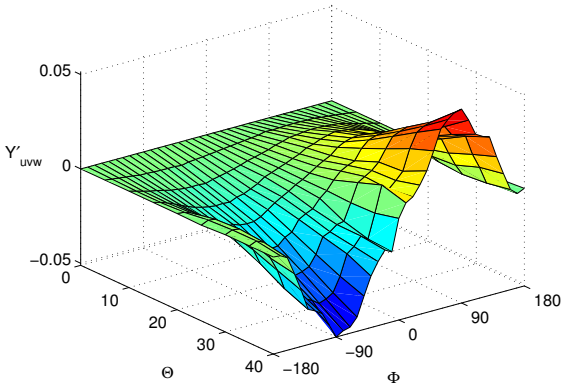
Steady CFD simulations were used to obtain paramentric damping surfaces in the style of Watt [124] for the GPAUV. The results of this analysis were not used in the creation of a state-space model, but are shown in Figure A.1.[1]

[1]Some of the parametric damping surfaces shown in Figure A.1 appear at first to exhibit a very noisy response (e. g., Figure A.1a), but careful attention should be paid to the different scales used from the vertical axes.

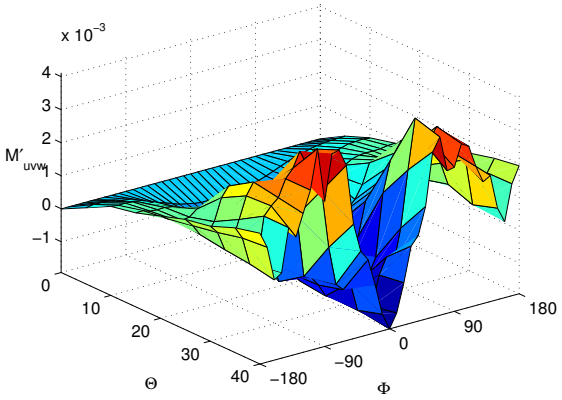


(a) X-force

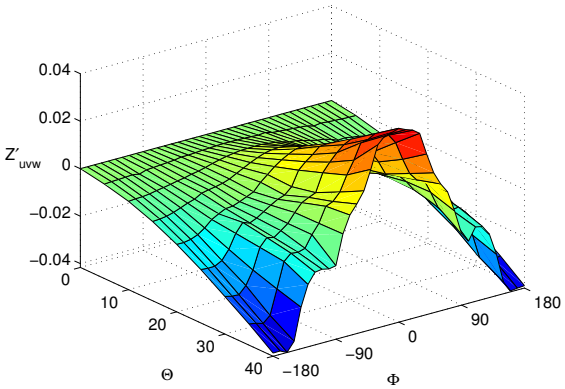
(b) K-moment



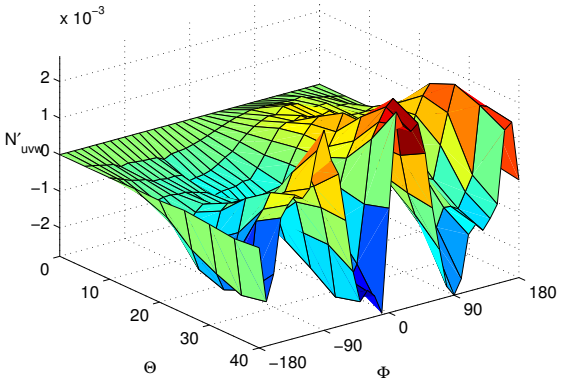
(c) Y-force



(d) M-moment



(e) Z-force



(f) N-moment

Figure A.1: Parametric damping surfaces for the GPAUV.

A.3 Prolate Spheroid State-Space Model

A state-space model with linear damping, using following form and coefficients, was obtained for a 6:1 prolate spheroid.

$$\left(\mathbf{M}_{RB}+\mathbf{M}_{A}\right)\dot{\vec{\nu}}+\left(\mathbf{C}_{RB}+\mathbf{C}_{A}\left(\vec{\nu}\right)\right)\vec{\nu}+\mathbf{D}\vec{\nu}+\vec{g}\left(\vec{\eta}\right)=\vec{\tau}_{CS} \tag{A.31}$$

$$M'_{A}=\operatorname{diag}\begin{bmatrix}-1.26 & -25.5 & 25.5 & 0 & -1.06 & -1.06\end{bmatrix}\times 10^{-3} \tag{A.32}$$

$$D'=\begin{bmatrix}-3.64 & -0.04 & -0.04 & -0.01 & -0.06 & 0.04\\ -0.00 & -18.65 & -0.00 & -0.20 & -0.01 & 3.12\\ -0.03 & -0.01 & -18.64 & -0.02 & -3.42 & 0.06\\ -0.00 & 0.04 & 0.01 & -0.00 & -0.01 & 0.06\\ 0.01 & 0.02 & -4.15 & -0.00 & -0.80 & 0.08\\ -0.01 & 4.21 & -0.05 & -0.01 & 0.02 & -0.83\end{bmatrix}\times 10^{-3} \tag{A.33}$$

A.4 State-Space Model System Identification Code

A.4.1 SixDOFpmm.m

```
% This script systematically applies a number of optimization loops to
% find the added mass, M_A, and damping, D, matrices of a vehicle based on
% data from captive maneuvering tests.
%
% Author:       Ryan G. Coe
% Date:         03-01-2013
% Revisions:    (none)

clc
clear
close all

SDPoptions.plotqual = 'low';
SDPoptions.modeltype = 'Linear';
% SDPoptions.modeltype = 'TaylorRC';

L = 2;
RHO = 997.561;

addpath('Util');
addpath('../util');

ftr = {};
ftr(end+1) = {'testData/GPAUV_u15.csv'};
ftr(end+1) = {'testData/GPAUV_v15.csv'};
ftr(end+1) = {'testData/GPAUV_w15.csv'};
ftr(end+1) = {'testData/GPAUV_p10.csv'};
```

```
ftr(end+1) = {'testData/GPAUV_q10.csv'};
ftr(end+1) = {'testData/GPAUV_r10.csv'};

trim = 100;                                                % used to remove data containing
    transients
for c = 1:size(ftr,2)
    [testData.t{c},testData.nu{c},...
        testData.nuDot{c},testData.TauIn{c}] = importTestData(cell2mat(ftr(c)),trim);
end


% Plot input data
opts.plotqual = 'high';
plotPMM(opts,testData.nu,testData.nuDot,testData.t,testData.TauIn)

%% Options Set

% Options for loops defined by this script and its functions
SDPoptions.maxOuterCycles = 4;
SDPoptions.maxInnerCycles = 1;
SDPoptions.goalChangeTol = 0.01;
SDPoptions.solChangeTol = 0.01;

% Options for fminunc and fmincon
Foptions = optimset();
Foptions.PlotFcns = @optimplotfval;
% Foptions.Display = 'iter';
% Foptions.Display = 'off';
Foptions.TolFun = 1e-5;
Foptions.TolX = 1e-5;
Foptions.MaxFunEvals = 5e4;
% Foptions.MaxIter = 5;
Foptions.Algorithm = 'interior-point';

Foptions.UseParallel = 'always';
matlabpool
matlabpool('addattachedfiles',{'m2c.m','Smtrx.m'});

%% Setup Constraints

load('GPAUVpotentialflowAddedMass2.mat')
[Mai,~] = QSSSMredim(-Ma,zeros(6),RHO,L);
clear Ma
[AeqZ,beq] = FormConstraints(Mai,zeros(6),'M_A');

constraints.Aeq = AeqZ;
constraints.beq = beq;

%% Optimization

[x,fval,output] = OptOneStage(testData,constraints,Foptions,SDPoptions);

% Close matlabpool if necessary
if (matlabpool('size') > 0); matlabpool close; end;

%% Assign output

switch SDPoptions.modeltype
    case 'Linear'
        [Ma,D] = x2MaD(x);
        TauModel = FossenLinear(Ma,D,testData.nu,testData.nuDot);
        [Ma,D] = QSSSMnondim(Ma,D,RHO,L);
    case 'Morison'
        [Ma,D] = x2MaD(x);
```

```
        TauModel = FossenMorison(Ma,D,testData.nu,testData.nuDot);
        [Ma,D] = QSSSMnondim(Ma,D,RHO,L);
    case 'TaylorRC'
        [Ma,Dn2,Dn1n1] = x2MaDnl(x);
        TauModel = FossenTaylorRC(Ma,Dn2,Dn1n1,testData.nu,testData.nuDot);
        [Ma,Dn2] = QSSSMnondim(Ma,Dn2,RHO,L);
        [~,Dn1n1] = QSSSMnondim(zeros(6),Dn1n1,RHO,L);
end

%% Plot data

plotPMM(testData.nu,testData.nuDot,testData.t,testData.TauIn,TauModel,SDPoptions)
```

A.4.2 importTestdata.m

```

function [t,nu,nuDot,TauIn] = importTestData(ftr,trim)
% Imports results from CFD simulation and trims them if necessary

if nargin < 2
    trim = 1;
end

newData1 = importdata(ftr);

t1 = newData1.data(trim:end,1)';
nu1 = newData1.data(trim:end,2:7)';          % nu = [u,v,w,p,q,r]'
nuDot1 = newData1.data(trim:end,8:13)';      % nuDot = [uDot,vDot,wDot,pDot,qDot,rDot]'
TauIn1 = newData1.data(trim:end,14:19)';     % Tau = [X,Y,Z,K,M,N]'
clear newData1;

% trim off initial transients
t = t1;
nu = nu1;
nuDot = nuDot1;
TauIn = TauIn1;
```

