NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

AUTONOMOUS CONTROL OF UNDERWATER
VEHICLES
AND LOCAL AREA MANEUVERING

by

David Bryan Marco

September 1996

Dissertation Advisor: Anthony J. Healey

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(leave bank)*	2. REPORT DATE September 1996	3. REPORT TYPE AND DATES COVERED Ph.D. Dissertation

4. TITLE AND SUBTITLE
AUTONOMOUS CONTROL OF UNDERWATER VEHICLES
AND LOCAL AREA MANEUVERING

5. FUNDING NUMBERS

6. AUTHOR(S) David Bryan Marco

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

The major thrust of this work is the development and demonstration of new capabilities for the use of small autonomous vehicles in mine countermeasure applications. Key to the new capabilities lies in an open architecture tri-level software structure for hybrid control, of which this work is the first validated implementation. The two upper levels run asynchronously in computing logical operations based on numerical decision making, while the lowest, the Execution Level, runs synchronously to maintain stability of vehicle motion. The top (Strategic) Level of control uses Prolog as a rule based language for the specification of the discrete event system (DES) aspects of the mission. Multiple servo controllers are coordinated by the middle (Tactical) Level software in performing the mission, while the Execution Level controllers guarantee robust motion stability through multiple sliding modes.

14. SUBJECT TERMS Hybrid Control, Sliding Mode Control, Local Area Maneuvering	15. NUMBER OF PAGES 362
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18298102

CONTINUATION OF BLOCK 13, (ABSTRACT).

This hardware / software arrangement provides the ability to operate a hybrid (mixed discrete state/continuous state) controller for semi-autonomous and autonomous vehicles in which the missions imply multiple task robot behavior. This work has defined and developed a set of vehicle "primitives", that are a set of stable modular control functions unique to a given vehicle's capabilities. It is demonstrated how these can easily be combined using rules to specify as simple, or as complex, a mission as desired. Completion of a mission is guaranteed through a 'complete plan' including time traps and error recovery procedures. Experimental results are given illustrating the performance attained.

A particular case of the techniques developed has resulted in a method to navigate an AUV in a local area (around a mine-like object) using a profiling sonar sensor for position information derived from underwater feature detection. Since sonar image feature extraction is necessarily time consuming, a dynamic model of the vehicle response is used for control between position updates. A structured formulation of this control / navigation method is presented followed by results from in water implementation using the NPS Phoenix vehicle and the tri-level software structure described above.

AUTONOMOUS CONTROL OF UNDERWATER VEHICLES AND LOCAL AREA MANEUVERING

David Bryan Marco
B.S., Oklahoma State University, 1983
M.S., University of Texas at Austin, 1987

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September, 1996

ABSTRACT

The major thrust of this work is the development and demonstration of new capabilities for the use of small autonomous vehicles in mine countermeasure applications. Key to the new capabilities lies in an open architecture tri-level software structure for hybrid control, of which this work is the first validated implementation. The two upper levels run asynchronously in computing logical operations based on numerical decision making, while the lowest, the Execution Level, runs synchronously to maintain stability of vehicle motion. The top (Strategic) Level of control uses Prolog as a rule based language for the specification of the discrete event system (DES) aspects of the mission. Multiple servo controllers are coordinated by the middle (Tactical) Level software in performing the mission, while the Execution Level controllers guarantee robust motion stability through multiple sliding modes.

This hardware / software arrangement provides the ability to operate a hybrid (mixed discrete state/continuous state) controller for semi-autonomous and autonomous vehicles in which the missions imply multiple task robot behavior. This work has defined and developed a set of vehicle "primitives", that are a set of stable modular control functions unique to a given vehicle's capabilities. It is demonstrated how these can easily be combined using rules to specify as simple, or as complex, a mission as desired. Completion of a mission is guaranteed through a "complete plan" including time traps and error recovery procedures. Experimental results are given illustrating the performance attained.

A particular case of the techniques developed has resulted in a method to navigate an AUV in a local area (around a mine-like object) using a profiling sonar sensor for position information derived from underwater feature detection. Since sonar image feature extraction is necessarily time consuming, a dynamic model of the vehicle response is used for control between position updates. A structured formulation of this control / navigation method is presented followed by results from in water implementation using the NPS Phoenix vehicle and the tri-level software structure described above.

TABLE OF CONTENTS

ACKNOWLEDGMENT

I. INTRODUCTION

At the present time, unmanned underwater vehicle (UUV) activities in military and scientific / commercial fields are usually performed by Remotely Operated Vehicles (ROV's). These vehicles are tethered to a surface ship or offshore platform by an umbilical cable which provides power and control signals. They are employed in the offshore oil and gas industries, salvage and recovery, and increasingly, ocean science, as well as in military mine countermeasures. Vehicle motion control is provided by a human operator (pilot) on the surface, who views the underwater environment through a video camera for short range visual feedback. Scanning sonar is often added for longer range information.

When deep water applications or large horizontal movements of a vehicle are necessary, the tether becomes an ever increasing liability. It adds tensile loading on the vehicle; often uncertain and time varying, and requires elaborate tether management equipment. It is this shortcoming, and the associated costs of the support ship, that has led to the development of Autonomous Underwater Vehicles (AUV's). An AUV is independent of a tether for power and control, and is free to maneuver more easily over larger distances or depths, with little or no direct human supervision. This type of vehicle is well suited for performing expensive and monotonous tasks such as ocean water quality, bathymetric, and geological survey. AUV's could also be utilized for harbor and underwater structural inspection tasks, and most importantly, mine countermeasures and neutralization, where there is a potential for loss of life.

Freedom from the tether is not without cost however, since the ability to satisfactorily perform a mission requires a more complex control system. Even though underwater communication with the vehicle is possible using acoustics, the transmission rates and distances are limited. Real time video feedback to the pilot cannot be accomplished. As opposed to control from the surface ship, AUV's have the need for increased automated functionality onboard not only for self navigation but also for "intelligent" tasks as in fault monitoring, control system reconfiguration, and environmental interpretation.

To implement such a highly automated system, multiple cooperating onboard computers or at least a single computer using multiple processes are used. One problem in designing and understanding the behavior of such a system is that a meaningful mission for an AUV can not be described or modeled using conventional feedback control theory

(Friedland, 1986). While this is an integral component of AUV control, a much higher level of coordination is required. A typical mission would consist of various phases, such as launch, transit, search, recovery, etc., and a method of sequencing these actions must be in place. Since the use of a single process/computer to perform all of these tasks would be extremely difficult, mutiprocess / mutiprocessor configurations are preferred. It simplifies combining the low level, servo control tasks, which must run in real time for stability considerations, with high level algorithms designed to manage the mission objectives. It eases programming difficulties, even for the simplest mission modification, or software changes required for vehicle enhancements. Since sequencing mission phases is inherently an asynchronous process, and operates with longer time scales than for the synchronous servo controllers, there is no need to run them on the same processor. The natural division of synchronous and asynchronous tasks maps well into the separation of symbolic and numeric operations and is well suited to multiple computers. Each system can be chosen based on their particular attributes with regard to computer languages, operating systems, timing constraints, and processor speed.

Since the vehicle motion exists in a physical world, it can be modeled theoretically by a set of differential equations as is well known in describing the behavior and control of Dynamic Systems (DCS). On the other hand, the mission sequencing lies in the domain of Discrete Event Systems (DES). Manufacturing literature is replete with studies of discrete event system control, and (Cassandras, 1993) gives a comprehensive overview, but few discuss the integration of DES and DCS. These combined systems are usually referred to as "Hybrid" control systems, and will be addressed in detail, as part of this work.

Early work on this subject was done by (Saridis, 1983), describing a hierarchical three level controller consisting of the organization, coordination, and hardware levels. The three levels act together using "cognitive and control systems methodologies" with ".. control intelligence .. distributed according to the principle of *increasing intelligence with decreasing precision*, evident in all hierarchical management systems". However, this was developed in the context of industrial robot manipulators rather than underwater vehicles.

As applied to underwater vehicles, the three levels have been named in (Byrnes, et. al., 1993, Byrnes, et. al.,1996) as the Strategic, Tactical, and Execution levels. The Strategic Level is a discrete event system managing the progress of the mission, while the Tactical Level coordinates the control modes required, and the Execution Level performs the motion control of the vehicle. Byrnes work included a thorough study of both backward and forward chaining versions of the "Rational Behavior Model" but was

2

restricted to workstation simulations, not implemented in an actual vehicle operating in the underwater environment.

Other approaches to robot control include the use of subsumption and layered control (Brooks, 1986). In principle, the concept is simple and eases the development of robot hardware. However, an added coordination layer was found to be necessary for AUV's when specific mission objectives have to be met (Bellingham and Consi, 1991). Their work has been applied to open water flight control for oceanographic survey operations, where only limited robot actions are involved.

Tri-level control of underwater robots has been studied in France and Portugal. The effort in France has focused on the development of a computer-aided design system (ORCCAD) (Simon, et. al., 1993) using Esterel - a synchronous language - that encodes the mission as a finite state automaton. Also, the PIRAT system is a computer-aided design software package for the development, implementation, and verification of servo level control laws. This work has been applied to the Vortex vehicle (Perrier, et. al., 1993), although the interfacing and integration of these packages with other vehicles could be difficult.

In Portugal, Petri net methodologies at the Strategic Level and gain scheduled control at the Execution Level are being employed for the MARIUS vehicle. As part of this activity, yet another language - CORAL - is being developed for the real time execution of Petri net mission controllers, although, in this case, the cross compiling into C code for running on general platforms and interfacing with other vehicles is a promising development.

In spite of these recent research efforts, no unified design methodology for robot controllers exists. Implementations on operational underwater vehicles are few and limited in scope. In fact, it has not been shown that any one robot control system is uniquely superior to another, although some systems appear to be easier to reconfigure than others. Therefore, the main purpose of this work is to develop, demonstrate, and validate an open architecture, three level control system for an AUV. Using commercially available languages, this would allow ease in reconfiguration of control modes, and in particular, the examination of robot performance in the context of hovering and local area maneuvering around a target. The importance of this work is the use of an existing AI computer language known as Prolog, as opposed to a special purpose language such as Esterel. Prolog is convenient for Strategic Level sequencing of the mission phases, and allows the use of the

3

full powers of the Prolog inference engine. It also provides a natural link between symbolic operations and the C language numerical computations in the Execution Level.

These control concepts have been validated on a small underwater vehicle (6 ft in length), which possesses both fins, cross-body thrusters, and rear propulsors for motion control. In the majority of this work, the thrusters and rear propulsors are used for movement in a small area. The vehicle is able to maneuver in many different directions and if needed, backwards. With this type of general motion capability, there is no fixed speed and direction of motion, and since the motion is limited, the vehicle rarely reaches a steady state condition. The vehicle is subject to highly non-linear effects from quadratic lift and drag forces, (Fossen, 1991, Yuh, 1990, Sarpkaya and Issacson, 1981) so that linear modeling and control techniques are not suitable. One approach in nonlinear control is to linearize about some nominal operating point. When operating points change slowly, gain scheduling techniques are often employed (Healey, et. al., 1995). In fact, linearization about a constant forward speed is the usual technique used for submarine and torpedo control design (Milliken, 1984, Lindgren, et al., 1967). However, in the cases studied in this work, slowly changing operating points do not exist, and linearization about such points is not possible. For these systems, non-linear techniques are required to ensure high performance control and stability.

For the Execution Level, application of the sliding mode methodology (SMC) is used throughout this work. It can deal with non-linear dynamics directly and is effective against parametric uncertainties and unmodeled disturbances. A tutorial covering general concepts is given in (Decarlo, et. al., 1988). Simplified applications to ROV's and AUV's are given in (Yoerger and Slotine, 1985), (Yoerger, et. al., 1986). A multivariable sliding mode autopilot for AUV's using decoupled modeling for speed, steering and diving was described by (Healey and Lienard,1993). Dynamic positioning of ROV's using sliding modes is described in (Fossen, 1991) and (Fossen and Sagatun, 1991).

The objectives of this work separate into five distinct parts.

1. Chapter II covers the general theory of vehicle control using Sliding Modes. A 6 DOF mathematical model of the vehicle and controller is given together with 3-D command generators for precise velocity and positioning maneuvers. Also included is a robustness analysis of the controller design validated by computer simulation results.

4

2. Robot control of the NPS Phoenix vehicle is discussed in Chapter III, starting with an overview of robot control structures in general from various other efforts throughout the world. Details of the computer hardware and software systems used for mission control are provided. Results from in-water experiments show the effectiveness of this control method, and a short critique of the control architecture is presented at the end.

3. In Chapter IV, an in-depth discussion and description of the acoustic sensors used by the vehicle for navigation and environmental assessment is given.

4. Chapter V gives in-water experimental results and performance evaluations of sliding mode control for the Phoenix. Results of submergence, rotation, and longitudinal motion control experiments are presented which include Kalman filters necessary to the successful implementation of the respective control designs. Results of coordinated maneuvers using combinations of submergence and rotation control are presented.

5. Local area maneuvering of the Phoenix with sonar is outlined in Chapter VI. This chapter contains a description of experiments conducted with sonar control algorithms for target recognition, relative range and bearing calculations. Two algorithms have been applied to this problem together with simulation and complete verification for each. In-water results using one of the algorithms are presented together with a performance evaluation.

A summary of the dissertation is given in Chapter VII, which contains concluding comments, remaining issues, and recommendations for future work. An extensive section of appendices is presented at the end of this work to provide the reader with implementation details and software that was not deemed appropriate for the body of the text.

II. GENERAL THEORY OF AUV MOTION CONTROL

This chapter discusses the foundation and design of a MIMO (Multiple Input/Multiple Output) Sliding Mode controller for an (AUV) in performing precision tracking to command signals. Precision control is needed for maneuvers such as automatic docking, recovery, and submerged object inspection / identification. Whenever an underwater vehicle maneuvers in the surrounding water column, it is inevitable that reaction forces, caused by time varying changes to the pressure distribution around the vehicle body, are generated. These forces arise from hydrostatic buoyancy as well as from the relative velocity and acceleration of motion. Hydrodynamic forces and moments are usually expressed in terms of lift, drag, and added mass components modeled with assumed constant coefficients. Since the hydrodynamic coefficients are only estimates, often based on poor representations of reality at best, large parameter uncertainty exists.

The system to be controlled is highly nonlinear and coupled. While tracking control could be implemented using a variety of techniques that have been proposed recently, it has been confirmed during the course of this research that control using sliding modes results in high gain robust controllers that are easily tuned and modified. This particular attribute is beneficial because it leads to the notion that control law adaptivity could be readily accomplished.

In this chapter the NPS Phoenix AUV is used as the basis for the development of the control theory and is assumed to have 8 independently controllable fins, two stern rudders, two stern planes, two bow rudders, and two bow planes. Propulsion is provided by four cross body thrusters, two lateral and two vertical, with two rear screws. The computer simulation model so developed is based on non-linear equations of motion for this vehicle. Section A describes the vehicle model with a controller design using sliding modes. A robustness analysis of the control design is given in Section B. A design methodology for robustness is presented which accounts for parametric discrepancies between the dynamic model used for controller design and the actual vehicle. Section C presents the specific sliding mode control implementation for the NPS Phoenix vehicle. Discussed in Section D is control allocation among the various actuators available to the vehicle. Since some vehicles, including the Phoenix, are "over-actuated", methods to allocate the control effort are of primary importance and several suggestions to handle this are presented. Two computer simulations of the performance of the control design are

presented in Section E. One case uses all available actuators (thrusters, fins, screws), while the second uses only fins and the rear screws for control. The simulation uses the complete set of equations of motion, although the particular cases presented here are restricted to the horizontal plane. Command generators for position, velocity, and acceleration as a function of time along a specified trajectory are developed and discussed in the performance of precision tracking control. The final section gives conclusions and recommendations for future work.

A. SLIDING MODE CONTROL AND MODEL BASED CONTROL LAW DESIGN

This section outlines a general theory and the design of a MIMO Sliding Mode Controller for an underwater vehicle. This method provides robust control of systems with nonlinear dynamics, parameter uncertainties, and disturbances. The main advantage of this approach is the use of a switching term in the control law which drives the plant's state trajectory onto a specified surface, known as the sliding surface. "Chattering" of the control input caused by high gain in the switching term, is reduced by the introduction of a boundary layer around the sliding surface. The effect of this boundary layer, is to lower the feedback gain when small errors exist, resulting in smooth control. This technique is particularly useful when feedback signals are noisy. Stability is analyzed using standard Lyapunov methodology.