A.4.3 FormComnstraints.m

```

function [Aeq,beq] = FormConstraints(M_Ai,Di,type)
% Forms constraints on added mass and damping matrices

Aeq = zeros(57,57);
beq = zeros(57,1);

switch type

    case 'diagonals'
        cvd = [1,3,6,10,15,21,22,29,36,43,50,57];
        b1 = [diag(M_Ai); diag(Di)];
        for i = 1:length(cvd)
            Aeq(cvd(i),cvd(i)) = 1;
            beq(cvd(i)) = b1(i);
        end

    case 'off-diagonals'
        cvod = [...
            2,...
            4:5,...
```

```
            7:9,...
            11:14,...
            16:20,...
            23:28,...
            30:35,...
            37:42,...
            44:49,...
            51:56];
        b1 = [M_Ai(find(triu(ones(6),1))); Di(find((ones(6) - eye(6))))];
        for i = 1:length(cvod)
            Aeq(cvod(i),cvod(i)) = 1;
            beq(cvod(i)) = b1(i);
        end

    case 'M_A'
        indMa = find(triu(ones(6)));
        for i = 1:21
            Aeq(i,i) = 1;
        end
        beq(1:21) = reshape(M_Ai(indMa),21,1);

    case 'D'
        for i = 22:57
            Aeq(i,i) = 1;
        end
        beq(22:57) = reshape(Di',36,1);

    case 'zeros'
        iM_A = zeros(6);
        ind = find(triu(ones(6)));
        iM_A(ind) = [1:21];
        iM_A = iM_A + triu(iM_A,1)';
        iD = vec2mat([22:57],6);

        [~,~,v1] = find(diag(iM_A,1));
        [~,~,v2] = find(diag(iM_A,3));
        [~,~,v3] = find(diag(iM_A,5));
        Cz.M_A = sort([v1;v2;v3]);
        clear v1 v2 v3
        [~,~,v1] = find(diag(iD,1));
        [~,~,v2] = find(diag(iD,3));
        [~,~,v3] = find(diag(iD,5));
        [~,~,v4] = find(diag(iD,-1));
        [~,~,v5] = find(diag(iD,-3));
        [~,~,v6] = find(diag(iD,-5));
        Cz.D = sort([v1;v2;v3;v4;v5;v6]);
        clear v1 v2 v3 v4 v5 v6

        CZ = [Cz.M_A;Cz.D];

        Aeq = zeros(57);
        for i = 1:length(CZ)
            Aeq(CZ(i),CZ(i)) = 1;
        end
        beq = zeros(57,1);

    case 'zerosMa'
        iM_A = zeros(6);
        ind = find(triu(ones(6)));
        iM_A(ind) = [1:21];
        iM_A = iM_A + triu(iM_A,1)';
        iD = vec2mat([22:57],6);

        [~,~,v1] = find(diag(iM_A,1));
```

```
        [~,~,v2] = find(diag(iM_A,3));
        [~,~,v3] = find(diag(iM_A,5));
        Cz.M_A = sort([v1;v2;v3]);
        clear v1 v2 v3
        CZ = [Cz.M_A];

        Aeq = zeros(57);
        for i = 1:length(CZ)
            Aeq(CZ(i),CZ(i)) = 1;
        end
        beq = zeros(57,1);

    otherwise
        fprintf('FormConstraints: ERROR: incorrect type\n')
end
end
```

A.4.4 OptOneStage.m

```

function [x,fval,output] = OptOneStage(PMM,constraints,Foptions,SDPoptions)
% Runs constrained optimization

nu = PMM.nu;
nuDot = PMM.nuDot;
TauIn = PMM.TauIn;

Aeq = constraints.Aeq;
beq = constraints.beq;

x0 = zeros(57,1);

tic
[x,fval,~,output] = ...
    fmincon(@(x)ErrorFull(x,nu,nuDot,TauIn,SDPoptions.modeltype),x0,...
    [],[],Aeq,beq,[],[],[],Foptions);
toc

end
```

A.4.5 ErrorFull.m

```

function modelError = ErrorFull(x,nu,nuDot,TauIn,modeltype)
% Returns error from full system

switch modeltype
    case 'Linear'
        [Ma,D] = x2MaD(x);
        TauModel = FossenLinear(Ma,D,nu,nuDot);
    case 'Morison'
        [Ma,D] = x2MaD(x);
        TauModel = FossenMorison(Ma,D,nu,nuDot);
    case 'TaylorRC'
        [Ma,Dn2,Dn1n1] = x2MaDnl(x);
        TauModel = FossenTaylorRC(Ma,Dn2,Dn1n1,nu,nuDot);
    otherwise
        fprintf('Invalid model type!\n');
        return
end
```

```
19
20 for f = 1:size(nu,2)
21     mE(f) = norm(TauModel{f} - TauIn{f}); % 6x1
22 end
23 modelError = norm(mE); % scalar
24 end
```

A.4.6 FossenLinear.m

```
1
2 function TauOut = FossenLinear(Ma,D,nu,nuDot)
3 % Find output from Linear damping state-space model
4
5 for f = 1:size(nu,2)            % Total number of tests
6
7     nuf = cell2mat(nu(f));
8     nuDotf = cell2mat(nuDot(f));
9
10    for i = 1:size(nu{f},2)     % Number of time steps in current test
11        nui = nuf(:,i);
12        nuDoti = nuDotf(:,i);
13        C_Ai = m2c(Ma,nui);
14        TauOuti(:,i) = Ma*nuDoti + C_Ai*nui + D*nui; % 6x(#-of-time-steps);
15    end
16
17    TauOut{f} = TauOuti(:,:);
18 end
19 end
```

A.4.7 FossenTaylorRC.m

```
1
2 function TauOut = FossenTaylorRC(Ma,Dn2,Dn1n1,nu,nuDot)
3 % Finds output from nonlinear damping (Taylor expansion) state-space model
4
5 for f = 1:size(nu,2)                % Total number of tests
6
7     nuf = cell2mat(nu(f));
8     nuDotf = cell2mat(nuDot(f));
9
10    for i = 1:size(nu{f},2)         % Number of time steps in current test
11
12        nui = nuf(:,i);
13        nuDoti = nuDotf(:,i);
14
15        C_Ai = m2c(Ma,nui);
16
17        TauOuti(:,i) = Ma*nuDoti + C_Ai*nui + Dn2*(nui).^2 + Dn1n1*(nui.*abs(nui)); % 6x(#-
              of-time-steps);
18    end
19
20    TauOut{f} = TauOuti(:,:);
21 end
22
23 end
```

A.4.8 QSSSMnondim.m

```
1
2  function [M_And,Dnd] = QSSSMnondim(M_Ad,Dd,rho,L)
3  % Nondimesionalizes added mass and damping matrices
4
5  if nargin == 2
6      rho = 997.561;
7      L = 2.03;
8      fprintf('No constants input. Setting automatically\n')
9      fprintf('L = %.2f    rho =   %.3f\n',L,rho)
10 end
11
12 d2 = 0.5 * rho * L^2;
13 d3 = 0.5 * rho * L^3;
14 d4 = 0.5 * rho * L^4;
15 d5 = 0.5 * rho * L^5;
16
17 D2 = d2*ones(3,3);
18 D3 = d3*ones(3,3);
19 D4 = d4*ones(3,3);
20 D5 = d5*ones(3,3);
21
22 DM = [D3 D4 ; D4 D5];
23 DD = [D2 D3 ; D3 D4];
24
25 M_And = M_Ad./DM;
26 Dnd = Dd./DD;
```

A.4.9 QSSSMredim.m

```
1
2  function [M_Ad,Dd] = QSSSMredim(M_And,Dnd,rho,L)
3  % Redimensionalizes added mass and damping matrices
4
5  if nargin == 2
6      rho = 997.561;
7      L = 2.03;
8      fprintf('No constants input. Setting automatically\n')
9      fprintf('L = %.2f    rho =   %.3f\n',L,rho)
10 end
11
12 d2 = 0.5 * rho * L^2;
13 d3 = 0.5 * rho * L^3;
14 d4 = 0.5 * rho * L^4;
15 d5 = 0.5 * rho * L^5;
16
17 D2 = d2*ones(3,3);
18 D3 = d3*ones(3,3);
19 D4 = d4*ones(3,3);
20 D5 = d5*ones(3,3);
21
22 DM = [D3 D4 ; D4 D5];
23 DD = [D2 D3 ; D3 D4];
24
25 M_Ad = M_And.*DM;
26 Dd = Dnd.*DD;
```

Appendix B

VFRM Implementation

The Java macro and other supporting material used to run VFRM simulations are supplied within the following sections.

B.1 Required Libraries

The VFRM macro uses methods from the following open source Java libraries, which must be located in STAR-CCM+'s classpath (the classpath can be specified when launching or running STAR-CCM+ in batch model with the flag '-classpath /the/absolute/class/path/'). These libraries and the locations from which they can be downloaded, are shown in Table B.1.

Table B.1: Nonstandard Java libraries used by VFRM.java.

Name	Location
Apache Commons Lang	`http://commons.apache.org/proper/commons-lang`
Apache Commons Math	`http://commons.apache.org/proper/commons-math`
Jama	`http://math.nist.gov/javanumerics/jama`
OpenCSV	`http://opencsv.sourceforge.net`

B.2 VFRM Macro Code

The following code listing gives the Java source code used in conjunction with Star-CCM+ to perform VFRM simulations.

```
1  package macro;
2
3  import java.io.IOException;
4  import java.io.File;
5  import java.io.FileInputStream;
6  import java.lang.reflect.Array;
7  import java.text.DateFormat;
8  import java.text.DecimalFormat;
9  import java.text.SimpleDateFormat;
10 import java.util.Arrays;
11 import java.util.Collection;
12 import java.util.Date;
13 import java.util.Scanner;
14
15 import java.awt.Toolkit;
16 import java.io.FileWriter;
17 import java.util.ArrayList;
18 import java.util.List;
19 import java.util.logging.Level;
20 import java.util.logging.Logger;
21 import javax.swing.JOptionPane;
22 import Jama.*;
23 import au.com.bytecode.opencsv.CSVWriter;
24 import java.io.BufferedWriter;
25 import java.io.PrintWriter;
26 import org.apache.commons.lang3.StringUtils;
27 import org.apache.commons.lang3.ArrayUtils;
28 import org.apache.commons.math3.analysis.UnivariateFunction;
29 import org.apache.commons.math3.analysis.interpolation.SplineInterpolator;
30 import org.apache.commons.math3.analysis.interpolation.UnivariateInterpolator;
31 import org.apache.commons.math3.analysis.polynomials.PolynomialFunction;
32 import org.apache.commons.math3.geometry.euclidean.threed.Vector3D;
33
34 import star.common.*;
35 import star.base.neo.*;
36 import star.sixdof.*;
37 import star.base.report.*;
38 import star.motion.*;
39
40 /**
41  * This macro controls a virtual free-running model (VFRM) simulation within
42  * STAR-CCM+. It has been written specifically for use with the Virginia Tech
43  * general purpose autonomous underwater vehicle (GPAUV), but may be adapted for
44  * use with other vehicles.
45  *
46  * @author Ryan Coe
47  * @author Brian McCarter
48  */
49 public class VFRM extends StarMacro {
50
51     // Constants and settings
52     public static final double DINC = 0.005;            // Min increment for CS position (
           rad)
53     public static final double RHO = 997.561;           // Fluid density (kg/m3)
54     public static final double L = 2.03;                // Vehicle length (m)
55     public static final double RPM = 1125;              // Propeller RPM
56     public static final String MACROPREFIX = "*Macro:"; // Used in messages to Star-CCM+
           output window
57     public static final String CONTROLTYPE = "Hinf";    // "Hinf" or "PI"
58     public static final boolean EXTERNALCS = false;
59     public static double[] refHeadR = //
60             {0, 0, 0, 0, Math.toRadians(0), Math.toRadians(0)};  // {phi,theta,psi}
61     public static final double INITTIME = 0.01;         // Time before maneuver begins
62     //*********************************************************************
```

```
63      public Simulation sim = null;
64      public String startTime;
65      public String baseName;
66      public CFDio cfdIO = null;
67      public CFDvehicle cfdVehicle = null;
68
69      public void execute() {
70          int controllerInc;
71          int iterationNum;
72
73          sim = getActiveSimulation();
74
75          // Create macro objects.
76          AUVcontroller vController = null;
77          try {
78              cfdIO = new CFDio();
79              cfdVehicle = new CFDvehicle();
80              vController = new AUVcontroller();
81          } catch (Exception e) {
82              sayL(e.getMessage());
83              StackTraceElement[] eS = e.getStackTrace();
84              fatalError(eS);
85          }
86
87          startTime = getDateAndTime();
88          ouputHeader("Beginning macro execution");
89
90          // ********************************************************************
91          // Begin simulation iteration
92          iterationNum = 0;
93          while (cfdIO.simState.tSim < cfdIO.simState.tLim) {
94              try {
95
96                  // Allow initialization period before beginning maneuver
97                  if (cfdIO.simState.tSim >= INITTIME) {
98                      cfdVehicle.vehState.unlockVehicleMotion();
99                  } else {
100                     cfdVehicle.vehState.lockVehicleMotion();
101                 }
102
103                 // Refresh vehicle state (eta,nu,nuDot)
104                 cfdVehicle.vehState.refreshVehicleState();
105
106                 // Refresh reference heading for pseudo zig-zag
107                 refHeadR[5] = pseudoZigZag(Math.toRadians(-10), Math.toRadians(10), 0.1,
                        cfdIO.simState.tSim);
108
109                 // Call control algorithm at designated controller frequency.
110                 controllerInc = vController.getIncrement(10, cfdIO.simState.tStep);
111                 if (iterationNum == 0 || iterationNum % controllerInc == 0) {
112                     vController.iterate(cfdVehicle.vehState.eta, cfdVehicle.vehState.etaDot,
                            refHeadR);
113                 }
114
115                 // Adjust flap rotation rates and propeller output.
116                 cfdVehicle.vControlSystem.setControlInput(RPM, vController.flapTargetR);
117
118                 // Iterate solution.
119                 cfdIO.runSim(1);
120                 iterationNum++;
121
122                 // Refresh simulation time.
123                 cfdIO.simState.refreshSimTimeState();
124
```

```
125             } catch (Exception e) {
126                 StackTraceElement[] eStr = e.getStackTrace();
127                 fatalError(eStr);
128                 break;
129             }
130         }
131         // Report macro complete.
132         ouputHeader("Finished Macro Execution");
133         Toolkit.getDefaultToolkit().beep();
134     }
135 
136     /**
137      * Class for interaction with solution iteration methods.
138      */
139     public class CFDio {
140 
141         private SimState simState = null;
142 
143         public CFDio() {
144             sayL("CFDio object.", 1);
145 
146             baseName = sim.getPresentationName();
147             String[] outA = baseName.split("@", 2);
148             baseName = outA[0];
149 
150             sim.getInterfaceManager().initialize();
151             simState = new SimState();
152             simState.refreshSimTimeState();
153 
154         }
155 
156         /**
157          * Runs simulation for numOfTsteps and refreshes simulation times.
158          * @param numOfTsteps number of steps to iterate
159          */
160         public void runSim(int numOfTsteps) {
161             sim.getSimulationIterator().run(numOfTsteps, true);
162         }
163 
164         /**
165          * Gets simulation time level, time step size and time limit. Also finds
166          * overset cell error status.
167          */
168         public class SimState {
169 
170             public Solution simSolutionL;
171             public PhysicalTimeStoppingCriterion simTimeStopCritL;
172             public ImplicitUnsteadySolver simSolverL;
173             public double tSim, tLim, tStep;
174 
175             /**
176              * SimState class constructor.
177              */
178             public SimState() {
179                 sayL("SimTimeState object.", 2);
180                 simSolutionL = sim.getSolution();
181                 simTimeStopCritL =
182                         ((PhysicalTimeStoppingCriterion) sim.
                                getSolverStoppingCriterionManager().
183                         getSolverStoppingCriterion("Maximum Physical Time"));
184                 simSolverL = ((ImplicitUnsteadySolver) sim.getSolverManager().
185                         getSolver(ImplicitUnsteadySolver.class));
186             }
187 
```

```
188             /**
189              * Sets simulation stopping criteria.
190              * @param timeLimit simulation time limit (s)
191              */
192             public void setSimTimeLimit(double timeLimit) {
193                 simTimeStopCritL.setMaximumTime(timeLimit);
194             }
195 
196             /**
197              * Sets simulation time-step.
198              * @param timeStep simulation time-step size (s)
199              */
200             public void setSimTimeStep(double timeStep) {
201                 simSolverL.getTimeStep().setValue(timeStep);
202             }
203 
204             /**
205              * Refreshes values for simulation time (current, step size and
206              * limit).
207              */
208             public void refreshSimTimeState() {
209                 tSim = simSolutionL.getPhysicalTime();
210                 tLim = simTimeStopCritL.getMaximumTime().getValue();
211                 tStep = simSolverL.getTimeStep().getValue();
212             }
213 
214             /**
215              * Reports simulation time state to output window
216              */
217             public void reportSimTimeState() {
218                 sayL("SIMULATION TIME STATUS", 1);
219                 sayL(" Current simulation time (sec): " + tStep, 2);
220                 sayL("      Current time level (sec): " + tSim, 2);
221                 sayL("      Current time limit (sec): " + tLim, 2);
222             }
223         }
224     }
225 
226     public class CFDvehicle {
227 
228         public VehicleState vehState = null;
229         private VehicleControlSystem vControlSystem = null;
230         private Body AUV6DOF = null;
231 
232         /**
233          * Constructor method for class CFDvehicle.
234          * @throws Exception
235          */
236         public CFDvehicle() throws Exception {
237             sayL("CFDvehicle object.", 1);
238             // Get AUV 6-DoF Body object.
239             AUV6DOF = ((Body) sim.get(BodyManager.class).getObject("AUV"));
240 
241             // Create VehichleState and VehicleControlSystem objects.
242             vehState = new VehicleState();
243             vControlSystem = new VehicleControlSystem();
244         }
245 
246         /**
247          * VehicleControlSystem class contains subclasses for AUV propeller and
248          * control flap management.
249          */
250         public class VehicleControlSystem {
251 
```

```
252             private CSactuator[] csFlaps = new CSactuator[4];
253             private PropellerModel propModel = null;
254 
255             /**
256              * VehicleControlSystem class constructor.
257              * @throws Exception
258              */
259             public VehicleControlSystem() throws Exception {
260                 sayL("VehicleControlSystem objec.", 2);
261                 // Create PropellerModel objects.
262                 propModel = new PropellerModel();
263 
264                 // Create CSactuator objects.
265                 for (int i = 0; i < 4; i++) {
266                     csFlaps[i] = new CSactuator(i);
267                 }
268 
269                 setControlInput(RPM, new double[]{0, 0, 0, 0});
270             }
271 
272             /**
273              * Sets control input to vehicle.
274              * @param RPM propeller RPM
275              * @param flapTargetR control surface targets (rad)
276              */
277             private void setControlInput(double RPM, double[] flapTargetR) {
278 
279                 propModel.n = RPM / 60;
280                 propModel.SetPropOutput();
281 
282                 for (int i = 0; i < 4; i++) {
283                     csFlaps[i].goToDeflection(flapTargetR[i]);
284                 }
285                 reportControlSystemState();
286             }
287 
288             /**
289              * Prints controlSystem state to output window.
290              */
291             public void reportControlSystemState() {
292 
293                 sayL("Propeller output for " + propModel.n * 60 + " RPM and u (m/s) = " +
                        vehState.u, 1);
294                 sayL("          Va = " + propModel.va, 2);
295                 sayL("           J = " + propModel.J, 2);
296                 sayL("  Thrust (N) = " + propModel.T, 2);
297                 sayL("Torque (N-m) = " + propModel.Q, 2);
298 
299                 sayL("FLAP STATUS", 1);
300                 say("  Flap position (deg): [", 2);
301                 for (int i = 0; i < 4; i++) {
302                     sim.print(Double.toString(Math.toDegrees(csFlaps[i].flapPosR)));
303                     sim.print(", ");
304                 }
305                 sim.print("] \n");
306                 say("  Flap error (deg): [", 2);
307                 for (int i = 0; i < 4; i++) {
308                     sim.print(Double.toString(Math.toDegrees(csFlaps[i].flapErrorR)));
309                     sim.print(", ");
310                 }
311                 sim.print("] \n");
312                 say("  Flap rotation rate (deg/s): [", 2);
313                 for (int i = 0; i < 4; i++) {
314                     sim.print(Double.toString(Math.toDegrees(csFlaps[i].flapRateR)));
```

```
315                     sim.print(", ");
316                 }
317                 sim.print("] \n");
318             }
319 
320             /**
321              * AUV propeller model class.
322              */
323             public class PropellerModel {
324 
325                 public double J, T, Q, va, n;
326                 double D = 0.12; // propeller diameter (m)