1. Vehicle Mathematical Modeling

The mathematical model of a system describes it's dynamic behavior, and is usually embodied in a set of differential equations. It forms the basis by which stability of feedback control laws can be assessed. It is also critical to the design of predictive and other more advanced model based controllers, of which, those using sliding modes, form one example. It follows that a mathematical model of an AUV must be established if advanced model based methods are to be developed. At that point, an array of analytical and computer tools can be used for analysis and synthesis purposes.

An underwater vehicle can be assumed to be a rigid body, capable of adopting any position and orientation in the water column. However, in contrast to high speed aircraft and spacecraft, significant departures from a predominantly horizontal attitude are

uncommon. Three independent position and three Euler angles are required for describing the location and attitude of the vehicle. For convenience, three coordinate frames are used. One, denoted as the body-fixed frame (O), is rigidly fixed to and rotates with the vehicle, the second, is a global, earth-fixed frame (G) taken as the inertial reference, and used for local area navigation. A third reference frame (C), parallel to the inertial reference, moves with the local fluid particles at a constant velocity, u_c, (the local current) where

$$u_c = \begin{pmatrix} u_{c_X} & u_{c_Y} & u_{c_Z} & 0 & 0 & 0 \end{pmatrix}^T.$$

Rotational currents and fluid accelerations are assumed to be zero in this work. The three reference frames and their relation to each other are shown in Figure 2.1 which show the body-fixed frame in an arbitrary location and the centers of mass and buoyancy described by the position vectors r_G and r_B respectively. The vehicle motion, expressed in the body-fixed frame, is defined in terms of the six components of the velocity vector $x(t)$, where

$$x(t) = \begin{pmatrix} u(t) & v(t) & w(t) & p(t) & q(t) & r(t) \end{pmatrix}^T. \tag{2.1}$$

A distinction is made in this definition between absolute velocity of the rigid-body and it's velocity relative to the surrounding water column. Typically, the vector $x(t)$ is viewed to be the absolute velocity of a rigid-body expressed in the rotating, body-fixed frame as described by (Fossen, 1991). However, this formulation introduces difficulties if a fluid current is present, since the hydrodynamic lift and drag terms are functions of the velocity of the fluid particles relative to the vehicle, rather than it's absolute velocity. The relative velocity formulation overcomes this difficulty. With this definition, the principle of relative velocities can be used to obtain the absolute velocity of the body given the three reference frames. The position of the vehicle in the global reference frame is given by

$$z(t) = \begin{pmatrix} X(t) & Y(t) & Z(t) & \phi(t) & \theta(t) & \psi(t) \end{pmatrix}^T, \tag{2.2}$$

An Euler angle transformation (the definition and use of Euler angle transformations is well known and described in standard texts on the dynamics of rigid-bodies) is used to relate the vehicle relative velocity components, $x(t)$, to the rate of change of global position, $\dot{z}(t)$. The global velocity of the vehicle can now be expressed as

$$\dot{z}(t) = h(z(t))x(t) + u_c(t), \tag{2.3}$$

where $h(z(t))$ is the 6×6 transformation matrix as a function of the Euler angles $\phi(t)$, $\theta(t)$, $\psi(t)$, (spin, elevation, and azimuth respectively), given by

$$h(z(t)) = \begin{bmatrix} T_t(z(t)) & 0 \\ 0 & T_r(z(t)) \end{bmatrix},$$

where

$$T_t(z(t)) = T(\psi)^T T(\theta)^T T(\phi)^T,$$

and

$$T(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \; ; \; T(\theta) = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \; ; \; T(\psi) = \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The matrix transformations for the translational, $T_t(z(t))$, and rotational, $T_r(z(t))$, velocity components can also be expressed as

$$T_t(z(t)) = \begin{bmatrix} cos\psi cos\theta & cos\psi sin\theta sin\phi - sin\psi cos\phi & cos\psi sin\theta cos\phi + sin\psi sin\phi \\ sin\psi cos\theta & sin\psi sin\theta sin\phi + cos\psi cos\phi & sin\psi sin\theta cos\phi - cos\psi sin\phi \\ -sin\theta & cos\theta sin\phi & cos\theta cos\phi \end{bmatrix},$$

and

$$T_r(z(t)) = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi / cos\theta & cos\phi / cos\theta \end{bmatrix}.$$

Note that $h(z(t))$ is not an orthogonal matrix, and $h(z(t))^{-1} \neq h(z(t))^T$.

With the coordinate frames defined, Newton's second law of motion may be used to formulate a dynamic model of the system. The dynamics of the AUV can be described by a set of six non-linear, coupled, second order differential equations with constant coefficients. For a submerged rigid-body, the equations of motion formulated in a body-fixed reference frame with an arbitrary origin and constant mass and inertia is given by

10

$$m\left(\dot{v}_o + \omega_o \times v_o + \dot{\omega}_o \times r_G + \omega_o \times (\omega_o \times r_G)\right) = \mathbf{F}_o$$

$$I_o\dot{\omega}_o + \omega_o \times (I_o\omega_o) + mr_G \times \dot{v}_o + mr_G \times (\omega_o \times v_o) = \mathbf{M}_o$$

where the first vector equation represents the translational motion, while the second describes the rotational motion. v_o, ω_o, and r_G are defined by

$$v_o = (u \ v \ w)^T$$

$$\omega_o = (p \ q \ r)^T$$

$$r_G = (x_G \ y_G \ z_G)^T ,$$

m is the vehicle mass, and the inertia tensor with respect to the body-fixed reference is

$$I_o = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}.$$

$\mathbf{F}_o$ and $\mathbf{M}_o$ represent the forces and moments respectively, derived from gravitational effects of weight and buoyancy, hydrodynamic lift and drag terms, which are functions of a set of hydrodynamic coefficients, and the relative velocities defined by $x(t)$. These terms also include the effects of hydrodynamic added mass, disturbance forces/moments, and any control inputs from thrusters and control surfaces, and each element can be described by

$$\mathbf{F}_o = (X_o \ Y_o \ Z_o)^T$$

$$\mathbf{M}_o = (K_o \ M_o \ N_o)^T ,$$

where X_o, Y_o, and Z_o are the surge, sway and heave forces, and K_o, M_o, N_o are the roll, pitch, and yaw moments.

In matrix form, the dynamics model may be represented as

$$M\dot{x}(t) = f(x(t),z(t),c) + g(x(t),z(t))u(t), \tag{2.4}$$

11

where M is the 6×6 mass matrix represented by

$$
M = \begin{bmatrix}
m - X_{\dot{u}} & 0 & 0 & 0 & mz_G & -my_G \\
0 & m - Y_{\dot{v}} & 0 & -(mz_G + Y_{\dot{p}}) & 0 & mx_G - Y_{\dot{r}} \\
0 & 0 & m - Z_{\dot{w}} & my_G & -(mx_G + Z_{\dot{q}}) & 0 \\
0 & -(mz_G + K_{\dot{v}}) & my_G & I_{xx} - K_{\dot{p}} & -I_{xy} & -(I_{xz} + K_{\dot{r}}) \\
mz_G & 0 & -(mx_G + M_{\dot{w}}) & -I_{xy} & I_{yy} - M_{\dot{q}} & -I_{yz} \\
-my_G & mx_G - N_{\dot{v}} & 0 & -(I_{xz} + N_{\dot{p}}) & -I_{yz} & I_{zz} - N_{\dot{r}}
\end{bmatrix}
$$

which includes the hydrodynamic added mass and off diagonal terms. The vector $f(x(t), z(t), c)$ is the 6×1 set of forces/moments from centrifugal and Coriolis effects, gravitation and hydrodynamic lift and drag terms, which are functions of a set of hydrodynamic coefficients, c, as described earlier. $u(t)$ is the input control vector with dimension $6 \times m$, of the form

$$
u(t) = \left(u_1(t) \; u_2(t) \; u_3(t) \cdots u_m(t) \right)^T,
$$

where m is the number of control inputs and is determined by the specific vehicle design. The input gain matrix, $g(x(t), z(t))$, contains coefficients that describe the effectiveness of each control input on the vehicle motion, and is speed dependent if control surfaces are used. For more detail refer to (Yuh, 1990, Fossen, 1991, Healey, 1993), and for specifics of the NPS Phoenix design, see Appendix A.

2. Sliding Mode Control Design

Now that the dynamic model of the vehicle has been formulated, the sliding mode control law can be designed. With the exception of cruising vehicles in flight control modes (where the main control modes are for vehicle speed, heading, and depth), it is usually desired to control the motion of an underwater vehicle relative to an inertial reference, not relative to the water column. Therefore, the appropriate error vector is defined in terms of global coordinates as

$$\begin{bmatrix} \dot{\tilde{z}}(t) \\ \tilde{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{z}_{com}(t) \\ z_{com}(t) \end{bmatrix} - \begin{bmatrix} \dot{z}(t) \\ z(t) \end{bmatrix} \qquad \in \ \Re^{12 \times 1}. \qquad (2.5)$$

The subscript *com* refers to the commanded value of the position or velocity in question, where commanded time variations of states must be consistent with vehicle physical capabilities and usually come from a separate path planning algorithm called a "command generator". Position, velocity, and acceleration profiles are ideally kinematically consistent, and continuous with the possible exception of acceleration. One method to generate position, velocity, and acceleration profiles is presented in detail in Appendix B.

Since Eqn. (2.1) in terms relative velocities, and Eqn. (2.5) consists of the global quantities to be tracked, modification of the dynamics equation, (2.4), is needed. By differentiating Eqn. (2.3), an expression for $\dot{x}(t)$ in terms of $\dot{z}(t)$, $\ddot{z}(t)$, and the current u_c is given by

$$\dot{x}(t) = h(z(t))^{-1}\ddot{z}(t) - h(z(t))^{-1}\dot{h}(z(t))h(z(t))^{-1}\big(\dot{z}(t) - u_c\big). \qquad (2.6)$$

Substituting Eqn. (2.6) into (2.4), rearranging and dropping the "function of" notation for clarity, the dynamic equations of motion are now compatible with the definition of the tracking errors

$$\ddot{z}(t) = hM^{-1}\big(f + gu(t)\big) + \dot{h}h^{-1}\big(\dot{z}(t) - u_c\big). \qquad (2.7)$$

The objective of a controller is to force the tracking error to zero as time increases. Using sliding mode methods (DeCarlo, et. al., 1988, Yoeger and Slotine, 1985), a set of sliding surfaces, $\sigma(\tilde{z}(t))$, is defined as

$$\sigma(\tilde{z}(t)) = \begin{bmatrix} S_1 & S_2 & S_3 \end{bmatrix} \begin{bmatrix} \dot{\tilde{z}}(t) \\ \tilde{z}(t) \\ \int \tilde{z}(t)dt \end{bmatrix}, \qquad (2.8)$$

where

$$\sigma(\tilde{z}(t)) \in \Re^{m \times 1}; \ S_1, S_2, S_3 \in \Re^{m \times n},$$

13

and for an underwater vehicle with 6 degrees of freedom, $n = 6$. If $m < n$, the system is under actuated and only a subset of the system modes will be controllable. The reader is referred to (DeCarlo, et. al., 1988) for a discussion on switching surfaces for a system with n states and m actuators.

In spite of the fact that normally there are the same number of sliding surfaces as control inputs, for the m inputs defined, there are only 6 independent combinations corresponding to the six independent degrees-of-freedom of the vehicle. It follows that six sliding surfaces are required for the description of six independent sliding modes rather than ten.

An integral term is included to remove any steady-state position errors that may arise from discrepancies between the assumed current magnitudes and the actual, as well as the presence of unknown disturbances that are not modeled. The elements of S_1, S_2, and S_3 can be selected to provide the desired performance of the closed loop system. Stable tracking behavior is achieved, if the condition:

$$\lim_{t \to \infty} \dot{\sigma}(\tilde{z}(t)) \to 0,$$

together with

$$\lim_{t \to \infty} \sigma(\tilde{z}(t)) \to 0,$$

will also imply

$$\tilde{z}(t) \to 0 \text{ as } t \to \infty,$$

so that the elements of S_1, S_2, and S_3 are chosen to provide stable polynomial functions of state errors. Also, if $S_1 = I$, there is no loss of generality. Conditions under which $\sigma(\tilde{z}(t))$ is always decreasing can be established using Lyapunov theory by defining $V(\sigma)$ as

$$V(\sigma) = \frac{1}{2}\sigma^T(\tilde{z}(t)) * \sigma(\tilde{z}(t)) > 0 \quad \forall \ t > 0, \tag{2.9}$$

such that $V(0) = 0$ and is always increasing with $\sigma(\tilde{z}(t))$, while

$$\dot{V}(t) = \dot{\sigma}^T(\tilde{z}(t)) * \sigma(\tilde{z}(t)) < 0 \quad \forall \ t > 0. \tag{2.10}$$

14

Global asymptotic stability is then guaranteed if $V(t)$ is positive definite while $\dot{V}(t)$ is negative definite. The quadratic nature of (2.9) assures the positive definiteness of $V(t)$, while negative definiteness of $\dot{V}(t)$ may be obtained by a criterion satisfied by

$$\dot{\sigma}_i^T(\tilde{z}(t)) = -\eta_i sgn(\sigma_i(\tilde{z}(t))) \quad i = 1...6, \tag{2.11}$$

where each η_i is a positive scalar. The positive definiteness of $V(t)$ and the negative definiteness of $\dot{V}(t)$, implies that given any initial condition, $\sigma(\tilde{z}(0))$, $\sigma(\tilde{z}(t))$ will remain bounded such that $V(\sigma(\tilde{z}(t))) \leq V(\sigma(\tilde{z}(0)))$.

Since $sgn(\sigma_i(\tilde{z}(t)))$ is discontinuous across $\sigma(\tilde{z}(t)) = 0$, undesirable switch chattering can occur, and can be alleviated by the use of a "boundary layer" around $\sigma(\tilde{z}(t)) = 0$. Therefore, instead of using a *sgn* function, a continuous one could be used such that

$$sat(\sigma_i(\tilde{z}(t)) / \phi_i) = \begin{cases} sgn(\sigma_i(\tilde{z}(t))) & if \; |\sigma_i(\tilde{z}(t))| > \phi_i \\ \\ \sigma_i(\tilde{z}(t)) / \phi_i & otherwise \; . \end{cases}$$

Another approach is to simply use the continuous function $tanh(\sigma(\tilde{z}(t)))$. All three functions are shown in Figure 2.2. Substituting the definition of *sat* into Eqn. (2.11) and noting Eqn. (2.8), it can be written in a more compact form as

$$\dot{\sigma}(\tilde{z}(t)) = \ddot{\tilde{z}}(t) + S_2\dot{\tilde{z}}(t) + S_3\tilde{z}(t) = -F(\sigma(\tilde{z}(t)), \phi), \tag{2.12}$$

where

$$F(\sigma(\tilde{z}(t)), \phi) = \begin{cases} \eta_1 sat(\sigma_1(\tilde{z}(t)) / \phi_1) \\ \\ \eta_2 sat(\sigma_2(\tilde{z}(t)) / \phi_2) \\ \\ \cdot \\ \\ \cdot \\ \\ \eta_6 sat(\sigma_6(\tilde{z}(t)) / \phi_6) \end{cases}.$$

Substituting the dynamics equation, (2.7), into the definition of the sliding surface (2.12), and again dropping the "function of" notation yields

$$\ddot{z}_{com}(t) - hM^{-1}f - hM^{-1}gu(t) - \dot{h}h^{-1}\left(\dot{z}(t) - u_c\right)$$
$$+ S_2\dot{\tilde{z}}(t) + S_3\tilde{z}(t) = -F(\sigma, \phi) ,$$

and after rearranging,

$$gu(t) =$$

$$Mh^{-1}\ddot{z}_{com}(t) - f - Mh^{-1}\dot{h}h^{-1}\left(\dot{z}(t) - u_c\right) + Mh^{-1}S_2\dot{\tilde{z}}(t)$$
$$+ Mh^{-1}S_3\tilde{z}(t) + Mh^{-1}F(\sigma, \phi) .$$

Since the matrices M, f, g, h, and u_c are uncertain in general, the control solution, $u(t)$, is formulated using their estimates, denoted as $\hat{M}$, $\hat{f}$, $\hat{g}$, $\hat{h}$, and $\hat{u}_c$. The control vector can be split in three parts

$$u(t) = u_1(t) + u_2(t) + u_3(t),$$

where

$$u_1(t) = \hat{G}\left(\hat{M}h^{-1}\ddot{z}_{com}(t) - \hat{f}\right) \tag{2.13}$$

contains the feed-forward compensation for acceleration requirements,

$$u_2(t) = \hat{G}\hat{M}\hat{h}^{-1}\left(\dot{\hat{h}}\hat{h}^{-1}(\hat{u}_c - \dot{z}) + S_2\dot{\tilde{z}}(t)\right) \tag{2.14}$$

contains feed-forward and feedback compensation for velocity, and finally

$$u_3(t) = \hat{G}\hat{M}\hat{h}^{-1}\left(S_3\tilde{z} + F(\sigma,\phi)\right) \tag{2.15}$$

16

contains feed-forward and feedback compensation for the position errors including the switching term based on sliding surface values. $\hat{G}$ is defined as the pseudo-inverse of $\hat{g}$ and in the case of an over-actuated vehicle, is an $m \times 6$ matrix, where $m > 6$, an optimization criterion is needed for uniqueness (See Appendix C).

In order to implement the control solution defined above, both h^{-1} and $\hat{G}$ must exist. The condition for existence of h^{-1} is that the elevation angle, θ, does not become $\pm 90°$, and for an underwater vehicle, this is unlikely if it is properly designed and controlled. As for the existence of $\hat{G}$, this depends on the actuator arrangement of the particular vehicle of interest. If certain modes are not directly controllable, $\hat{G}$ is rank deficient and modifications to the control solution will be required, and is the topic of Section C.

3. A Comment on Full State Feedback

The analysis presented requires full state feedback, and in many cases this is not unreasonable. Current inertial navigation systems provide all six rotational states. Doppler sonar gives both speed over the water as well as ground speed, and Long Baseline acoustic positioning systems provide X and Y, while pressure sensors give a measurement of Z. The use of extended Kalman filters for navigation can provide all twelve states.

B. ROBUSTNESS ANALYSIS AND ASSESSMENT OF UNCERTAINTY

This section presents a robustness analysis of the sliding mode controller algorithms previously developed. Accurate modeling of "real" systems can be very difficult especially underwater vehicles. Determination of the mass properties, hydrodynamic coefficients and the control gains for the various actuators can be very time consuming and in some cases impractical. It is a relatively simple task to exactly measure the system's "dry" mass parameters but more difficult to determine the added mass coefficients. The force characteristics of thrusters can be accurately measured as a separate unit but when incorporated into the vehicle, it's behavior becomes dependent on vehicle motion, orientation, and any currents that may be present. Even if these parameters are carefully identified initially, over time, mechanical wear or environmental changes can cause inaccuracies. Since controller designs based on inaccurate models can cause poor

17

performance or even instability, provisions for this must be incorporated into the design. Sliding mode control can, so long as the extent of the parameter imprecision is known and bounded, provide a systematic approach to this problem. The following analysis deals only with structured (or parametric) uncertainties present in the model, and assumes no unstructured uncertainties involving unmodeled dynamics are present.

Recalling the dynamics model in Section A,

$$\ddot{z}(t) = hM^{-1}(f + gu(t)) + \dot{h}h^{-1}(\dot{z}(t) - u_c), \tag{2.16}$$

the forces, f, the input gain matrix, g, the mass matrix M, and the current u_c are not exactly known. Therefore, estimates of these quantities must be made to formulate the control solution and are denoted as $\hat{f}$, $\hat{g}$, $\hat{M}$, and $\hat{u}_c$, which are related to their exact values by:

$$\left| \hat{f}_i - f_i \right| \leq F_i$$

$$g = \Delta_g \hat{g} \qquad \left| \Delta_{g_{i,j}} \right| \leq D_{g_{i,j}}$$

$$M = \Delta_M \hat{M} \qquad \left| \Delta_{M_{i,j}} \right| \leq D_{M_{i,j}}$$

$$\left| \hat{u}_{c_i} - u_{c_i} \right| \leq U_{c_i}$$

Each F_i, $D_{g_{i,j}}$, $D_{M_{i,j}}$, and U_{c_i} represent bounded maximums on the estimation errors for any maneuver. Uncertainty in the transformation matrix, h, is assumed small and will not be included in the analysis.

With the above defined, and returning to the definition of the sliding surface

$$\sigma(\tilde{z}(t)) = S_1 \dot{\tilde{z}}(t) + S_2 \tilde{z}(t) + S_3 \int \tilde{z}(t)dt, \tag{2.17}$$

it's derivative

$$\dot{\sigma}(\tilde{z}(t)) = S_1 \ddot{\tilde{z}}(t) + S_2 \dot{\tilde{z}}(t) + S_3 \tilde{z}(t) \tag{2.18}$$

and the condition for global asymptotic stability

$$\dot{\sigma}_i(\tilde{z}(t)) = -\eta_i(\sigma_i(\tilde{z}(t))) \quad i = 1 \dots 6. \tag{2.19}$$

Dropping the "function of" notation, and recalling that $h = \hat{h}$, the sliding mode control law from Section A is

$$u = \hat{G}\hat{M}h^{-1}\left(\ddot{z}_{com} - h\hat{M}^{-1}\hat{f} + \dot{h}h^{-1}(\hat{u}_c - \dot{z}) + S_2\dot{\tilde{z}} + S_3\tilde{z} + \eta.*sgn(\sigma)\right). \tag{2.20}$$

where $\eta.*sgn(\sigma)$ is functionally equivalent to $F(\sigma, \phi)$, and the notation ".*" refers to element by element multiplication.

For the purposes of stability robustness analysis, the *sat* function used previously in the control law is replaced with a *sgn* function, since pure switching does not occur within the boundary layer, $-\phi_i < \sigma_i < +\phi_i$, the effectiveness of the gain, η_i, is reduced. In order to simplify further discussion, the matrix S_1 can be taken to be the identity matrix without loss of generality. Further, S_2 and S_3 are assumed to be diagonal matrices with separate bandwidth parameters for each mode. It follows that

$$\dot{\sigma} = \ddot{z}_{com} - hM^{-1}\Big(f + g\hat{G}\hat{M}h^{-1}(\ddot{z}_{com} - hM^{-1}\hat{f} + \dot{h}h^{-1}(\hat{u}_c - \dot{z})$$
$$+ S_2\dot{\tilde{z}} + S_3\tilde{z} + \eta.*sgn(\sigma))\Big) + \dot{h}h^{-1}(\dot{z} - u_c) + S_2\dot{\tilde{z}} + S_3\tilde{z} \tag{2.21}$$

which describes the dynamics of the closed loop sliding surface. For clarity (2.21) can be separated into n scalar equations written as

$$\dot{\sigma}(t)_i = \alpha(t)_i + \beta(t)_i \eta_i sgn(\sigma(t)_i) \quad i = 1\dots n, \tag{2.22}$$

where $\alpha_i(t)$ and $\beta_i(t)$ are time dependent scalar quantities for each $\dot{\sigma}_i(t)$ whose output will be bounded stable only if

$$\eta_i > \left\|\beta_i^{-1}(t)\right\|_\infty \left\|\alpha_i(t)\right\|_\infty \quad i = 1\dots n, \tag{2.23}$$

19

where $\left\|\alpha_i(t)\right\|_\infty$ denotes the infinity norm of $\alpha_i(t)$, also defined as $max\left|\alpha_i(t)\right| \ \forall \ t:[0,\infty)$. Selecting each η_i according to Eqn. (2.23) will overcome any destabilizing effects due uncertainty in either M, f, g, or u_c and provide a robust controller design. Although the uncertainties were originally defined in terms of maximum bounds, the complexity of (2.21) prevents their direct use in the analysis. Since the terms $\alpha_i(t)$ and $\beta_i(t)$ are not only functions of the parameter estimates, but are also functions of the time dependent states, simulations for each control maneuver could be performed to determine lower bounds for each η_i. Naturally, actuator saturation may limit the ability of the control to perform its task, while guaranteeing stability, and this issue may also be studied by simulation.

1 . A One-Dimensional Example

To more clearly illustrate the robustness analysis techniques outlined above, a scalar example will be given utilizing the maximum bounds formulation for parametric uncertainty. The analysis is an extension of the results from (Slotine and Li, 1991).

A simplified model of an underwater vehicle operating in the surge direction, x, can be written as

$$M\ddot{x} = f + gu, \tag{2.24}$$

where u is the control input. M is the mass of the vehicle, including the added mass associated with motion in the surge direction, f, the hydrodynamic drag force, and g is the input gain and is assumed to be positive. Estimation errors of the mass and input gain can be described multiplicatively by

$$\frac{1}{\mu} \leq \frac{\hat{M}}{M} \leq \mu \quad ; \quad \mu > 1, \tag{2.25}$$

$$\frac{1}{\gamma} \leq \frac{\hat{g}}{g} \leq \gamma \quad ; \quad \gamma > 1, \tag{2.26}$$

while the bound on the drag force uncertainty may be formulated as

$$\left| \hat{f} - f \right|_{\infty} = F, \tag{2.27}$$

and F is the infinity norm of $\left(\hat{f} - f \right)$, since f is a function of the time-varying state and can be either positive or negative.

Defining the sliding surface as

$$\sigma = \dot{\tilde{x}} + \lambda \tilde{x}, \tag{2.28}$$

and it's derivative

$$\dot{\sigma} = \ddot{\tilde{x}} + \lambda \dot{\tilde{x}}, \tag{2.29}$$

the resulting control law in terms of the estimates is

$$u = \frac{\hat{M}}{\hat{g}} \left(\ddot{x}_{com} - \frac{\hat{f}}{\hat{M}} + \lambda \dot{\tilde{x}} + \eta sgn(\sigma) \right). \tag{2.30}$$

Substituting (2.30) into (2.24) provides the closed loop system dynamics

$$\ddot{x} = \frac{f}{M} + \frac{\hat{M}}{M} \frac{g}{\hat{g}} \left(\ddot{x}_{com} - \frac{\hat{f}}{\hat{M}} + \lambda \dot{\tilde{x}} + \eta sgn(\sigma) \right), \tag{2.31}$$

and using Eqn. (2.29), the condition for global asymptotic stability becomes

$$\dot{\sigma}\sigma =$$

$$\sigma \left(\left| -\frac{f}{M} + \frac{\hat{M}}{M} \frac{g}{\hat{g}} \frac{\hat{f}}{\hat{M}} \right| + \left| \left(1 - \frac{\hat{M}}{M} \frac{g}{\hat{g}} \right) \cdot \left(\ddot{x}_{com} + \lambda \dot{\tilde{x}} \right) \right| \right) - \frac{\hat{M}}{M} \frac{g}{\hat{g}} \eta |\sigma| \leq 0. \tag{2.32}$$

21

By choosing the value of η to be large enough, the condition for stability will be satisfied despite the parametric uncertainty. After several steps of algebraic manipulation of Eqn. (2.32), and noting that $\mu^{-1} \le M\hat{M}^{-1} \le \mu$ and $f = \hat{f} + \left(f - \hat{f}\right)$, the minimum value of the switching gain, η, that will satisfy the stability requirement is

$$\eta \ge \frac{\gamma F}{\hat{M}} + \frac{\left|\hat{f}\right|_\infty}{\hat{M}} \cdot \left|\left(1 - \gamma\right)\right| + \left|\left(\mu\gamma - 1\right)\right| \cdot \left|\left(\ddot{x}_{com} + \lambda\dot{\tilde{x}}\right)\right|_\infty . \tag{2.33}$$

Inspection of (2.33) reveals that the required value of η for stability increases with the degree of parametric uncertainty. On the other hand, if a "perfect" system model is available, (2.33) reduces to

$$\eta \ge 0,$$

Although the robustness analysis relied on the use of a *sgn* function as the switching term, in practice the *sat* function is used to alleviate undesirable actuator chatter which gives rise to lower tracking precision. Since this approach does not provide fast switching within the boundary layer, tracking performance will degrade. Therefore a design trade-off between tracking performance and control activity exists. The degree of parameter uncertainty also plays a role in performance issues. The greater the modeling uncertainties, the greater the required control effort, which can cause actuator saturation and possibly system instability.

Use of a state observer can reduce the performance and robustness of the control (Cristi, et al., 1990(a), Cristi, et al., 1990(b)) show that errors may still be bounded, but robustness guarantees can not be proven to be superior than those using linear LQG/LTR techniques.

C. SLIDING MODE CONTROL IMPLEMENTATION FOR THE NPS PHOENIX VEHICLE

In order to verify the above theory, simulations have been conducted using a mathematical model of the NPS Phoenix vehicle, developed amongst other reasons for the purpose of experimental validation of the control concepts contained in this dissertation.

While details of the vehicle are given later in Chapter III, we should note for now that it has four cross-body thrusters and eight independently controllable fins, as well as twin rear propulsion screws to effect its motion control.

For the NPS Phoenix vehicle, the input control vector is thus defined as

$$u(t) = \left(\delta_{br} \ \ \delta_{bp} \ \ \delta_{sr} \ \ \delta_{sp} \ \ F_{ls} \ \ F_{rs} \ \ F_{bvt} \ \ F_{blt} \ \ F_{svt} \ \ F_{slt} \right)^T, \tag{2.34}$$

which contains ten independent actuators. The inputs δ_{br}, δ_{bp}, δ_{sr}, δ_{sp} are the bow rudder, bow plane, stern rudder, and stern plane surface deflections respectively (± 0.4 radian maximum. deflection). F_{ls} and F_{rs} are the forces due to the left and right rear screws (± 5 lb. maximum force). F_{blt} and F_{slt} are the bow and stern lateral thruster forces, while the bow and stern vertical thruster forces are defined as F_{bvt} and F_{svt} (± 2 lb. maximum force), all four of which are through-hull and can operate bi-directionally. All control inputs are shown pictorially in Figures 2.3 and 2.4. Only four fin commands are needed since they act in coupled pairs, and the simulation is based on the assumed numerical values of hydrodynamic and thruster coefficients given in Appendix A.

With the defined control inputs, the resulting 6×10 input gain matrix for this actuator configuration is

$$\hat{g} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ u|u|\hat{Y}_{dbr} & 0 & u|u|\hat{Y}_{dsr} & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & u|u|\hat{Z}_{dbp} & 0 & u|u|\hat{Z}_{dsp} & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u|u|\hat{M}_{dbp} & 0 & u|u|\hat{M}_{dsp} & 0 & 0 & -x_{bvt} & 0 & -x_{svt} & 0 \\ u|u|\hat{N}_{dbr} & 0 & u|u|\hat{N}_{dsr} & 0 & -y_{ls} & -y_{rs} & 0 & x_{blt} & 0 & x_{slt} \end{bmatrix}.$$

From this, it can be seen that $\hat{g}$ does not have full rank since the roll mode is not directly controllable using the inputs available. In this situation, the inverse of $\hat{g}$ does not exist, and no direct control solution can be obtained. The uncontrollable mode is the roll motion and must be passively stable for a properly designed underwater vehicle. With this assumption, the input gain matrix can be redefined by re-ordering the state vector and separating the roll equation of motion. This results in a new state vector

$$x^{*}(t) = \left(u(t) \ \ v(t) \ \ w(t) \ \ q(t) \ \ r(t) \ \ p(t) \right)^{T}, \tag{2.35}$$

23

and

$$z^*(t) = \begin{pmatrix} X(t) & Y(t) & Z(t) & \theta(t) & \psi(t) & \phi(t) \end{pmatrix}^T, \tag{2.36}$$

where the corresponding input gain matrix is

$$\hat{g}^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ u|u|\hat{Y}_{\delta br} & 0 & u|u|\hat{Y}_{\delta sr} & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & u|u|\hat{Z}_{\delta bp} & 0 & u|u|\hat{Z}_{\delta sp} & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & u|u|\hat{M}_{\delta bp} & 0 & u|u|\hat{M}_{\delta sp} & 0 & 0 & -x_{bvt} & 0 & -x_{svt} & 0 \\ u|u|\hat{N}_{\delta br} & 0 & u|u|\hat{N}_{\delta sr} & 0 & -y_{ls} & -y_{rs} & 0 & x_{blt} & 0 & x_{slt} \end{bmatrix},$$

which has full rank and is invertible, such that $\hat{g}^* \hat{G}^* = I$. Rearrangement of the state vector not only impacts $\hat{g}$, but the mass and transformation matrices must also be altered resulting in the following

$$M^* = \begin{bmatrix} m - X_u & 0 & 0 & mz_G & -my_G & 0 \\ 0 & m - Y_v & 0 & 0 & mx_G - Y_r & -(mz_G + Y_{\dot{p}}) \\ 0 & 0 & m - Z_w & -(mx_G + Z_q) & 0 & my_G \\ mz_G & 0 & -(mx_G + M_{\dot{w}}) & I_{yy} - M_q & -I_{yz} & -I_{xy} \\ -my_G & mx_G - N_v & 0 & -I_{yz} & I_{zz} - N_r & -(I_{xz} + N_{\dot{p}}) \\ 0 & -(mz_G + K_{\dot{v}}) & my_G & -I_{xy} & -(I_{xz} + K_r) & I_{xx} - K_p \end{bmatrix},$$

and

$$h^*(z^*(t)) = \begin{bmatrix} T_t^*(z^*(t)) & 0 \\ 0 & T_r^*(z^*(t)) \end{bmatrix},$$

24

where

$$T_l^*(z^*(t)) = \begin{bmatrix} cos\,\psi cos\theta & cos\,\psi sin\theta sin\phi \,-\, sin\,\psi cos\phi & cos\,\psi sin\theta cos\phi \,+\, sin\,\psi sin\phi \\ sin\,\psi cos\theta & sin\,\psi sin\theta sin\phi \,+\, cos\,\psi cos\phi & sin\,\psi sin\theta cos\phi \,-\, cos\,\psi sin\phi \\ -sin\theta & cos\theta sin\phi & cos\theta cos\phi \end{bmatrix},$$

and

$$T_r^*(z^*(t)) = \begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi \,/\, cos\theta & cos\phi \,/\, cos\theta & 0 \\ sin\phi tan\theta & cos\phi tan\theta & 1 \end{bmatrix},$$

where the "*" superscript denotes the matrices of the rearranged system. Eqns. (2.13) through (2.15) must now use the redefined matrices and vectors to compute the control solution, $u(t)$.