327                 double wt = 0.3; // wake fraction
328                 private double[] propTau = {0, 0, 0, 0, 0, 0};
329                 private double[] coefKt = {0.1246, -0.3654, 0.8327, -1.4798, 0.8872};
330                 private double[] coefKq = {0.0051, 0.0024, -0.0111, -0.0093};
331                 private PolynomialFunction polyKt, polyKq;
332                 private UserFieldFunction ff_T, ff_Q;
333                 private PropulsionForce propThurst6DOF = null;
334                 private ExternalMoment propTorque6DOF = null;
335                 private ExpressionReport repPropJ, repPropT, repPropQ, repPropRPM;
336 
337                 /**
338                  * PropellerModel class constructor.
339                  */
340                 public PropellerModel() {
341                     sayL("PropellerModel object.", 3);
342                     polyKt = new PolynomialFunction(coefKt);
343                     polyKq = new PolynomialFunction(coefKq);
344 
345                     // Thrust and Torque fieldfunction objects
346                     ff_T = (UserFieldFunction) sim.getFieldFunctionManager().getFunction("T"
                            );
347                     ff_Q = (UserFieldFunction) sim.getFieldFunctionManager().getFunction("Q"
                            );
348 
349                     propThurst6DOF = ((PropulsionForce) AUV6DOF.
                            getExternalForceAndMomentManager().getObject("Propulsion Thrust"));
350                     propTorque6DOF = ((ExternalMoment) AUV6DOF.
                            getExternalForceAndMomentManager().getObject("Propulsion Torque"));
351 
352                     repPropJ = ((ExpressionReport) sim.getReportManager().getReport("Prop_J"
                            ));
353                     repPropT = ((ExpressionReport) sim.getReportManager().getReport("Prop_T"
                            ));
354                     repPropQ = ((ExpressionReport) sim.getReportManager().getReport("Prop_Q"
                            ));
355                     repPropRPM = ((ExpressionReport) sim.getReportManager().getReport("
                            Prop_RPM"));
356                 }
357 
358                 /**
359                  * Sets propeller output by calling FindPropOutput and using the
360                  * calculated vector to set field functions for rigid-body
361                  * reaction and actuator disk output.
362                  */
363                 public void SetPropOutput() {
364                     FindPropOutput();
365                     T = propTau[0];
366                     Q = propTau[3];
367 
368                     // Set actuator disk thrust and torque
369                     ff_T.setDefinition(Double.toString(T));
370                     ff_Q.setDefinition(Double.toString(Q));
```

```
371
372                     // Set rigid-body propeller thrust and torque
373                     propThurst6DOF.getThrust().setValue(T);
374                     propTorque6DOF.getMoment().setComponents(-Q, 0, 0); // <- must apply
                            negative torque to vehicle for right-hand propeller
375
376                     repPropJ.setDefinition(Double.toString(J));
377                     repPropT.setDefinition(Double.toString(T));
378                     repPropQ.setDefinition(Double.toString(Q));
379                     repPropRPM.setDefinition(Double.toString(n * 60));
380                 }
381
382                 /**
383                  * Finds propeller output based on commanded RPM and vehicle
384                  * speed.
385                  * @return An array containing propeller thrust and torque
386                  */
387                 public double[] FindPropOutput() {
388                     va = vehState.u * (1 - wt);
389                     J = va / (n * D); // Advance ratio
390                     if (J < 0) {
391                         sayL("ERROR: PROPELLER ADVANCE RATIO LESS THAN ZERO", 0);
392                     }
393
394                     // Find Kt and Kq based on polynomials
395                     double Kt = polyKt.value(J);
396                     double Kq = polyKq.value(J);
397
398                     // Find thrust (T) and torque (Q)
399                     T = Kt * RHO * Math.pow(n, 2) * Math.pow(D, 4);
400                     Q = Kq * RHO * Math.pow(n, 2) * Math.pow(D, 5);
401                     propTau[0] = T;
402                     propTau[3] = Q;
403
404                     return propTau;
405                 }
406             }
407
408             /**
409              * Class to simulate control surface servo.
410              */
411             public class CSactuator {
412
413                 private final double DRMAX = 6.3; // Max CS rotational rate rounded to
                        nearest 0.1 (rad/s)
414                 private String flapName = null;
415                 private int flapID;
416                 private double flapPosR = 0;
417                 private double flapRateR = 0;
418                 private double flapErrorR = 0;
419                 private double apRateSign;
420                 private RotatingMotion cSysRotMotionR;
421                 private CartesianCoordinateSystem flapCSys;
422                 private ExpressionReport repFlapError, repFlapPosition, repFlapRate;
423                 private SixDofPlusRotatingMotion flapRotMotionR;
424                 private double targR, dt;
425
426                 /**
427                  * CSactuator constructor.
428                  * @param flapIDi flap name (STBD, Port, Lower or Upper)
429                  * @throws Exception
430                  */
431                 public CSactuator(int flapIDi) throws Exception {
432                     flapID = flapIDi;
```

```
433                     switch (flapID) {
434                         case 0: // STBD Flap
435                             flapName = "STBD";
436                             apRateSign = 1;
437                             break;
438                         case 1: // Port Flap
439                             flapName = "Port";
440                             apRateSign = 1;
441                             break;
442                         case 2: // Lower Flap
443                             flapName = "Lower";
444                             apRateSign = 1;
445                             break;
446                         case 3: // Upper Flap
447                             flapName = "Upper";
448                             apRateSign = 1;
449                             break;
450                         default:
451                             throw new Exception("INVALID FLAP DESIGNATION");
452                     }
453                     LabCoordinateSystem cSysLab = sim.
454                             getCoordinateSystemManager().
455                             getLabCoordinateSystem();
456                     CartesianCoordinateSystem cSysAUV =
457                             ((CartesianCoordinateSystem) cSysLab
458                             .getLocalCoordinateSystemManager()
459                             .getObject("AUV-CSys"));
460                     CartesianCoordinateSystem cSysFSD =
461                             ((CartesianCoordinateSystem) cSysAUV
462                             .getLocalCoordinateSystemManager()
463                             .getObject("FSD"));
464                     flapCSys = ((CartesianCoordinateSystem) cSysFSD
465                             .getLocalCoordinateSystemManager()
466                             .getObject("CSys-" + flapName));
467                     cSysRotMotionR = ((RotatingMotion) sim.
468                             get(MotionManager.class).
469                             getObject("CSys " + flapName));
470                     flapRotMotionR = ((SixDofPlusRotatingMotion) sim.
471                             get(MotionManager.class).
472                             getObject(flapName));
473
474                     // Create simulation report objects.
475                     repFlapPosition = ((ExpressionReport) sim.
476                             getReportManager().
477                             getReport("FlapPosition_" + flapName));
478                     repFlapError = ((ExpressionReport) sim.getReportManager().
479                             getReport("FlapError_" + flapName));
480                     repFlapRate = ((ExpressionReport) sim.getReportManager().
481                             getReport("FlapRate_" + flapName));
482
483                     // Zero flap motions
484                     flapRotMotionR.getRotationRate().setValue(0);
485                     cSysRotMotionR.getRotationRate().setValue(0);
486
487                     // Refresh flap position
488                     flapPosR = refreshFlapPosition();
489
490                     sayL("CSactuator " + flapName + " object.", 3);
491                     sayL("Postion (deg): " + Math.toDegrees(flapPosR), 4);
492                 }
493
494                 /**
495                  * Finds flap position, flap error, necessary rotation rate and
496                  * set's that rate.
```

```
497                  * @param csTargetR target flap deflection (rad)
498                  */
499                 public void goToDeflection(double csTargetR) {
500                     targR = csTargetR;
501                     dt = cfdIO.simState.tStep;
502                     try {
503                         refreshFlapPosition();
504                         getFlapError();
505                         getFlapRate();
506                         setFlapRate();
507                         recordFlapState();
508                     } catch (Exception ex) {
509                         StackTraceElement[] eS = ex.getStackTrace();
510                         fatalError(eS);
511                     }
512                 }
513
514                 /**
515                  * Records flap state to simulation reports.
516                  */
517                 public void recordFlapState() {
518                     repFlapPosition.setDefinition(Double.toString(flapPosR));
519                     repFlapError.setDefinition(Double.toString(flapErrorR));
520                     repFlapRate.setDefinition(Double.toString(flapRateR));
521                 }
522
523                 /**
524                  * Set's flap rate to 6-DOF Superposed Rotation motion.
525                  */
526                 public void setFlapRate() {
527                     double apFlapRateR;
528                     apFlapRateR = apRateSign * flapRateR;
529                     flapRotMotionR.getRotationRate().setValue(apFlapRateR);
530                     cSysRotMotionR.getRotationRate().setValue(apFlapRateR);
531                 }
532
533                 /**
534                  * Calculates necessary flap rotation rate given error and
535                  * time-step size.
536                  * @return flap rate to move flap
537                  */
538                 public double getFlapRate() {
539                     if (Math.abs(flapErrorR) < DRMAX * dt) {
540                         flapRateR = roundToDec(flapErrorR, DINC) / dt;
541                     } else {
542                         flapRateR = roundToDec(Math.signum(flapErrorR) * DRMAX, 0.1);
543                     }
544                     return flapRateR;
545                 }
546
547                 /**
548                  * Finds flap position error.
549                  * @return error in flap position
550                  */
551                 public double getFlapError() {
552                     flapErrorR = targR - flapPosR;
553                     return flapErrorR;
554                 }
555
556                 /**
557                  * Refreshes flap position.
558                  * @return current flap position (rad)
559                  * @throws Exception
560                  */
```

```
561                 public double refreshFlapPosition() throws Exception {
562                     Vector3D vCSysFlap = new Vector3D(flapCSys.getBasis0().toDoubleArray());
563                     double alpha = vCSysFlap.getAlpha();
564                     double delta = vCSysFlap.getDelta();
565 
566                     switch (flapID) {
567                         case 0: // STBD Flap
568                             flapPosR = -delta;
569                             break;
570                         case 1: // Port Flap
571                             flapPosR = delta;
572                             break;
573                         case 2: // Lower Flap
574                             flapPosR = alpha;
575                             break;
576                         case 3: // Upper Flap
577                             flapPosR = -alpha;
578                             break;
579                         default:
580                             throw new Exception("ERROR FINDING FLAP POSITION");
581                     }
582                     return flapPosR;
583                 }
584             }
585         }
586 
587         /**
588          * Vehicle state.
589          */
590         public class VehicleState {
591 
592             private ArrayList<Report> repEta = new ArrayList<Report>();
593             private ArrayList<Report> repEtaDot = new ArrayList<Report>();
594             private ArrayList<Report> repNu = new ArrayList<Report>();
595             private ArrayList<Report> repNuDot = new ArrayList<Report>();
596             public double[] eta, etaDot, nu, nuDot;
597             public double u;
598 
599             /**
600              * VehicleState class constructor.
601              */
602             public VehicleState() {
603                 sayL("VehicleState object.", 2);
604 
605                 // eta
606                 eta = new double[6]; // eta = {p;THETA} = {N;E;D;phi;theta;psi}
607                 repEta.add(sim.getReportManager().getReport("eta_N"));
608                 repEta.add(sim.getReportManager().getReport("eta_E"));
609                 repEta.add(sim.getReportManager().getReport("eta_D"));
610                 repEta.add(sim.getReportManager().getReport("eta_phi"));
611                 repEta.add(sim.getReportManager().getReport("eta_theta"));
612                 repEta.add(sim.getReportManager().getReport("eta_psi"));
613 
614                 // etaDot
615                 etaDot = new double[6]; // etaDot = {pDot;THETADot} = {NDot;EDot;DDot;phiDot
                        ;thetaDot;psiDot}
616                 repEtaDot.add(sim.getReportManager().getReport("etaDot_NDot"));
617                 repEtaDot.add(sim.getReportManager().getReport("etaDot_EDot"));
618                 repEtaDot.add(sim.getReportManager().getReport("etaDot_DDot"));
619                 repEtaDot.add(sim.getReportManager().getReport("etaDot_phiDot"));
620                 repEtaDot.add(sim.getReportManager().getReport("etaDot_thetaDot"));
621                 repEtaDot.add(sim.getReportManager().getReport("etaDot_psiDot"));
622 
623                 // nu
```

```
624                 nu = new double[6]; // nu = {v;omega} = {u;v;w;p;q;r}
625                 repNu.add(sim.getReportManager().getReport("nu_u"));
626                 repNu.add(sim.getReportManager().getReport("nu_v"));
627                 repNu.add(sim.getReportManager().getReport("nu_w"));
628                 repNu.add(sim.getReportManager().getReport("nu_p"));
629                 repNu.add(sim.getReportManager().getReport("nu_q"));
630                 repNu.add(sim.getReportManager().getReport("nu_r"));
631
632                 // nuDot
633                 nuDot = new double[6]; // nuDot = {vDot;omegaDot} = (uDot;vDot;wDot;pDot;
                        qDot;rDot}
634                 repNuDot.add(sim.getReportManager().getReport("nuDot_uDot"));
635                 repNuDot.add(sim.getReportManager().getReport("nuDot_vDot"));
636                 repNuDot.add(sim.getReportManager().getReport("nuDot_wDot"));
637                 repNuDot.add(sim.getReportManager().getReport("nuDot_pDot"));
638                 repNuDot.add(sim.getReportManager().getReport("nuDot_qDot"));
639                 repNuDot.add(sim.getReportManager().getReport("nuDot_rDot"));
640
641                 refreshVehicleState();
642             }
643
644             /*
645              * Locks all degrees of freedom except x-translation.
646              */
647             public void lockVehicleMotion() {
648                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeTranslationY(false);
649                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeTranslationZ(false);
650                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeRotationX(false);
651                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeRotationY(false);
652                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeRotationZ(false);
653             }
654
655             /*
656              * Unlocks all degrees of freedom.
657              */
658             public void unlockVehicleMotion() {
659                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeTranslationY(true);
660                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeTranslationZ(true);
661                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeRotationX(true);
662                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeRotationY(true);
663                 ((BodyFreeMotion) AUV6DOF.getMotionType()).setFreeRotationZ(true);
664             }
665
666             /**
667              * Gets vehicle state (eta = orientation; nu = velocities; nuDot =
668              * accelerations).
669              */
670             private void refreshVehicleState() {
671                 for (int i = 0; i < 6; i++) {
672                     eta[i] = repEta.get(i).getReportMonitorValue();
673                     etaDot[i] = repEtaDot.get(i).getReportMonitorValue();
674                     nu[i] = repNu.get(i).getReportMonitorValue();
675                     nuDot[i] = repNuDot.get(i).getReportMonitorValue();
676                 }
677                 u = nu[0];
678                 // Offset to initially commanded depth
679                 sayL("Vehicle state refreshed.", 1);
680                 sayL("   Eta: " + Arrays.toString(eta), 2);
681                 sayL("EtaDot: " + Arrays.toString(etaDot), 2);
682                 sayL("    Nu: " + Arrays.toString(nu), 2);
683                 sayL(" NuDot: " + Arrays.toString(nuDot), 2);
684             }
685         }
686     }
```

```
687
688     private class AUVcontroller {
689
690         private Hinf myHinf;
691         private DepthController myDepthControl;
692         public ExpressionReport[] repEtaRef = new ExpressionReport[3];
693         public ExpressionReport[] repEtaError = new ExpressionReport[3];
694         public ExpressionReport[] repFlapCommand = new ExpressionReport[4];
695         private double[] uErrorD = new double[3];
696         private double[] uErrorR = new double[3];
697         private double[] flapTargetD = new double[4];
698         private double[] flapTargetR = new double[4];
699
700         private AUVcontroller() {
701             sayL("AUVcontroller object.", 1);
702
703             repEtaRef[0] = ((ExpressionReport) sim.getReportManager().getReport("etaRef_phi"
                    ));
704             repEtaRef[1] = ((ExpressionReport) sim.getReportManager().getReport("
                    etaRef_theta"));
705             repEtaRef[2] = ((ExpressionReport) sim.getReportManager().getReport("etaRef_psi"
                    ));
706
707             repEtaError[0] = ((ExpressionReport) sim.getReportManager().getReport("
                    etaError_phi"));
708             repEtaError[1] = ((ExpressionReport) sim.getReportManager().getReport("
                    etaError_theta"));
709             repEtaError[2] = ((ExpressionReport) sim.getReportManager().getReport("
                    etaError_psi"));
710
711             repFlapCommand[0] = ((ExpressionReport) sim.getReportManager().getReport("
                    FlapCommand_STBD"));
712             repFlapCommand[1] = ((ExpressionReport) sim.getReportManager().getReport("
                    FlapCommand_Port"));
713             repFlapCommand[2] = ((ExpressionReport) sim.getReportManager().getReport("
                    FlapCommand_Lower"));
714             repFlapCommand[3] = ((ExpressionReport) sim.getReportManager().getReport("
                    FlapCommand_Upper"));
715
716             try {
717                 myHinf = new Hinf();
718                 myDepthControl = new DepthController();
719             } catch (IOException ex) {
720                 say(ex.getMessage());
721                 fatalError(ex.getStackTrace());
722             }
723         }
724
725         /**
726          * Calls Depth and H-infinity controllers.
727          * @param etaR [x, y, z, phi, theta, psi] (m,rad)
728          * @param etaDotR [xDot, yDot, zDot, phiDot, thetaDot, psiDot]
729          * (m/s,rad/s)
730          * @param etaRefR [x, y, z, phi, theta, psi] (m,rad)
731          * @return
732          */
733         private double[] iterate(double[] etaR, double[] etaDotR, double[] etaRefR) {
734             double[] etaDotD = new double[3];
735
736             // Get pitch reference from depth controller
737             etaRefR[4] = Math.toRadians(myDepthControl.iterate(etaR[2], etaRefR[2]));
738
739             // Find phi,theta,psi error
740             uErrorD[0] = Math.toDegrees(0); // <- No roll control
```

```
741             uErrorD[1] = Math.toDegrees(etaRefR[4] - etaR[4]);
742             uErrorD[2] = angle_180to180(Math.toDegrees(etaRefR[5] - etaR[5]));
743             uErrorR = arrayToRad(uErrorD);
744 
745             // Iterate Hinf
746             etaDotD[0] = Math.toDegrees(etaDotR[3]);
747             etaDotD[1] = Math.toDegrees(etaDotR[4]);
748             etaDotD[2] = Math.toDegrees(etaDotR[5]);
749             double[] cYOutR = arrayToRad(myHinf.iterate(uErrorD, etaDotD));
750 
751             // Mix fin output
752             double[] deltaRM = mixFins(cYOutR, etaR[3]);
753 
754             // Limit control surface flap deflection to 18deg
755             for (int i = 0; i < 2; i++) {
756                 deltaRM[i] = limitFlapDeflection(deltaRM[i], 0.32); // <- 0.32 rad ~= 18.33
                        deg
757                 deltaRM[i] = roundToDec(deltaRM[i], 0.01);
758             }
759 
760             // Assign output (STBD, Port, Lower, Upper)
761             flapTargetR[0] = deltaRM[0];
762             flapTargetR[1] = -deltaRM[0];
763             flapTargetR[2] = deltaRM[1];
764             flapTargetR[3] = -deltaRM[1];
765             flapTargetD = arrayToDeg(flapTargetR);
766 
767             recordOutput(etaR);
768             return flapTargetR;
769         }
770 
771         /**
772          * Provides a PI depth controller.
773          */
774         private class DepthController {
775 
776             private double Kp = -5.0;
777             private double Ki = -0.5;
778             private double dt = 0.1;   // <- must run at 10 Hz
779             private double xC_depth;
780             private PrintWriter stateFileDepth;
781             private String cStateFileName = baseName + "_cStateDepth.csv";
782 
783             private DepthController() throws IOException {
784                 sayL("DepthController object.", 2);
785                 xC_depth = 0;
786                 if (cfdIO.simState.tSim > 0) {
787                     try {
788                         String workingDir = System.getProperty("user.dir");
789                         Scanner cStateScanner = new Scanner(new File(workingDir + File.
                                separator + cStateFileName));
790                         xC_depth = cStateScanner.nextDouble();
791                         sayL("Depth controller state succesfully imported.", 3);
792                     } catch (Exception ex1) {
793                         sayL("Warning: Unable to import depth controller state... Creating
                                new zero state.", 3);
794                     }
795                 } else {
796                     sayL("Simulation time = 0... Creating new zero state.", 3);
797                 }
798 
799                 sayL("xC_depth = " + Double.toString(xC_depth), 3);
800             }
801 
```

```
802             /**
803              * Iterates depth controller to find pitch command.
804              * @param z current z-position (m)
805              * @param zRef z-position reference (m)
806              * @return
807              */
808             private double iterate(double z, double zRef) {
809                 double zError = zRef - z;
810                 double thetaComD;
811 
812                 xC_depth = xC_depth + dt * zError;
813                 saveState();
814 
815                 thetaComD = (Kp * zError + Ki * xC_depth);
816                 return thetaComD; // <- pitch command (deg)
817             }
818 
819             /**
820              * Saves controller state to file.
821              */
822             private void saveState() {
823                 try {
824                     stateFileDepth = new PrintWriter(new BufferedWriter(new FileWriter(
                            cStateFileName)));
825                     stateFileDepth.write(Double.toString(xC_depth));
826                     stateFileDepth.flush();
827                     stateFileDepth.close();
828                 } catch (Exception e) {
829                     sayL("Warning: Unable to output depth controller state to file.");
830                     sayL(e.getMessage());
831                     sayL(e.getLocalizedMessage());
832                 }
833             }
834         }
835 
836         /**
837          * Provides an H-infinity pitch and yaw controller
838          */
839         private class Hinf {
840 
841             private Matrix cA, cB, cC, cD, cu, cx, cy, x0;
842             private int nx, nu, ny; // size of state vector, input vector, output vector
843             private double cf; // controller frequency
844             private double yawIntegratorSaturation = 240;
845             private double pitchIntegratorSaturation = 1000;
846             private PrintWriter stateFileHinf;
847             private String cStateFileName = baseName + "_cStateHinf.csv";
848 
849             private Hinf() throws IOException {
850                 sayL("HinfController object.", 2);