As an aid to understanding this procedure and verifying the stability of the result, the following definitions and remarks are given:

Definition: A system, Σ,

Σ: $\dot{x}(t) = f(x(t)) + g(x(t))u(t)$, $x(t) \in \Re^{n \times l}$, $u(t) \in \Re^{m \times l}$, $m \geq n$,   and f,   are smooth vector fields on $\Re^n$ and g are smooth matrix functions $\in \Re^{n \times m}$ is said to be _directly controllable_ if g has rank of n, and a unique G exists such that $gG = I \in \Re^{n \times n}$ $\forall$ $t:[t_0, \infty)$.

Remark 1: It follows that Σ, a directly controllable system may be globally asymptotically stabilized by the control $u(t) = G(x(t))\{-f(x(t)) - \eta .*sgn(x(t))\}$. This can be proven using the sliding surfaces $\sigma = x$, and $\eta > 0$.

Lemma: For the system, Σ, where rank g is $< n$, Σ can be separated into Σ_1 and Σ_2 where Σ_1 is a directly controllable subsystem and Σ_2 is unforced by $u(t)$.

Proof: If g is rank deficient, a reordering of the state vector can be performed with components

25

$$x = \begin{bmatrix} x_1 \\ -- \\ x_2 \end{bmatrix}, \ x_1 \in \Re^{p \times 1}, \ x_2 \in \Re^{(n-p) \times 1}$$

such that if the rank of g is p, with $p < n$, $g \in \Re^{n \times m}$ is then written as

$$g = \begin{bmatrix} g_1 \\ -- \\ 0 \end{bmatrix}; \ g_1 \in \Re^{p \times m},$$

and since rank $rank(g_1) = p$, then a G can always be found so that $g_1 G = I \in \Re^{p \times p}$. Σ_1 is then *directly controllable*, and Σ_2 is unforced by $u(t)$.

Σ_1: $\dot{x}_1(t) = f_1\big(x_1(t), x_2(t)\big) + g_1\big(x_1(t), x_2(t)\big)u(t)$, $x_1 \in \Re^{p \times 1}$, $x_2 \in \Re^{(n-p) \times 1}$, $u(t) \in \Re^{m \times 1}$

Σ_2: $\dot{x}_2(t) = f_2\big(x_1(t), x_2(t)\big)$

Lemma: The system Σ with $rank(g_1) = p < n$ may be globally ultimately stabilized if its subsystem, Σ_2 is passively stable, i.e.

$$I = \int_{\tau=0}^{\tau=t} x_2^T(\tau) f_2\big(x_1(\tau), x_2(\tau)\big) d\tau < 0, \ x_1(t) \rightarrow 0, \ t \rightarrow \infty.$$

Proof: Using the concept of passive stability, we define Lyapunov candidate functions, $V_1\big(x_1(t)\big) = 0.5 x_1^T(t) x_1(t)$, with $V_2\big(x_2(t)\big) = 0.5 x_2^T(t) x_2(t)$. For the entire state, $x(t)$, the Lyapunov function is $V(t)$, where

$$V(t) = V_1(t) + V_2(t).$$

It follows that $\dot{V} < 0$ if $\dot{V}_1 < 0$, and $\dot{V}_2 < 0$. $\dot{V}_1$ is negative definite for all t, by virtue of the sliding mode control design for $u(t)$ remembering that both $x_1(t)$ and $x_2(t)$ are known. Thus $t \rightarrow \infty$ implies that $x_1(t) \rightarrow 0$. $\dot{V}_2$ is < 0 for all $t > 0$ *iff*, $x_2^T(t)\dot{x}_2(t) < 0$, $\forall \ t:[0, \infty)$ leading to the sufficient condition,

$$I = \int_{\tau=0}^{\tau=t} x_2^T(\tau) f_2\big(x_1(\tau), x_2(\tau)\big) d\tau < 0, \ x_1(t) \rightarrow 0, \ t \rightarrow \infty.$$

The integral I, is a measure of the net energy into the subsystem, Σ_2, induced by $x_1(t)$, and dissipated by $x_2(t)$.

Remark: Ultimate stabilization of Σ_2, and thus Σ can be proved if there exists a t_2 such that $I(t_2,\infty) < 0$ for all $t > t_2$.

Remark: Passive stability for Σ_2 also implies that in tracking problems where the state $x_1(t)$ is driven to the bounded value $x_{com}(t)$ such that the error $\tilde{x}_1(t) \rightarrow 0$, $t \rightarrow \infty$, $x_2(t)$ will be bounded.

(Note that $x_1 = (u \; v \; w \; q \; r)^{\mathrm{T}}$ and $x_2 = p$ for the control design implementation studied here.)

D.   CONTROL ALLOCATION

One method to solve the inverse of $\hat{g}(x(t),z(t))$ is to use the minimum norm solution or weighted minimum norm solution outlined in detail in Appendix C. Use of the weighted minimum norm will be the most appropriate since certain actuators will have large or small effects depending on the vehicle speed and orientation with respect to a commanded path. Equal weighting of all inputs will cause certain actuators to saturate under different operating conditions. Since a continuous control over the entire speed range is desired, the elements of the weighting matrix, W, should be designed as functions of vehicle forward speed, and perhaps the magnitude and direction of any currents present, scaled by the maximum level of individual actuators. For the simulations to follow, the weighting matrix is diagonal and represented by

$$W = \begin{bmatrix} w_{\delta_{br}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{\delta_{bp}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{\delta_{sr}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{\delta_{sp}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{F_{ls}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{F_{rs}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_{F_{bvt}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{F_{blt}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{F_{svt}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{F_{slt}} \end{bmatrix},$$

where each w_i corresponds to each actuator given in Eqn. (2.34). Using this means continuous control over the entire speed range can accomplished. For example, the control surfaces should be given the highest weighting at cruising speed, while the thrusters should dominate when the vehicle is maneuvering at low speed or during hovering operations. Manipulation of the weights either as a function of state or by a pre-defined plan is a convenient way to reconfigure control systems while maintaining stability.

E.   SIMULATION RESULTS

A two dimensional tracking simulation is now presented which demonstrates the performance of the sliding mode controller. Motion of the vehicle is restricted to the horizontal plane with the desired trajectory shown in Figure 2.5, which is composed of two straight segments 30 feet long and a quarter circle of radius 60 feet. Using the command generator algorithms outlined in Appendix B, motion profiles for position, velocity, and acceleration of the vehicle surge motion have been specified. For the maneuver, the trajectory arc length is 154.24 feet with a specified maximum velocity, u_{max}, of 1.0 ft / sec and maximum acceleration, $\dot{u}_{max}$, of 0.035 ft / sec^2. Since the trajectory involves a turn of 90 degrees, the global position and heading commands must be kinematically consistent, and this can be achieved by using Eqn. (2.3). Assuming no current, the global commands can be derived from

$$z_{com}(t) = \int h(z(t)) x_{com}(t) dt$$

$$\dot{z}_{com}(t) = h(z(t)) x_{com}(t)$$

$$\ddot{z}_{com}(t) = \dot{h}(z(t)) x_{com}(t) + h(z(t)) \dot{x}_{com}(t)$$

and for each segment,

$$x_{com}(t) = \begin{bmatrix} u_{com}(t) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad 0 \leq t \leq 58.7 \ sec. \quad ; \quad 0 \leq s \leq 30.0 \ ft.$$

$$x_{com}(t) = \begin{bmatrix} u_{com}(t) \\ 0 \\ 0 \\ r_{com}(t) \\ 0 \\ 0 \end{bmatrix} \qquad 58.7 \leq t \leq 153.0 \ sec. \quad ; \quad 30.0 \leq s \leq 124.24 \ ft.$$

$$x_{com}(t) = \begin{bmatrix} u_{com}(t) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad 153.0 \leq t \leq 211.2 \ sec. \quad ; \quad 124.24 \leq s \leq 154.24 \ ft.$$

and

$$r_{com}(t) = \frac{u_{com}(t)}{R} \quad ; \quad 58.7 \leq t \leq 153.0 \ sec.,$$

29

where $R = 60\,ft.$, the radius of the turn. From this, the global motion command vectors are

$$z_{com}(t) = \begin{bmatrix} X(t) \\ Y(t) \\ \psi(t) \end{bmatrix}_{com} \quad \dot{z}_{com}(t) = \begin{bmatrix} \dot{X}(t) \\ \dot{Y}(t) \\ \dot{\psi}(t) \end{bmatrix}_{com} \quad \ddot{z}_{com}(t) = \begin{bmatrix} \ddot{X}(t) \\ \ddot{Y}(t) \\ \ddot{\psi}(t) \end{bmatrix}_{com} .$$

Since the path is composed of straight segments and a quarter circle, some of the global motion command vectors are discontinuous at the beginning and end of the turn as shown in Figures 2.6, 2.7, and 2.8. A more sophisticated trajectory generation method could be used to remove these discontinuities.

Two control cases are performed for tracking the desired path, the first, (Case 1), allows all control inputs to be available, while the second, (Case 2), uses no lateral thruster assistance, relying only on the rudders and rear screws for control. To demonstrate controller robustness, simulation results for Case 1 also contain the responses for varying degrees of structured parametric uncertainty. Results using four different levels of mismatch in the input gain matrix, $\hat{g}^*$, and a single value of mismatch for the non-linear feedforward terms, $\hat{f}^*$, are given. The differences from the values used in the model were

$$\hat{g}_1^* = g^*$$

$$\hat{g}_2^* = 10.0.*g^*$$

$$\hat{g}_3^* = 12.0.*g^*$$

$$\hat{g}_4^* = 0.4.*g^*$$

and

$$\hat{f}_1 = \begin{pmatrix} 4.0 \\ 4.0 \\ 1.0 \\ 1.0 \\ 4.0 \\ 1.0 \end{pmatrix}.*f .$$

30

No parameter uncertainties for Case 2 were considered nor any disturbances such as current, and the tunnel thrusters were modeled as simple forces; no thruster dynamics were used. Matrix S_1 was chosen for convenience to be identity and since no current was present, no integral control was necessary, therefore S_3 was set to 0. For matrix S_2, a diagonal matrix was used of the form

$$S_2 = \begin{bmatrix} \lambda_X & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_Y & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_Z & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_\theta & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_\psi & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_\phi \end{bmatrix},$$

and with this choice, a simple sliding surface design followed such that the prescribed tracking error dynamics on the surface become

$$\dot{\tilde{z}}_i(t) + \lambda_i \tilde{z}_i(t) = 0 \; ; \; i = X, Y, Z, \theta, \psi, \text{ and } \phi,$$

which has well behaved dynamics, and

$$\sigma(\tilde{z}(t)) = \begin{pmatrix} \sigma_X & \sigma_Y & \sigma_Z & \sigma_\theta & \sigma_\psi & \sigma_\phi \end{pmatrix}^T.$$

The values of each η_i and ϕ_i, and λ_i used in the simulation were:

Table 2.1 Sliding Mode Control Gains

Mode	X	Y	Z	θ	ψ	ϕ
λ	0.5	0.5	0.5	0.5	1.0	N/A
η	0.5	0.5	0.1	0.1	0.3	N/A
ϕ	1.0	1.0	1.0	1.0	0.1	N/A

Note that the last sliding surface is included, although the roll motion is not controlled. It follows that, maintaining dimensional consistency, the controls are now computed as

$$u_I(t) = [\hat{G}^* : 0]\left(\hat{M}^* \hat{h}^{*-1} \ddot{z}^*_{com}(t) - \hat{f}^*\right) \qquad (2.37)$$

$$u_2(t) = [\hat{G}^* : 0]\hat{M}^* \hat{h}^{*-1}\left(-\dot{\hat{h}}^* \hat{h}^{*-1} \dot{z}^* + S_2^* \dot{\tilde{z}}^*(t)\right) \qquad (2.38)$$

$$u_3(t) = [\hat{G}^* : 0]\hat{M}^* \hat{h}^{*-1} F(\sigma^*, \phi^*) \qquad (2.39)$$

Results of the first case are shown in Figures 2.9 through 2.18. Inspection of the position and heading response show that the vehicle perfectly tracked the commanded trajectory when using the exact input gain matrix $\hat{g}_I^*$ in the controller design. Although the system is non-minimum phase, the controller design has been able to compensate for this since full state feedback is available. Degradation of the tracking response from increasing levels of mismatch in $\hat{g}^*$ and $\hat{f}^*$ is evident, however, the trajectories of each σ_i approach zero as shown in Figures 2.11 through 2.13.

Referring to Figures 2.14 through 2.16, spikes in the control input at the beginning and end of the turn are evident, and are caused by the discontinuities of the acceleration commands at these locations. Using the input gain matrix $\hat{g}_4^*$ in the control design provided tracking performance equivalent to using $\hat{g}_I^*$ (no mismatch), but this came at the expense of high actuator activity. Figure 2.17 shows a comparison of the stern rudder response for both $\hat{g}_I^*$ and $\hat{g}_4^*$. Figure 2.18, shows that the roll response is passively stable and is only slightly excited by the turning maneuver.

The second case uses no lateral thrusters for control. Removal of the thruster contribution was accomplished by setting the values w_{Fblt} and w_{Fslt} to zero in the weighting matrix, W. Figures 2.19 through 2.23 show the results of this simulation, using two different weightings for the rudders $w_{\delta rb}$ and $w_{\delta rs}$. Using Weighting 1, $w_{\delta rb} = w_{\delta rs} = 1.0$, resulted in tracking errors as shown in Figures 2.19 and 2.20, and was caused by rudder saturation (Figure 2.21). The stern rudder, after an initial negative deflection, rotated positive in effort to satisfy the lateral force requirement, resulting in a decreased turning moment.

Since rudder saturation is an undesirable condition, a method must be found that can reduce the rudder stroke and still maintain accurate tracking. There are two solutions to this problem: (1) use larger area rudders to increase their effectiveness, or (2) reduce the weights $w_{\delta rb}$ and $w_{\delta rs}$ until the maximum stroke reduces below the saturation limit. Using the latter will not only reduce control surface saturation, it also prevents the stern rudder

from decreasing the turning moment. The second set of curves denoted by Weighting 2, $w_{\delta rb} = w_{\delta rs} = 0.1$, shows that reducing the weights provides perfect tracking and an overall reduction in the control effort from both the rudders and rear screws. The roll response is again passively stable shown in Figure 2.23.

1. Remark on Non-Minimum Phase Effects

In course of this work, it has become apparent that non-minimum phase effects can arise in two different ways. First, combinations of elements in the input dynamics matrices (for a linearized system) that result in unstable zeros in some input/output transfer functions. Secondly, in some systems, unstable zeros arise because of particular elements in the output matrix. Output linearization of a system belonging to the second class will exhibit unstable internal dynamics, and can not be stabilized by output feedback. In the first case, sliding mode formulations including the complete state, compensate for any unstable zeros and are stabilizable. For the case of underwater vehicles with full state feedback, non-minimum phase effects on the stern planes and rudders is fully compensated.

2. Discrete Time Implementation

While theoretical development has been performed in the continuous time domain, real time control is implemented in the discrete time domain. Time discretization is performed using the Euler transformation, $\dot{x}(kT) \approx \left(x_{k+1} - x_k \right) / T$.

F.   CONCLUSIONS

Results from this chapter have shown that the formulation for a MIMO Sliding Mode controller performs very well, even with parameter mismatch. The simulations have also shown that adequate control authority is needed when performing path tracking maneuvers, and the weights for the inverse solution of the input gain matrix can play an important role in the control performance. Further work is needed to automate the computation of the weights for the control surfaces and thrusters, preferably as a continuous function of vehicle forward speed. From this, control saturation can be kept to minimum throughout a given maneuver.

A robustness analysis was also presented which included any modeling inaccuracies in the mass, dynamics, and input gain parameters. The analysis provided a design procedure to ensure stability despite the uncertainties. Further work is needed to develop a design methodology to eliminate possible actuator saturation due to these modeling errors.

Although, the control design methodology presented appears robust and well behaved in simulation, other real factors not included are lags in thruster response. If relatively long lags are present, these effects should be included in the general analysis.

G. CHAPTER II FIGURES

RIGID – BODY

C.G.

r_G

BODY – FIXED FRAME

y v, q

O

u, p

r_B

C.B.

w, r

x

z

$r_{O/C}$ $r_{O/G}$

Y_C C

CURRENT FRAME X_C

Z_C $r_{C/G}$

GLOBAL FRAME

G

Y X

Z

$r_{O/G} = r_{C/G} + r_{O/G}$
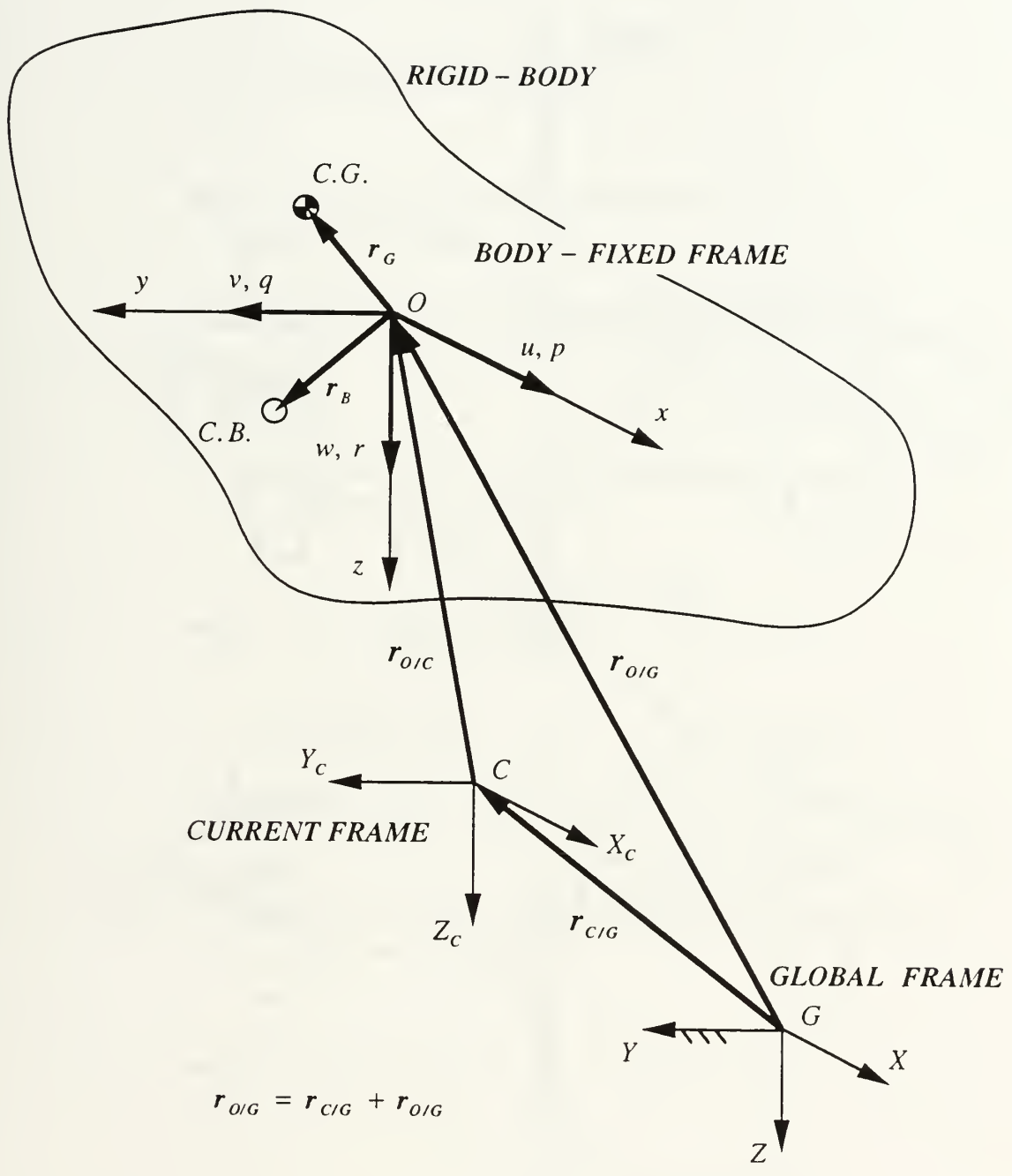
2.1 Coordinate Frame Representation.
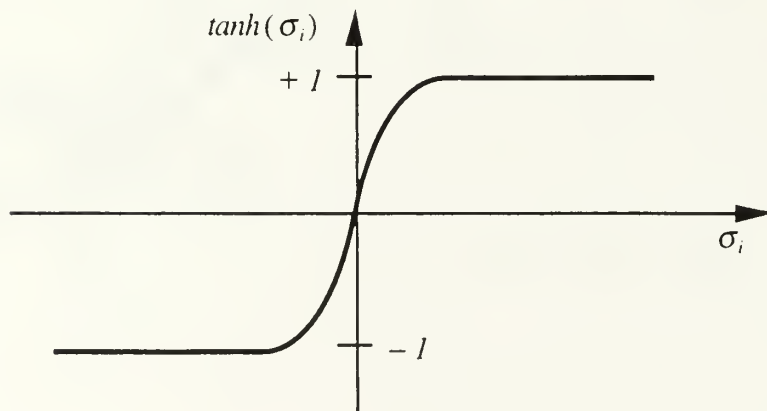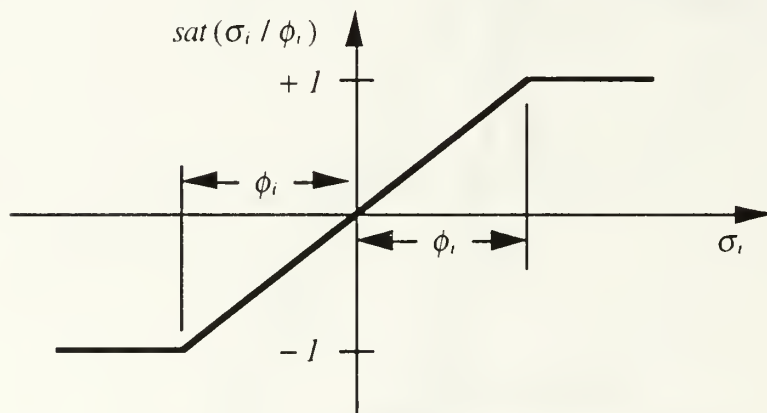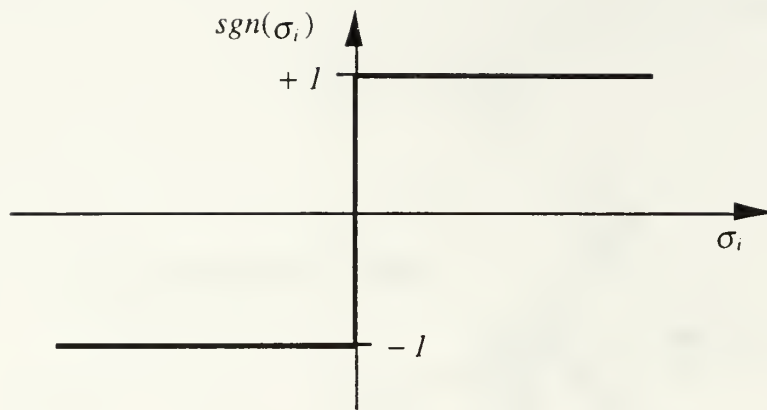
2.2 Three Possible Switching Functions for the Sliding Mode Controller.