851                 String configFileName = "AUVHinfControllerDJS_13Oct2010_1.txt";
852                 String workingDir = System.getProperty("user.dir");
853                 File configFile = new File(workingDir + File.separator + configFileName);
854                 try {
855                     Scanner fscan = new Scanner(configFile);
856                     nx = fscan.nextInt();
857                     nu = fscan.nextInt();
858                     ny = fscan.nextInt();
859                     cf = fscan.nextDouble();
860                     x0 = new Matrix(nx, 1);
861                     cA = new Matrix(nx, nx);
862                     cB = new Matrix(nx, nu);
863                     cC = new Matrix(ny, nx);
864                     cD = new Matrix(ny, nu);
```

```
865                     cu = new Matrix(nu, 1);
866                     cx = new Matrix(nx, 1);
867                     cy = new Matrix(ny, 1);
868                     fscan.nextLine();
869                     buildMatrixFromLine(x0, fscan);
870                     buildMatrixFromLine(cA, fscan);
871                     buildMatrixFromLine(cB, fscan);
872                     buildMatrixFromLine(cC, fscan);
873                     buildMatrixFromLine(cD, fscan);
874                     fscan.close();
875                 } catch (Exception e) {
876                     throw new IOException("ERROR: Unable to import Hinf controller
                        parameters.");
877                 }
878 
879                 if (cfdIO.simState.tSim > 0) {
880                     try {
881                         Scanner cStateScanner = new Scanner(new File(workingDir + File.
                            separator + cStateFileName));
882                         buildMatrixFromLine(cu, cStateScanner).transpose();
883                         buildMatrixFromLine(cx, cStateScanner).transpose();
884                         sayL("Hinf controller state succesfully imported.", 3);
885                     } catch (Exception e) {
886                         sayL("Warning: Unable to import Hinf controller state... Creating
                            new zero state.", 3);
887                     }
888                 } else {
889                     sayL("Simulation time = 0...  Creating new zero state.", 3);
890                 }
891                 sayL("cu = " + Arrays.toString(cu.getColumnPackedCopy()), 3);
892                 sayL("cx = " + Arrays.toString(cx.getColumnPackedCopy()), 3);
893             }
894 
895             /**
896              * Iterates H-infinity controller to find control surface
897              * deflections.
898              * @param uErrorD phi, theta, psi angle errors (deg)
899              * @param uRateD phi, theta, psi rates (deg/s)
900              * @return
901              */
902             private double[] iterate(double[] uErrorD, double uRateD[]) {
903                 double[] uErrorIntD = new double[3];
904                 double[] cyD = new double[2];
905 
906                 uErrorIntD[0] = 0;
907                 uErrorIntD[1] = saturationLimit(cu.get(2, 0) + uErrorD[1] / cf,
                        pitchIntegratorSaturation);
908                 uErrorIntD[2] = saturationLimit(cu.get(5, 0) + uErrorD[2] / cf,
                        yawIntegratorSaturation);
909 
910                 // Pack the input vector
911                 cu.set(0, 0, -uRateD[1]);       // pitch
912                 cu.set(1, 0, uErrorD[1]);
913                 cu.set(2, 0, uErrorIntD[1]);
914                 cu.set(3, 0, -uRateD[2]);       // yaw
915                 cu.set(4, 0, uErrorD[2]);
916                 cu.set(5, 0, uErrorIntD[2]);
917 
918                 // Salculate state and output vectors
919                 cx = cA.times(cx).plus(cB.times(cu));
920                 cy = cC.times(cx).plus(cD.times(cu));
921 
922                 // Save controller state to file
923                 saveState();
```

```
924
925                 cyD[0] = cy.get(0, 0);
926                 cyD[1] = cy.get(1, 0);
927                 return cyD;
928             }
929
930             /**
931              * Saves controller state to file.
932              */
933             private void saveState() {
934                 try {
935                     stateFileHinf = new PrintWriter(new BufferedWriter(new FileWriter(
                            cStateFileName)));
936                     cu.transpose().print(stateFileHinf, 5, 3);
937                     cx.transpose().print(stateFileHinf, 5, 3);
938                     stateFileHinf.flush();
939                     stateFileHinf.close();
940                 } catch (Exception e) {
941                     sayL("Warning: Unable to output Hinf controller state to file.");
942                     sayL(e.getMessage());
943                     sayL(e.getLocalizedMessage());
944                 }
945             }
946         }
947
948         /**
949          * Finds the increment of time-steps at which the controller should be
950          * allowed to iterate.
951          * @param controllerFreq Operating frequency of the controller (Hz)
952          * @param tStep Simulation time-step size (s)
953          * @return
954          * @throws Exception
955          */
956         public int getIncrement(double controllerFreq, double tStep) throws Exception {
957             // Simulation time step must be set as a multipe of 0.1s
958             int vContInc = (int) (1 / (controllerFreq * tStep));
959             sayL("Controller increment: " + vContInc, 1);
960             if (1 / (controllerFreq * tStep) - Math.floor(1 / (controllerFreq * tStep)) !=
                    0) {
961                 throw new Exception("ERROR: SIMULATION TIME STEP MUST BE A MULTIPLE OF
                        CONTROLLER FREQUENCY");
962             }
963             return vContInc;
964         }
965
966         /**
967          * Records heading reference (command) and error to STAR-CCM+ reports.
968          */
969         public void recordOutput(double[] etaR) {
970             sayL("CONTROLLER OUTPUT", 1);
971             sayL("Depth control [reference, current] (m): [" + refHeadR[2] + ", " + etaR[2]
                    + "]", 2);
972             sayL("       Reference heading  (deg): " + Arrays.toString(arrayToDeg(refHeadR))
                    , 2);
973             sayL("            Heading error (deg): " + Arrays.toString(uErrorD), 2);
974             sayL("Flap delfection commanded (deg): " + Arrays.toString(flapTargetD), 2);
975             for (int j = 0; j < 3; j++) {
976                 repEtaRef[j].setDefinition(Double.toString(refHeadR[j]));
977                 repEtaError[j].setDefinition(Double.toString(uErrorR[j]));
978             }
979             for (int i = 0; i < 4; i++) {
980                 repFlapCommand[i].setDefinition(Double.toString(flapTargetR[i]));
981             }
982         }
```

```
983
984         /**
985          * Limits signals.
986          * @param signal
987          * @param lim
988          * @return Limited signal.
989          */
990         public double saturationLimit(double signal, double lim) {
991             if (signal < -Math.abs(lim)) {
992                 signal = -Math.abs(lim);
993                 say("Control signal saturation.");
994             }
995             if (signal > Math.abs(lim)) {
996                 signal = Math.abs(lim);
997                 say("Control signal saturation.");
998             }
999             return signal;
1000        }
1001
1002        /**
1003         * Creates mixed fin output
1004         * @param yC control surface deflection commands
1005         * @param phi roll angle (deg)
1006         * @return
1007         */
1008        public double[] mixFins(double[] yC, double phi) {
1009            double[] deltaMixed = new double[2];
1010            deltaMixed[0] = yC[0] * Math.cos(phi) + yC[1] * Math.sin(phi);
1011            deltaMixed[1] = -yC[0] * Math.sin(phi) + yC[1] * Math.cos(phi);
1012            return deltaMixed;
1013        }
1014
1015        /**
1016         * Limits rudder deflection command.
1017         * @param ang1R
1018         * @param angLimitR
1019         * @return
1020         */
1021        public double limitFlapDeflection(double ang1R, double angLimitR) {
1022            double ang2R = Math.signum(ang1R)
1023                    * Math.min(angLimitR, Math.abs(ang1R));
1024            return ang2R;
1025        }
1026
1027        /**
1028         * Fixes angle to between -180 and +180.
1029         * @param angD
1030         * @return
1031         */
1032        public double angle_180to180(double angD) {
1033            while (angD < -180) {
1034                angD = angD + 360;
1035            }
1036            while (angD > 180) {
1037                angD = angD - 360;
1038            }
1039            return angD;
1040        }
1041    }
1042
1043    /**
1044     * Logic for pseudo zig-zag maneuver where the control algorithm is used.
1045     * @param centerAngleR heading angle about which maneuver is centered
1046     * @param extAngleR magnitude of zig-zag from centerAngleR
```

```
1047      * @param time current simulation time (s)
1048      * @param freq maneuver frequency (Hz)
1049      * @return heading
1050      */
1051     public double pseudoZigZag(double centerAngleR, double extAngleR, double freq, double
              time) {
1052         double nextRefHeadR;
1053         if (time == 0) {
1054             time = 0.00001;
1055         }
1056         nextRefHeadR = centerAngleR - extAngleR * Math.signum(Math.sin(time * Math.PI * freq
                 ));
1057         return nextRefHeadR;
1058     }
1059 
1060     /**
1061      * Builds a matrix from a Scanner object.
1062      * @param M
1063      * @param fsc
1064      * @return A matrix composed of the values from the scanned line.
1065      */
1066     public static Matrix buildMatrixFromLine(Matrix M, Scanner fsc) {
1067         for (int i = 0; i < M.getRowDimension(); i++) {
1068             for (int j = 0; j < M.getColumnDimension(); j++) {
1069                 M.set(i, j, fsc.nextDouble());
1070             }
1071         }
1072         return M;
1073     }
1074 
1075     public static double[] arrayToRad(double[] arrayD) {
1076         double[] arrayR = new double[arrayD.length];
1077         for (int i = 0; i < arrayD.length; i++) {
1078             arrayR[i] = Math.toRadians(arrayD[i]);
1079         }
1080         return arrayR;
1081     }
1082 
1083     public static double[] arrayToDeg(double[] arrayR) {
1084         double[] arrayD = new double[arrayR.length];
1085         for (int i = 0; i < arrayR.length; i++) {
1086             arrayD[i] = Math.toDegrees(arrayR[i]);
1087         }
1088         return arrayD;
1089     }
1090 
1091     public static double roundToDec(double dIn, double decFactor) {
1092         double dOut = Math.round(dIn / decFactor) * decFactor;
1093         return dOut;
1094     }
1095 
1096     /**
1097      * Gets current date and time.
1098      * @return
1099      */
1100     public static String getDateAndTime() {
1101         DateFormat dateFormat = new SimpleDateFormat("MM_dd_yy_HH_mm_ss");
1102         Date date = new Date();
1103         return dateFormat.format(date);
1104     }
1105 
1106     public void fatalError(StackTraceElement[] eStr) {
1107         sayL("FATAL ERROR WHEN ATTEMPTING TO INITIALIZE MACRO OBJECTS");
1108         for (int i = 0; i < eStr.length; i++) {
```

```
1109            sayL(eStr[i].toString(), 1);
1110        }
1111    }
1112
1113    /**
1114     * Prints message to output window with time, sim file name and boarder
1115     * @param message
1116     */
1117    public void ouputHeader(String message) {
1118        String time = getDateAndTime();
1119
1120        sim.println("*************************************************************");
1121        sim.println("*************************************************************");
1122        sayL(message);
1123        sayL(baseName + " @ " + time);
1124        sim.println("*************************************************************");
1125        sim.println("*************************************************************");
1126    }
1127
1128    /**
1129     * Prints a message to the Star-CCM+ output window indented to a specified
1130     * level.
1131     * @param msg message to be printed
1132     * @param indLevel level of indentation
1133     */
1134    public void sayL(String msg, int... indLevel) {
1135        String indentS;
1136        if (indLevel.length == 0) {
1137            indentS = "";
1138        } else {
1139            indentS = StringUtils.repeat("  ", indLevel[0]);
1140        }
1141        sim.println(indentS + MACROPREFIX + " " + msg);
1142    }
1143
1144    /**
1145     * Prints a message to the Star-CCM+ output window indented to a specified
1146     * level.
1147     * @param msg message to be printed
1148     * @param indLevel level of indentation
1149     */
1150    public void say(String msg, int... indLevel) {
1151        String indentS;
1152        if (indLevel.length == 0) {
1153            indentS = "";
1154        } else {
1155            indentS = StringUtils.repeat("  ", indLevel[0]);
1156        }
1157        sim.print(indentS + MACROPREFIX + " " + msg);
1158    }
1159}
```