TOP VIEW

$+\delta_{sr}$

$+\delta_{br}$

x

$q, \dot{q}$

$v, \dot{v}$

y

SIDE VIEW

$u, \dot{u}$

$p, \dot{p}$

x

$+\delta_{sp}$

$+\delta_{bp}$

$r, \dot{r}$

$w, \dot{w}$

z
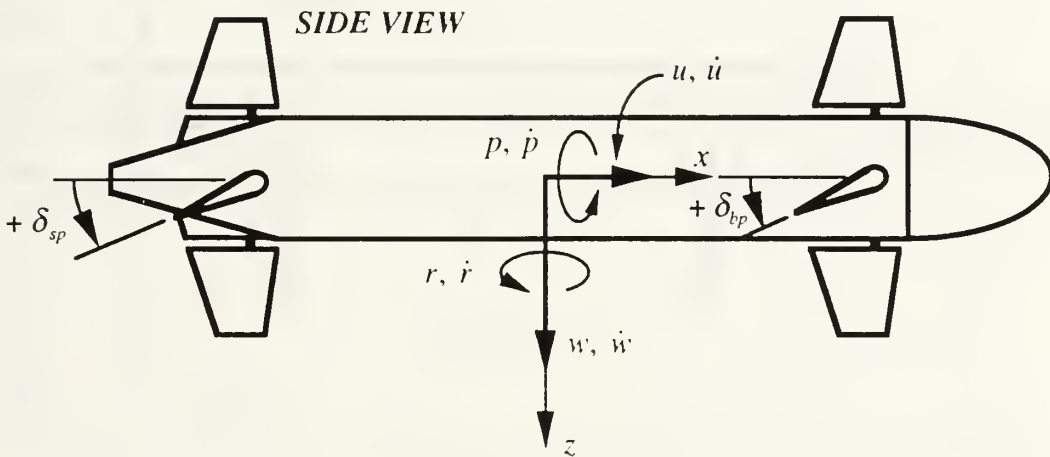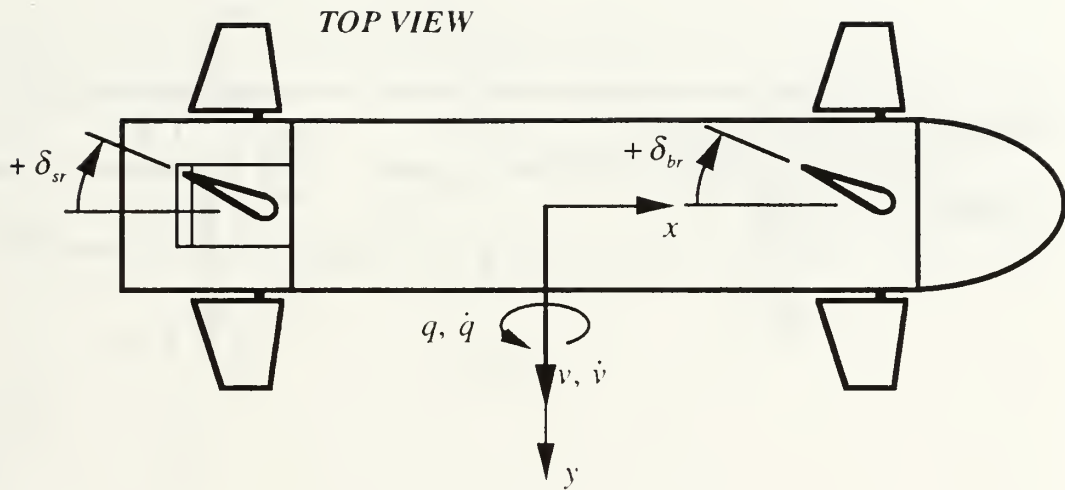
2.3 Control Surface Convention of the NPS Phoenix.
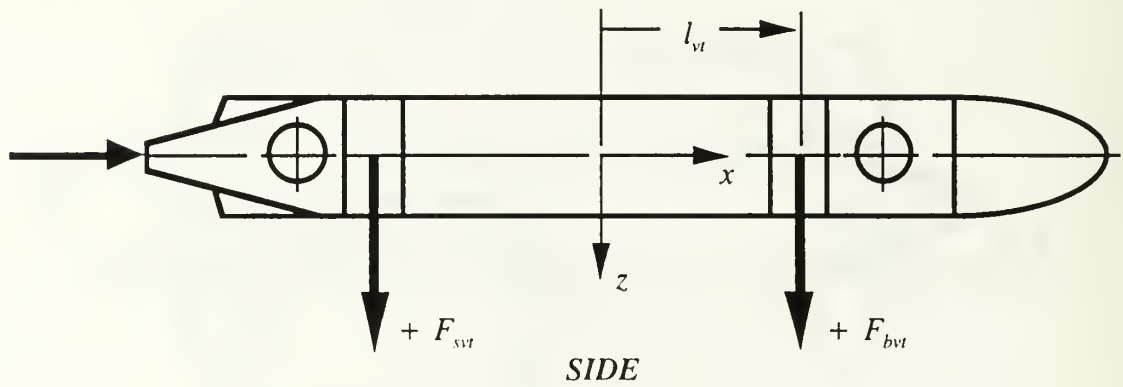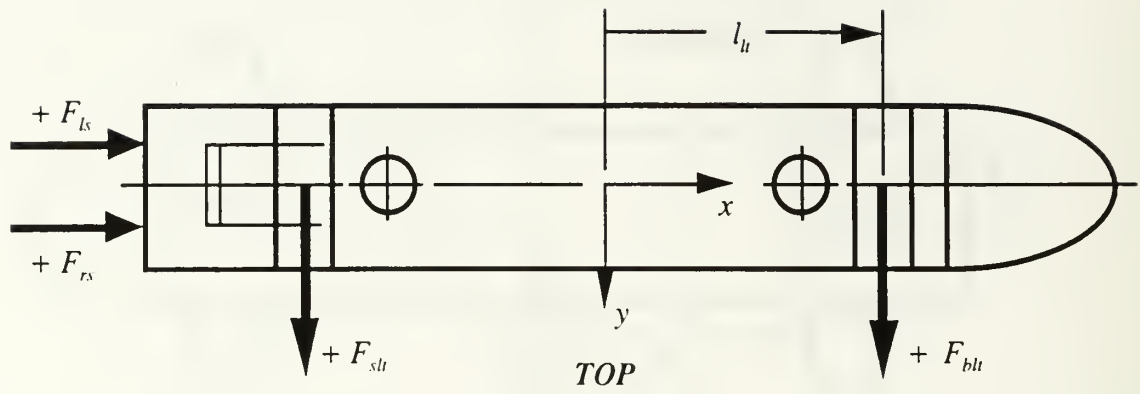
2.4 Thruster Force Convention of the NPS Phoenix.
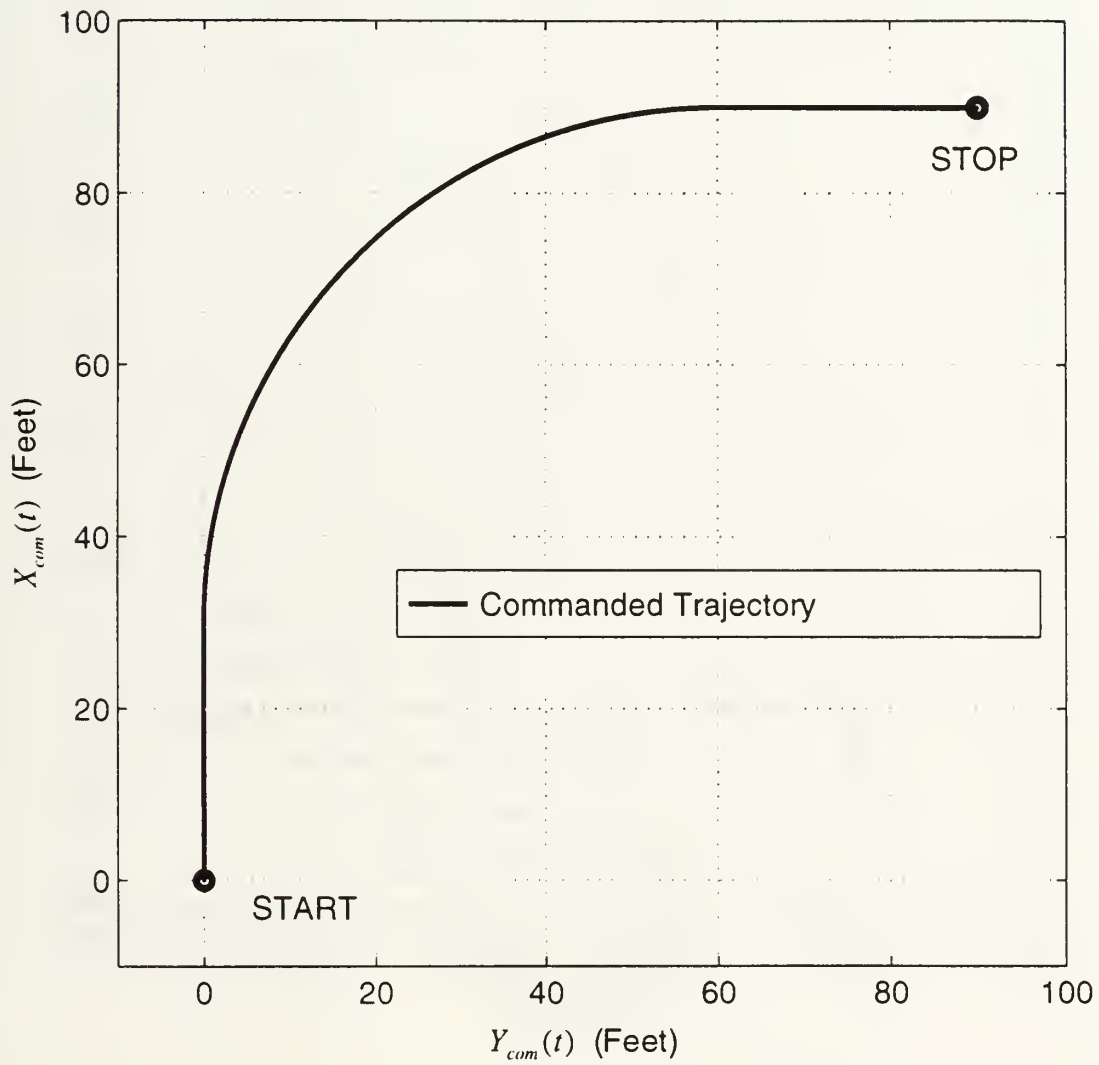
2.5 Commanded Trajectory for Vehicle Performance Simulation.
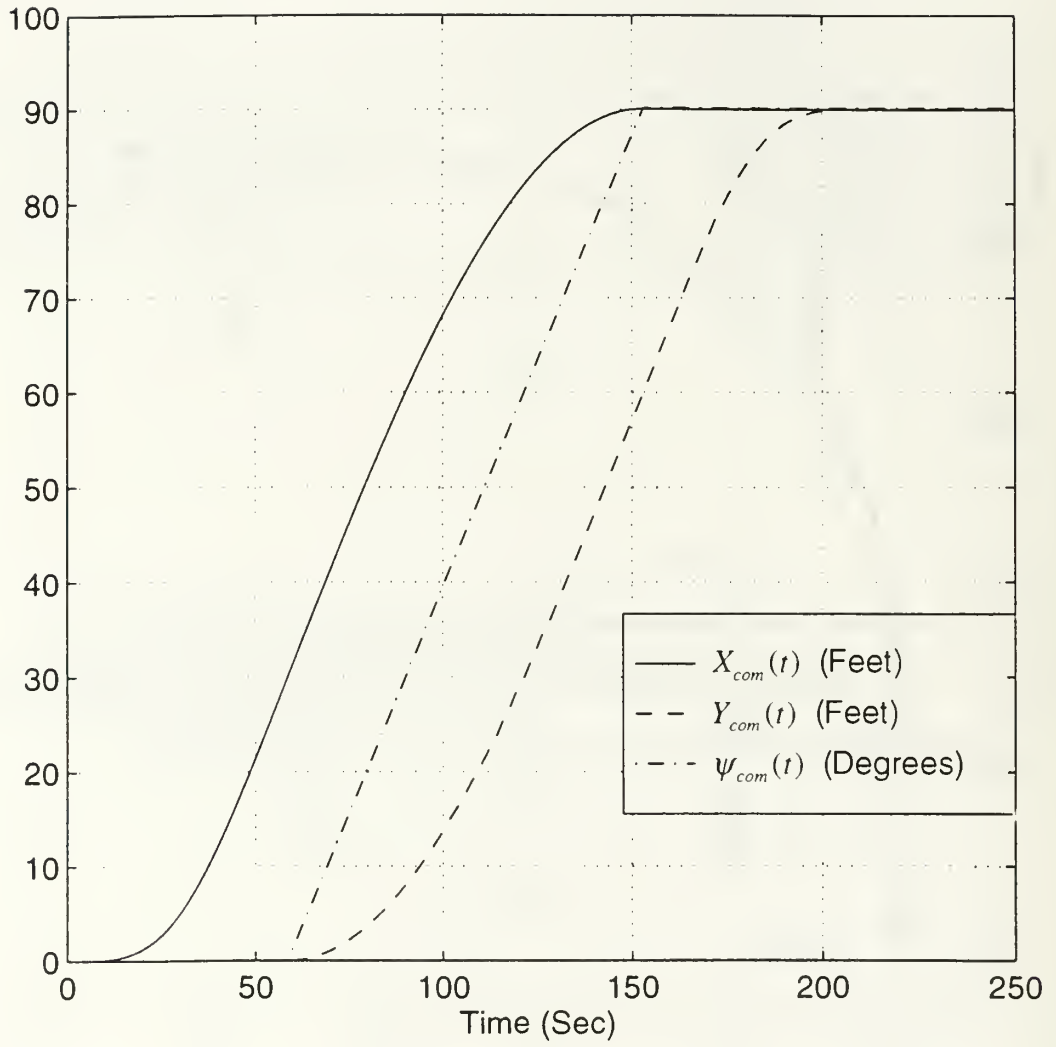
2.6 Commanded Global Position Profiles vs. Time for Vehicle Performance Simulation.
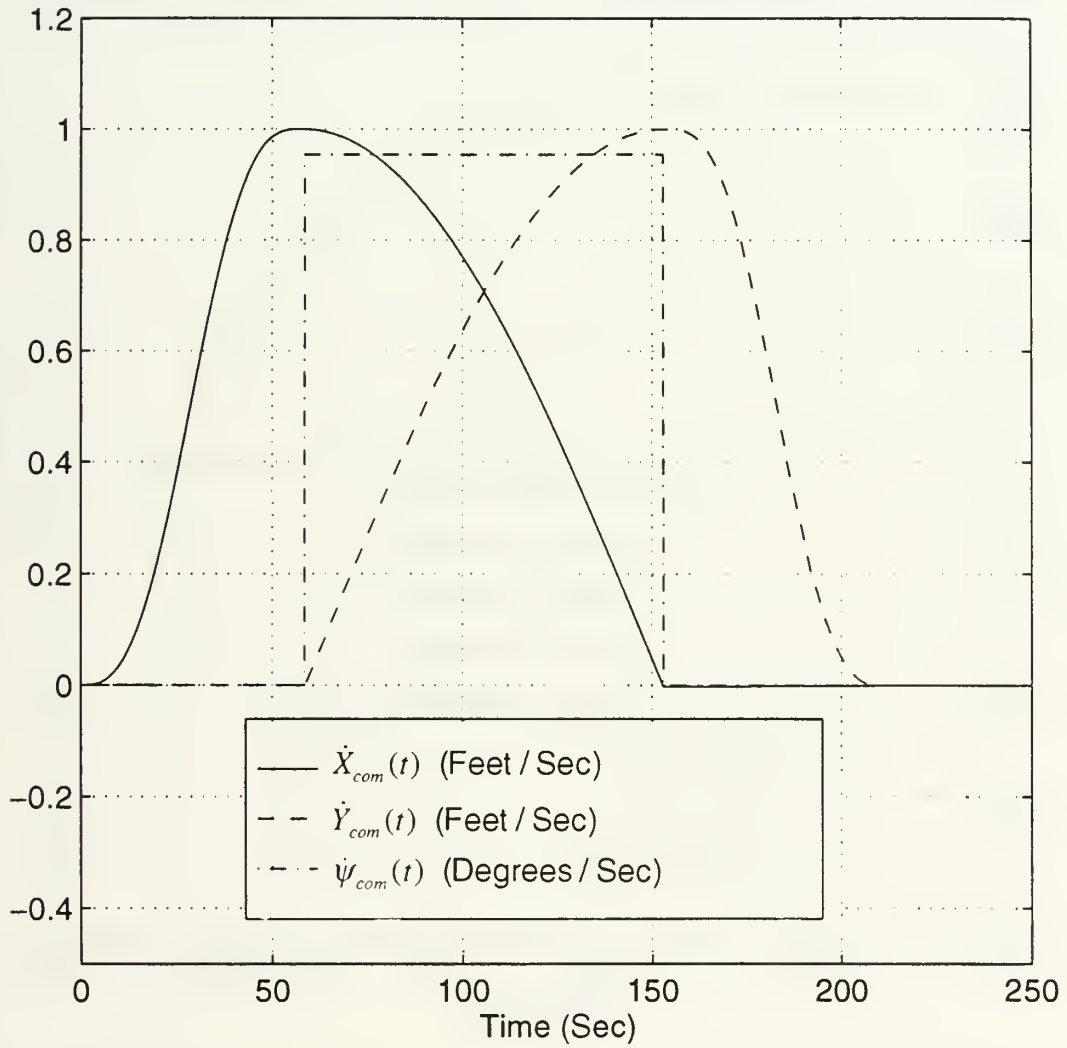
2.7 Commanded Global Velocity Profiles vs. Time for Vehicle Performance Simulation.
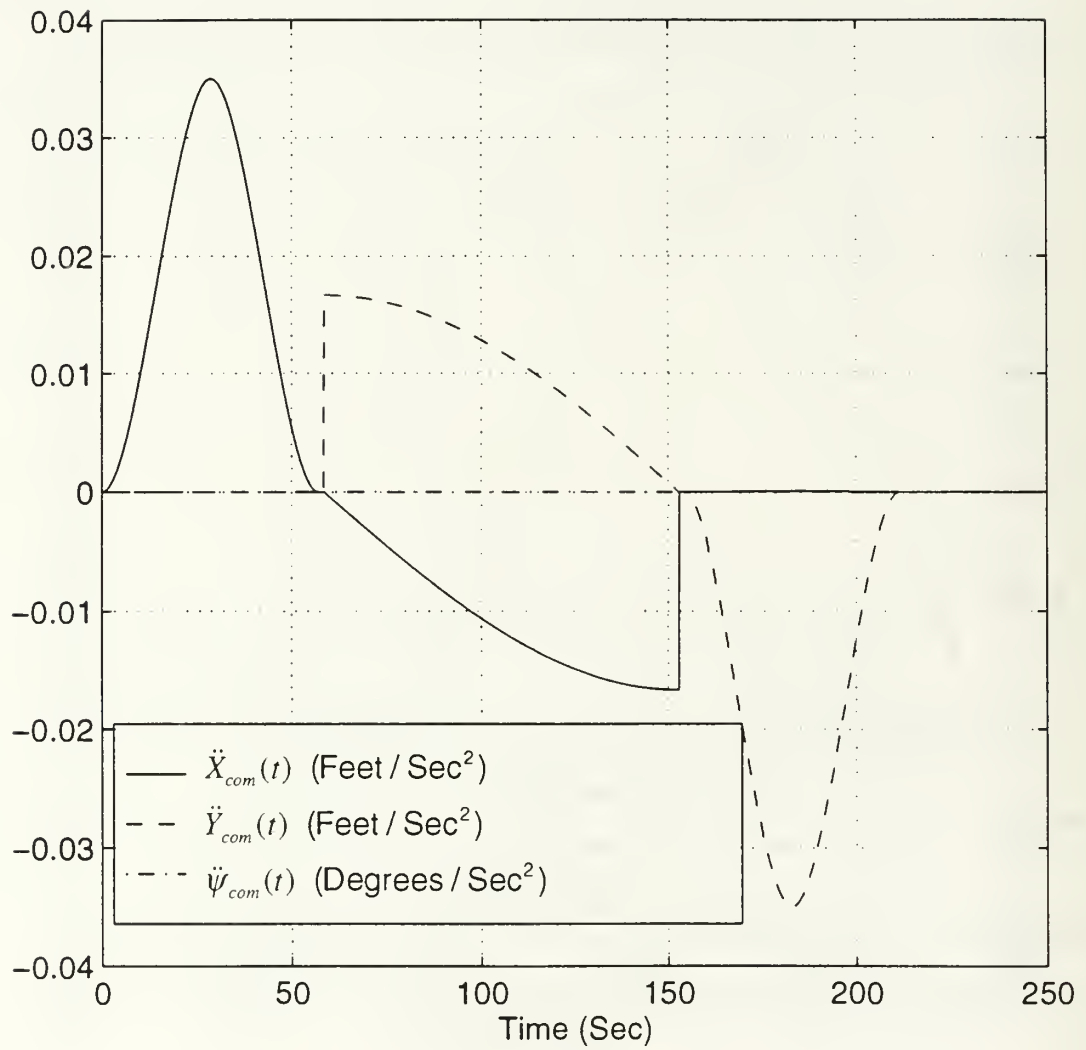
2.8 Commanded Global Acceleration Profiles vs. Time for Vehicle Performance Simulation.
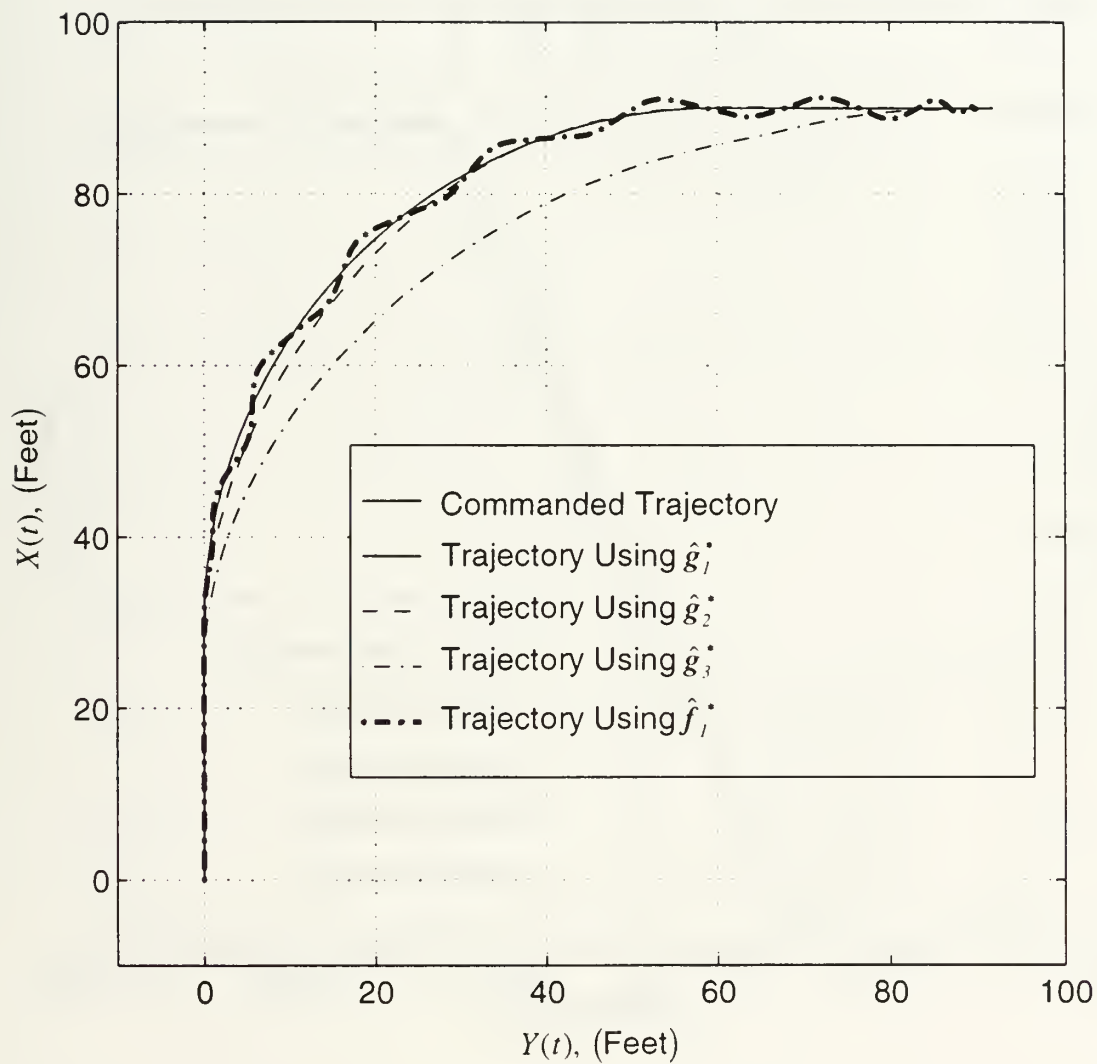
2.9 Position Response for Case 1, Showing the Tracking Degradation Using Various Levels of Parameter Mismatch.
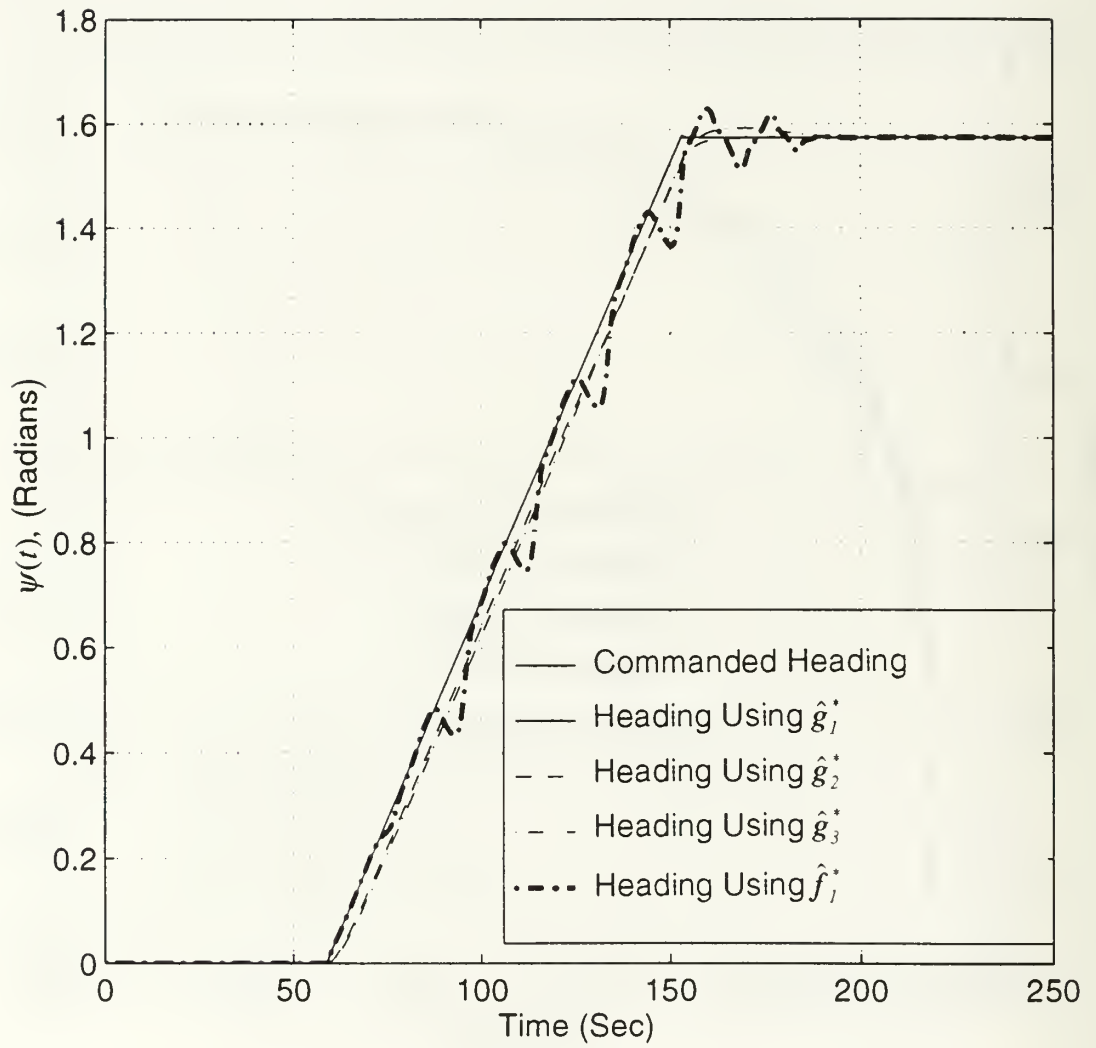
2.10 Heading Angle Response vs. Time for Case 1, Showing the Tracking Degradation Using Various Levels of Parameter Mismatch.
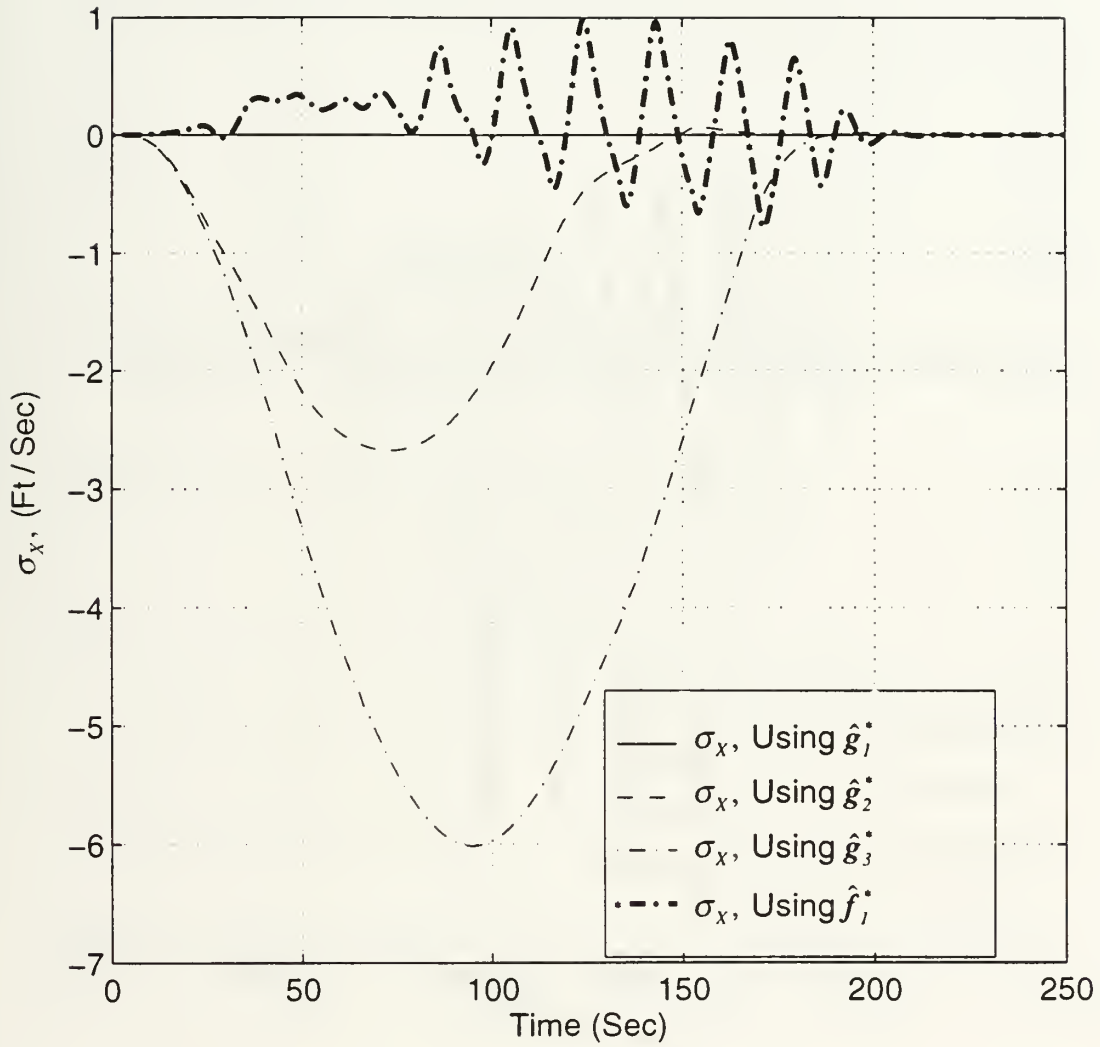
2.11 Response of σ_x Using Various Levels of Parameter Mismatch.