Appendix C

Computing Hardware

A number of desk-side, server and cluster computing resources were utilized for computation of CFD simulations. Specifications are given here for the machines that are referenced throughout this dissertation.

C.1 Breaker

Breaker is a desk-side Dell computer with a dual-core 3.16 GHz processor and 8 GB of RAM. It runs 64-bit Windows 7 and was used primarily for graphics processing and running state-space models.

C.2 Cyclone

A "desk-side" BoxCluster NXi system allowed for medium-sized simulations to be run in an interactive manner (with GUI monitoring). This cluster machine is composed of four nodes running Red Hat Linux, connected by Inifiniband Ethernet. Each node contains an Intel Xeon 3.06 GHz quad core processor and 48 GB of RAM.

C.3 Arcturus, Altus and Blackswift

Arcturus and Blackswift are server-style machines (with a larger number of CPUs within a single node operated by the Department of Aerospace and Ocean Engineering at Virginia Tech. These machines run Red Hat Linux and contain four 12-core AMD Opteron processors with 2.2 GHz clock speeds.

C.4 Advanced Research Computing Machines

A number a machines operated by Advanced Research Computing (ARC)[1] at Virginia Tech were utilized in this study.

C.4.1 Ithaca

Ithaca has 66 research-dedicated nodes, each of which has dual-socket quad-core 2.26 GHz Intel Nehalem processors and either 24 GB of memory (higher memory nodes with 48 GB are available for memory-intensive applications.

C.4.2 Blueridge

BlueRidge us an Appro GreenBlade cluster, with 318 nodes, each of which is equipped with two octa-core Intel Sandy Bridge CPUs and 64 GB of memory (this gives a total of 5,088 cores on the cluster as a whole).

[1] `http://www.arc.vt.edu`

Appendix D

CFD Design Analysis

In addition to the goal of developing and applying innovative methods of analyzing vehicle maneuvering performance, this study has also the synchronous design process of the GPAUV.

D.1 GPAUV Drag Analysis

One of the most central drivers in the design on an AUV, especially one tasked with performing long ingresses and egresses during missions, is the drag experienced by the vehicle in its nominal operating condition. When performing long transits, the GPAUV is designed to operate at a forward speed of 2 m/s. This corresponds to a length-based Reynolds number of 4.6×10^6 in freshwater and 3.9×10^6 in seawater. Steady simulations predicted an area-based drag coefficient for this condition of $C_d = 0.217$. A full plot showing the drag coefficient for a range of speeds is shown in Figure D.1.

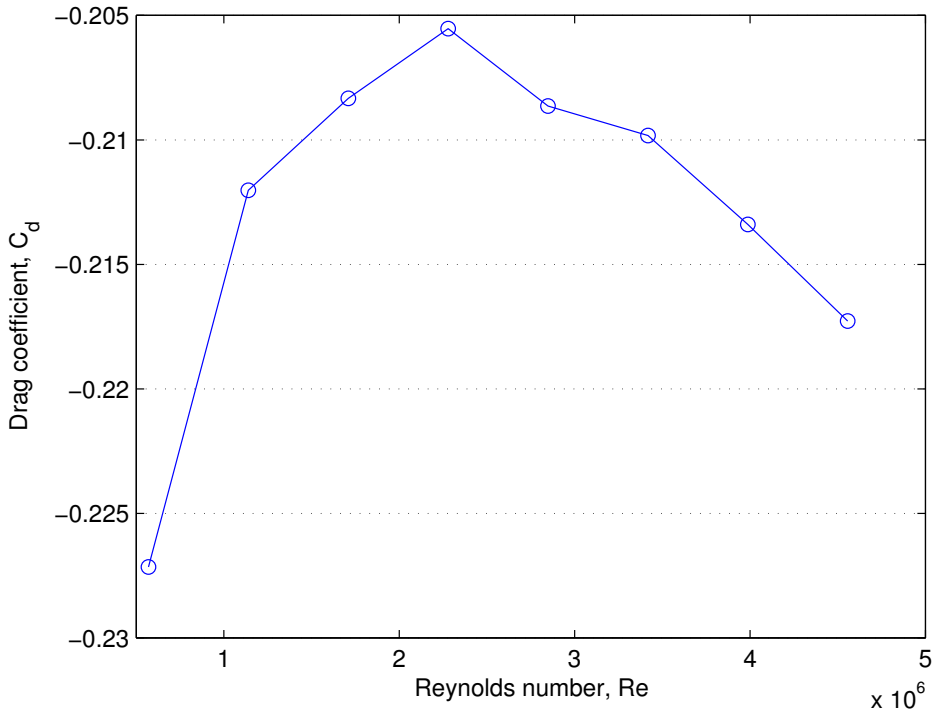
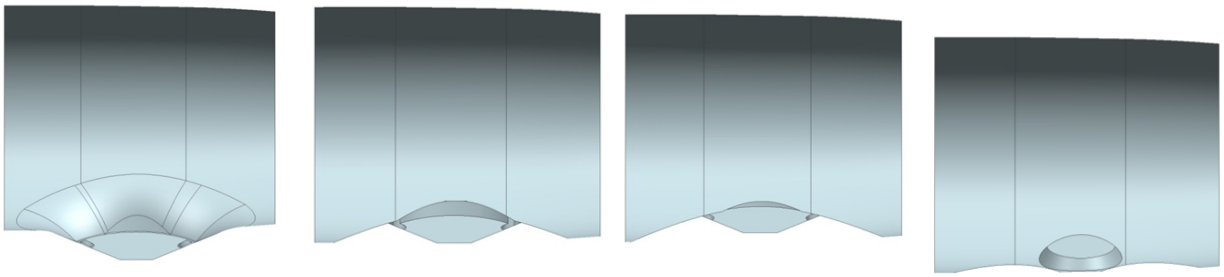


Figure D.1: Predicted drag coefficient for GPAUV at a range of Reynolds numbers.

D.2 Appendage Drag Analysis

Steady-state analyses were conducted on a number of GPAUV design variants. Figure D.2 shows a series of configuration options considered for the GPAUVs digital velocity logger (DVL). The longitudinal mounting location for the DVL, chosen based on mission requirements and internal arrangements, was fixed directly upstream of the sensor fairing (see Figure 1.1).

Simulations were run with geometry to represent each of the mounting options shown in Figure D.2. The additional powering required to propel the vehicle when fitted with each of the options is shown in Figure D.3. While Option 1, with the DVL exposed to the flow, was expected to provide slightly better measurement accuracy, it is also the most obstructive to the flow.



(a) Option 1 **(b)** Option 2 **(c)** Option 3 **(d)** Option 4

Figure D.2: Digital velocity logger (DVL) mounting options for GPAUV.

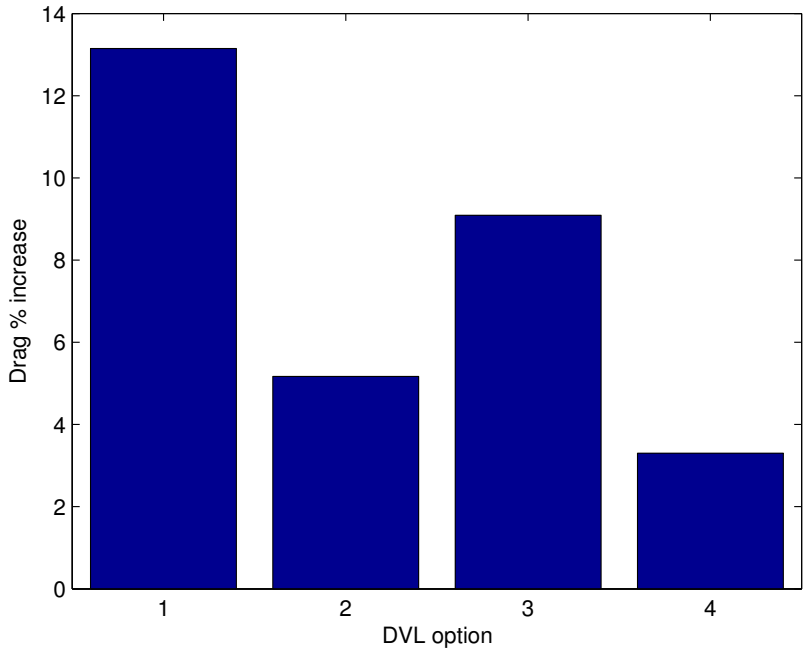


Figure D.3: Increase in powering required for the different DVL mounting options shown in Figure D.2.

D.3 Control Surface Geometry

A fluid-based vehicle experiences drag and lift forces as a result of integrated pressure and shear acting over its surface. The control forces and moments supplied by a deflected control flap come from the change in pressure due to a change in the flow field. This altered pressure distribution acts not only the control surfaces themselves, but on adjacent surfaces. Thus while the control surfaces function as the "drivers" in creating control forces and moments, they are not the only surfaces from which reactions are derived.[1] This is the supporting concept for the fixed-strake and movable-flap control geometry employed on the GPAUV (see Figure 1.2).

D.3.1 Motivation

Figure D.4 shows the velocity vectors and a surface pressure scalar from a simulation of the GPAUV operating with its original upper rudder deflected to 20°. Of particular interest here, is the high rate of flow beneath the rudder, escaping from the pressure side of the control surface to its suction side.

D.3.2 Improved Geometry

Based on this simulation and others like it, a new control surface geometry was proposed in which the control flaps fit more closely to the body of the GPAUV. Figure D.5 shows the original and improved controlled flap geometry.

The improved control flap geometry was shown to supply significantly more control authority, with relatively little impact to vehicle's drag when undeflected. When deflected to 20°, the improved control flap geometry supplied the GPAUV with 13.5% more yaw moment than the original geometry. The effects of this change can be seen by comparing Figure D.6, which shows the GPAUV with this improved control flap geometry, again deflected to 20°, and Figure D.4. The significant flow from the pressure to suction side of the flap is greatly diminished with the improved geometry.

This result is even more evident in Figure D.7, which shows a side-by-side comparison of the pressure acting on the hull of the GPAUV with the rudder deflected to 20°. The improved geometry creates larger pressures on itself and adjacent surfaces, in turn creating larger control forces and moments.

[1] In order to formulate state-space model with the highest possible level of reality, this mechanism must me be taken into account. Control input models that use an "effective angle of attack" do not fully address this issue. For example, the control forces produced when the AUV is flying straight, with zero side-slip, and the rudders deflected to $\delta = 15°$ will be different than the those produced at 15° side-slip and no rudder deflection. The pressure leakage that occurs in these two situations will be different, thus creating different control forces.

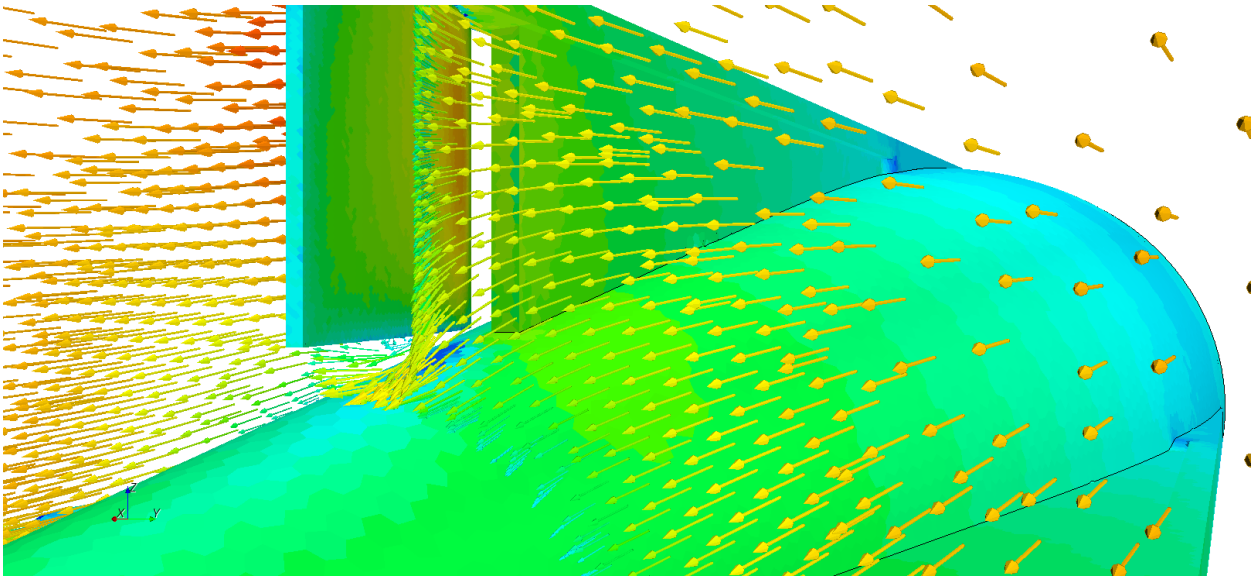


Figure D.4: GPAUV with original control surface deflected to 20° (in out-of-page direction); velocity vectors and surface pressure scalar.

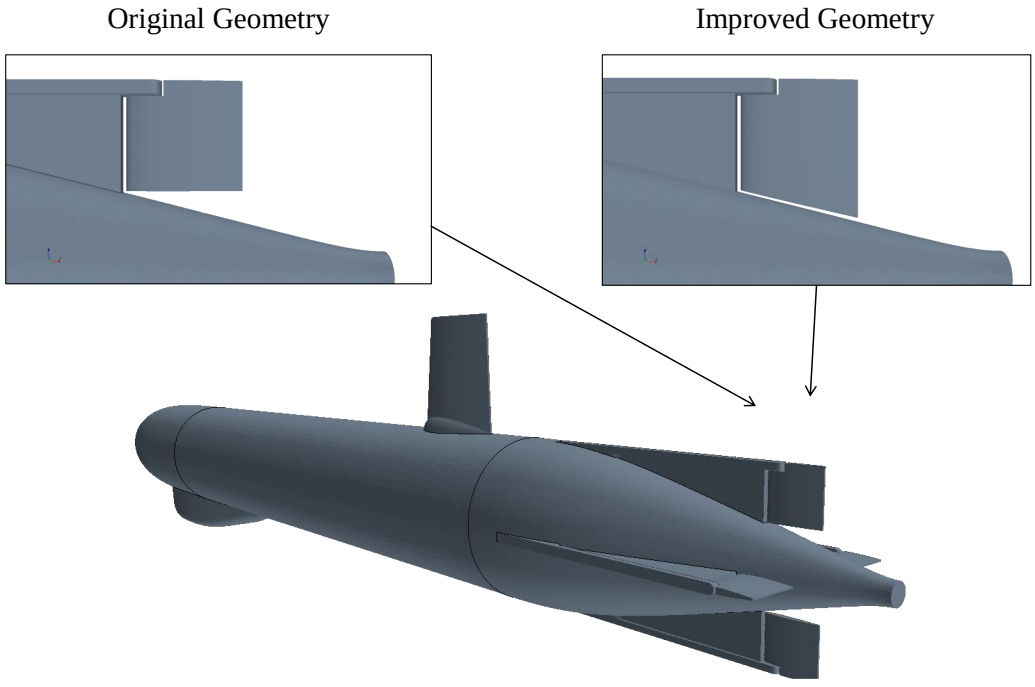


Figure D.5: GPAUV with original and improved control flap geometry.

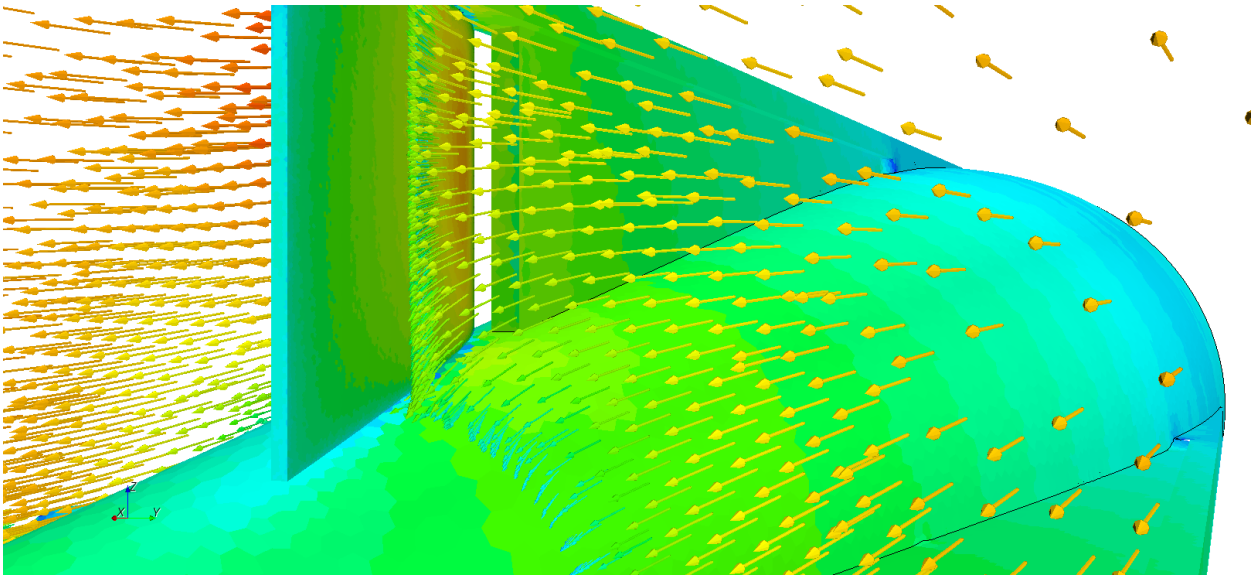
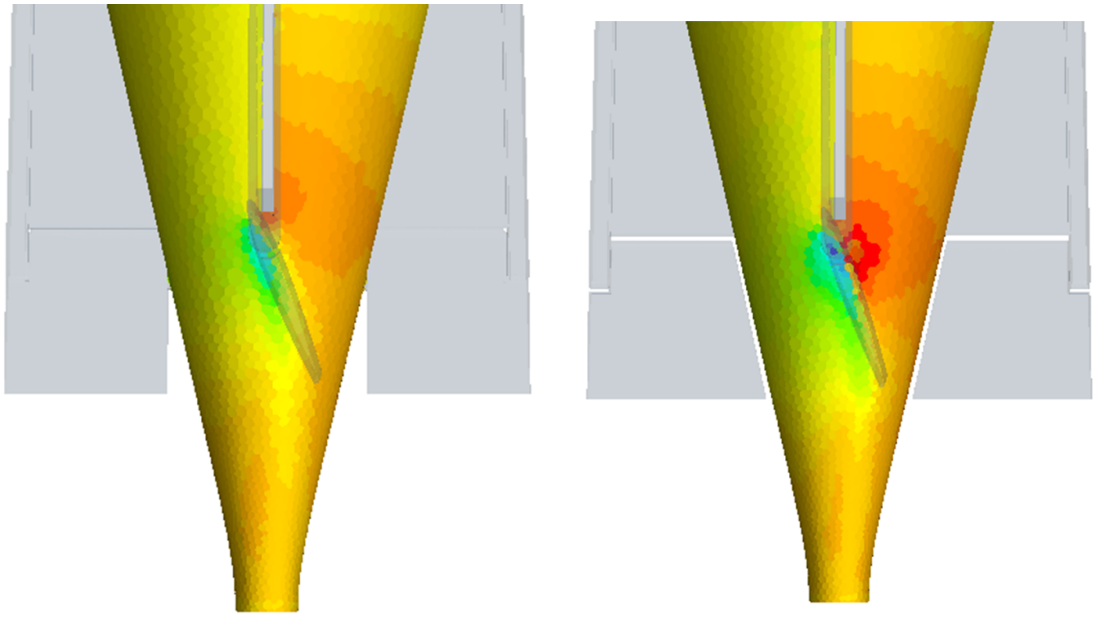


Figure D.6: GPAUV with improved control surface deflected to 20°; velocity vectors and surface pressure scalar.



(a) Old CS geometry **(b)** New CS geometry

Figure D.7: Comparison of surface pressure coefficient on GPAUV hull on different rudder designs deflected to 20°.