47
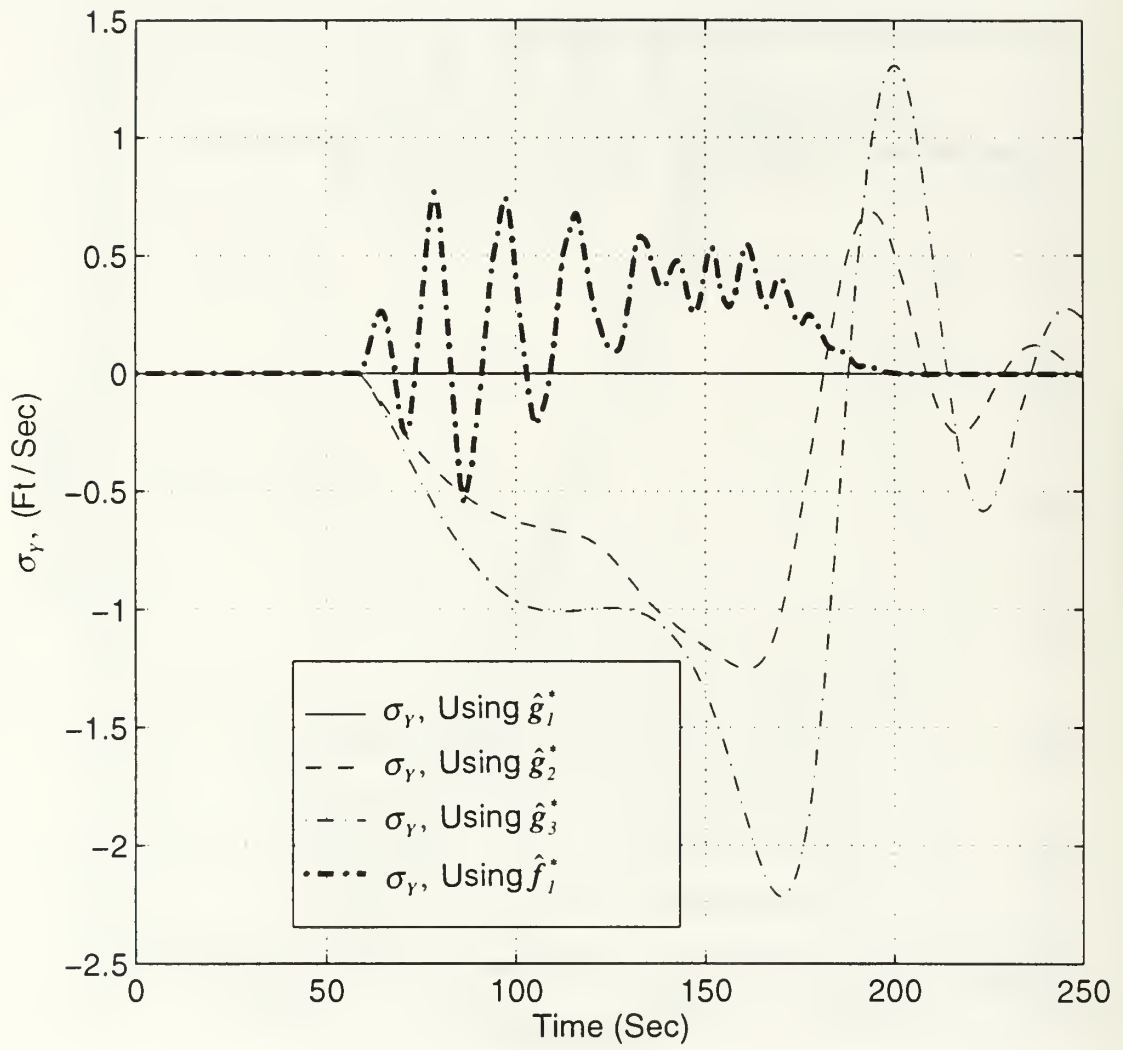
2.12 Response of σ_Y Using Various Levels of Parameter Mismatch.
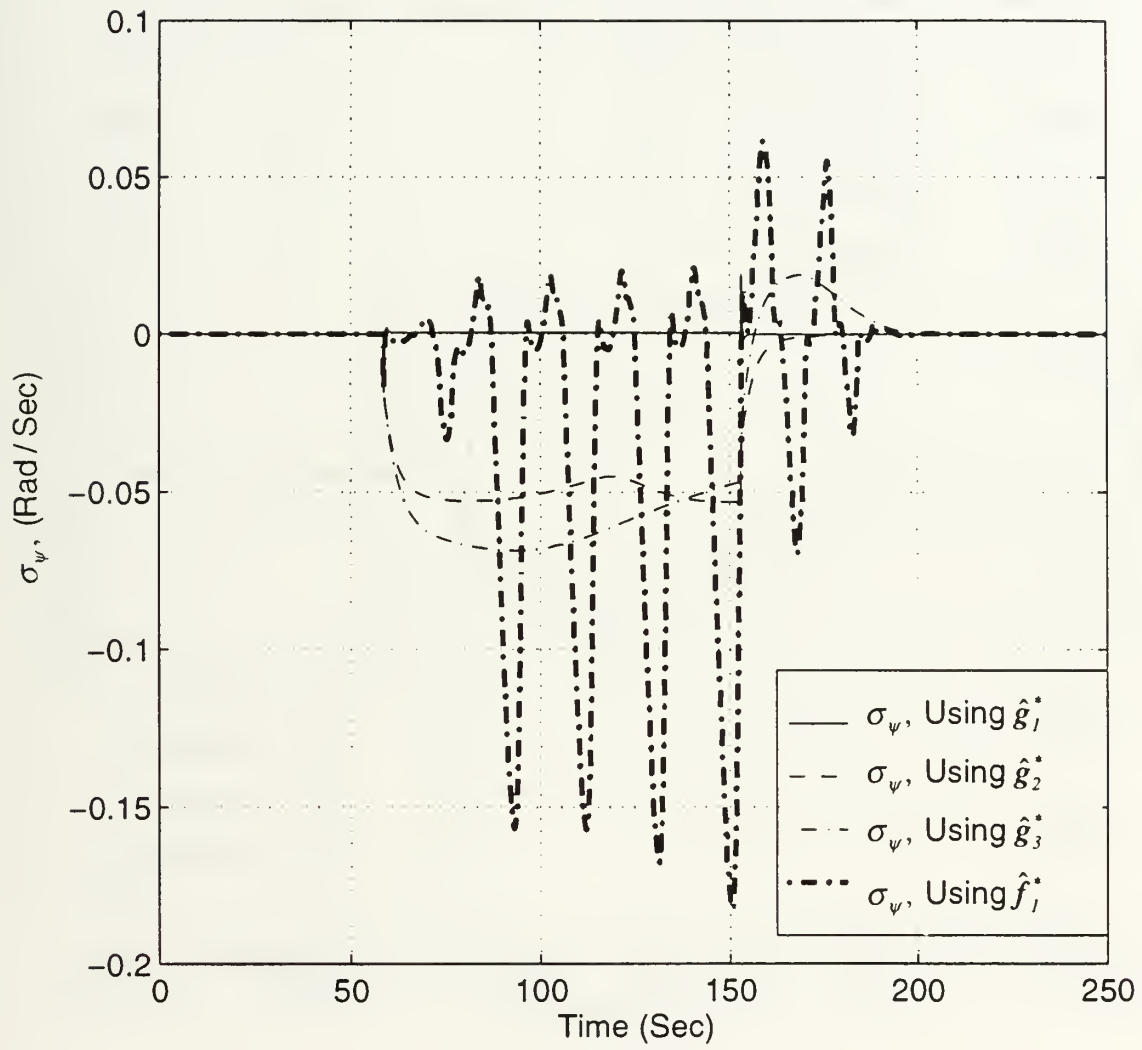
2.13 Response of σ_ψ Using Various Levels of Parameter Mismatch.
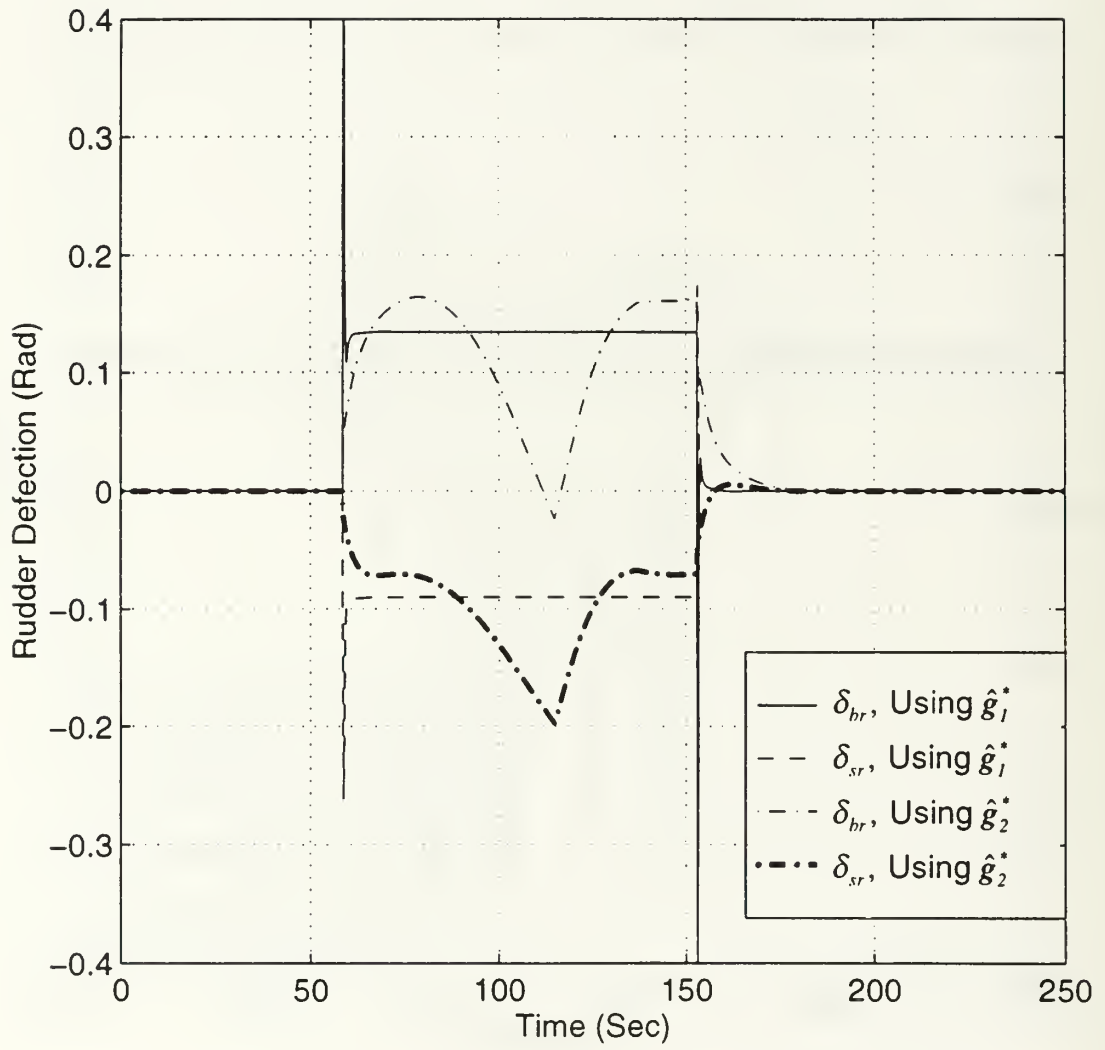
2.14 Rudder Deflection Angle vs. Time for Case 1, Using $\hat{g}_1^*$ and $\hat{g}_2^*$.
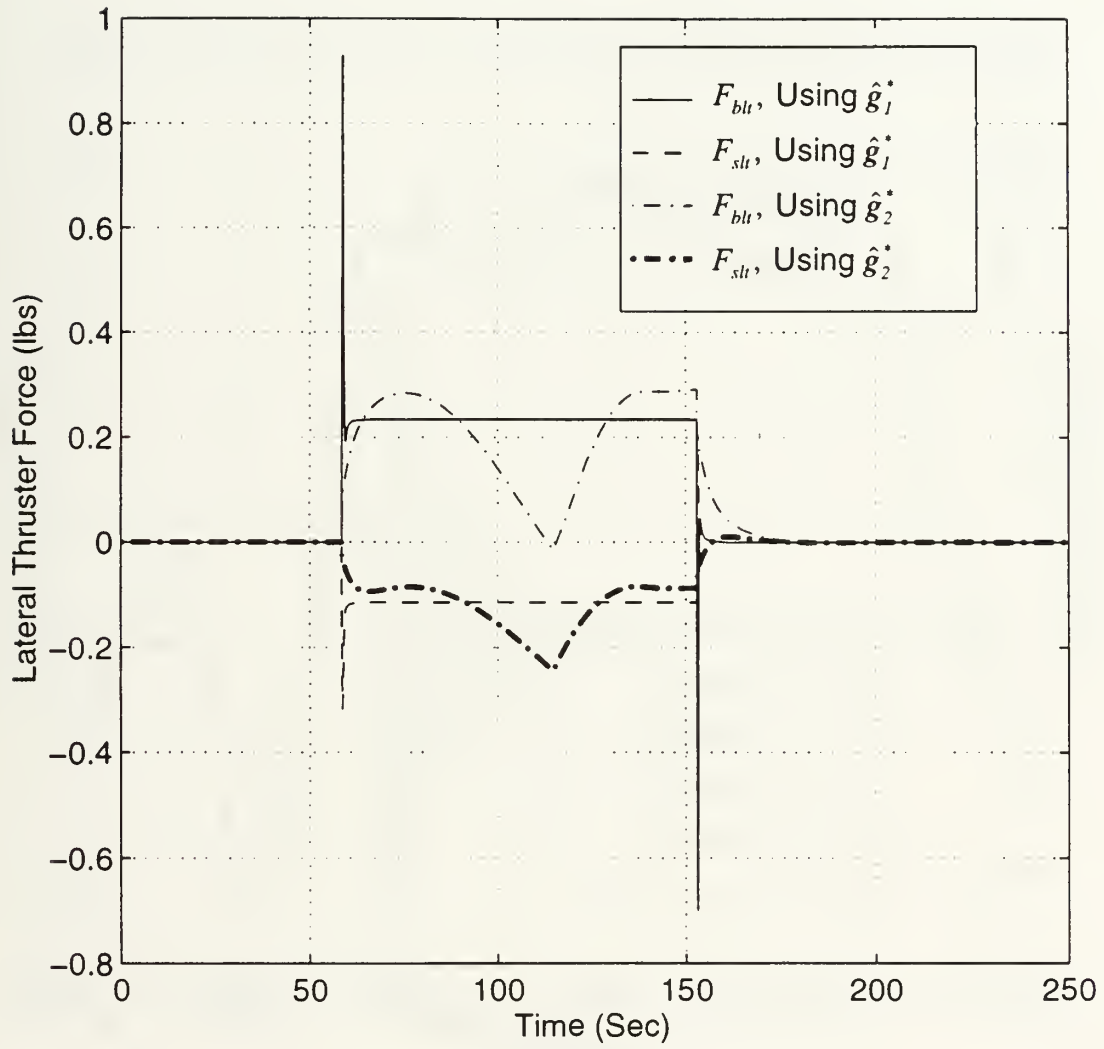
2.15 Lateral Thruster Control Force vs. Time for Case 1, Using $\hat{g}_1^*$ and $\hat{g}_2^*$.
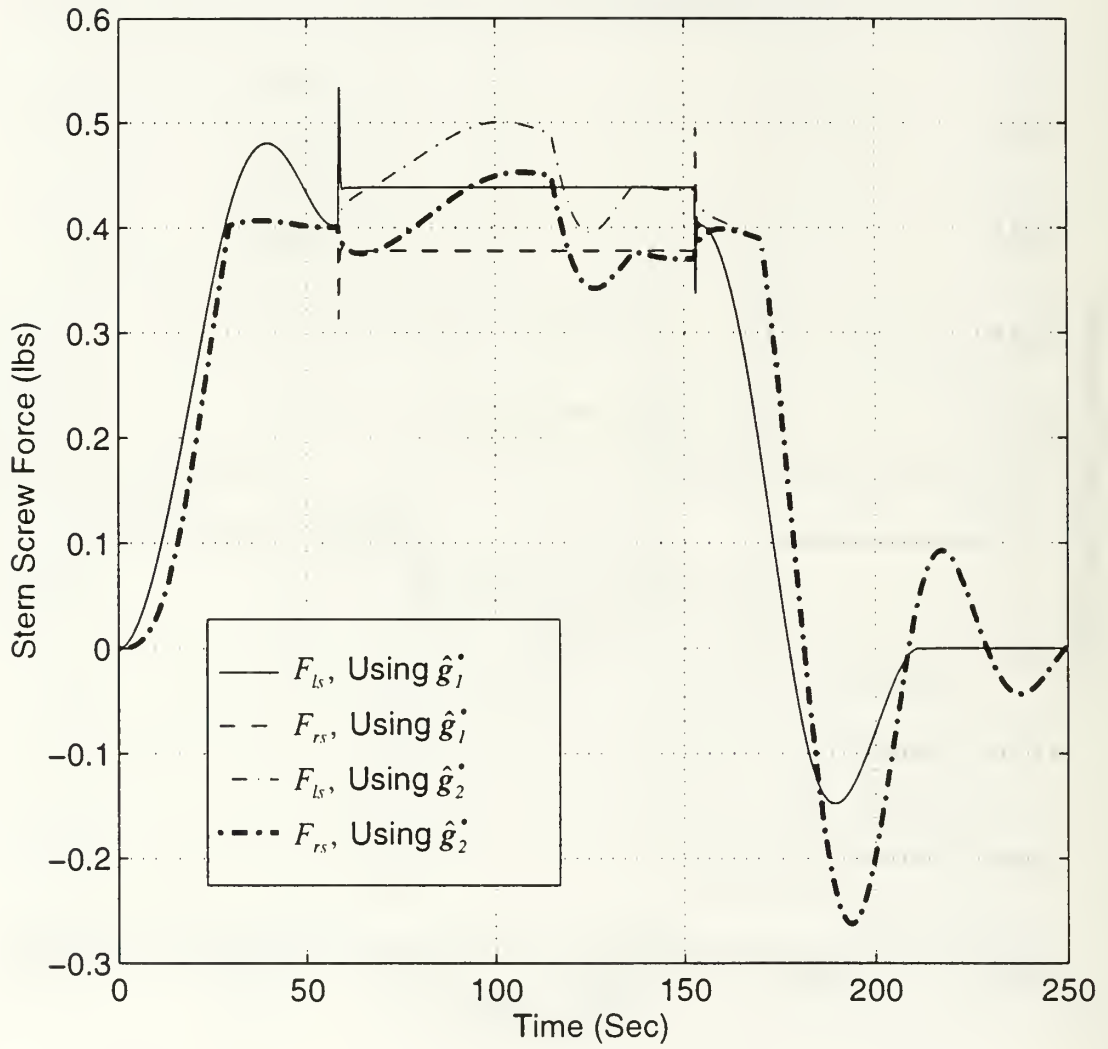
2.16 Rear Screw Control Force vs. Time for Case 1, Using $\hat{g}_1^*$ and $\hat{g}_2^*$.
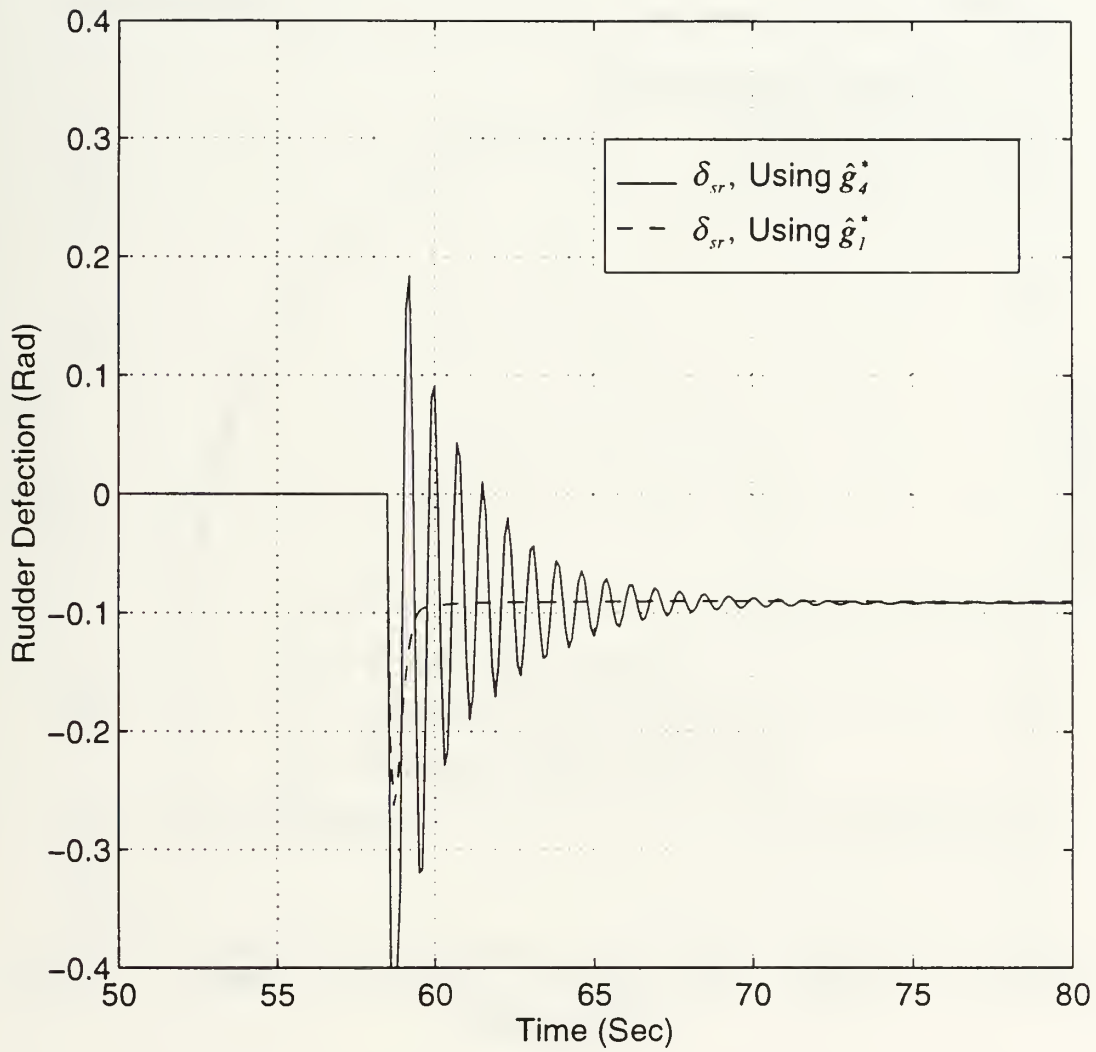
2.17 Rudder Deflection Angle vs. Time for Case 1, Using $\hat{g}_1^*$ and $\hat{g}_4^*$.
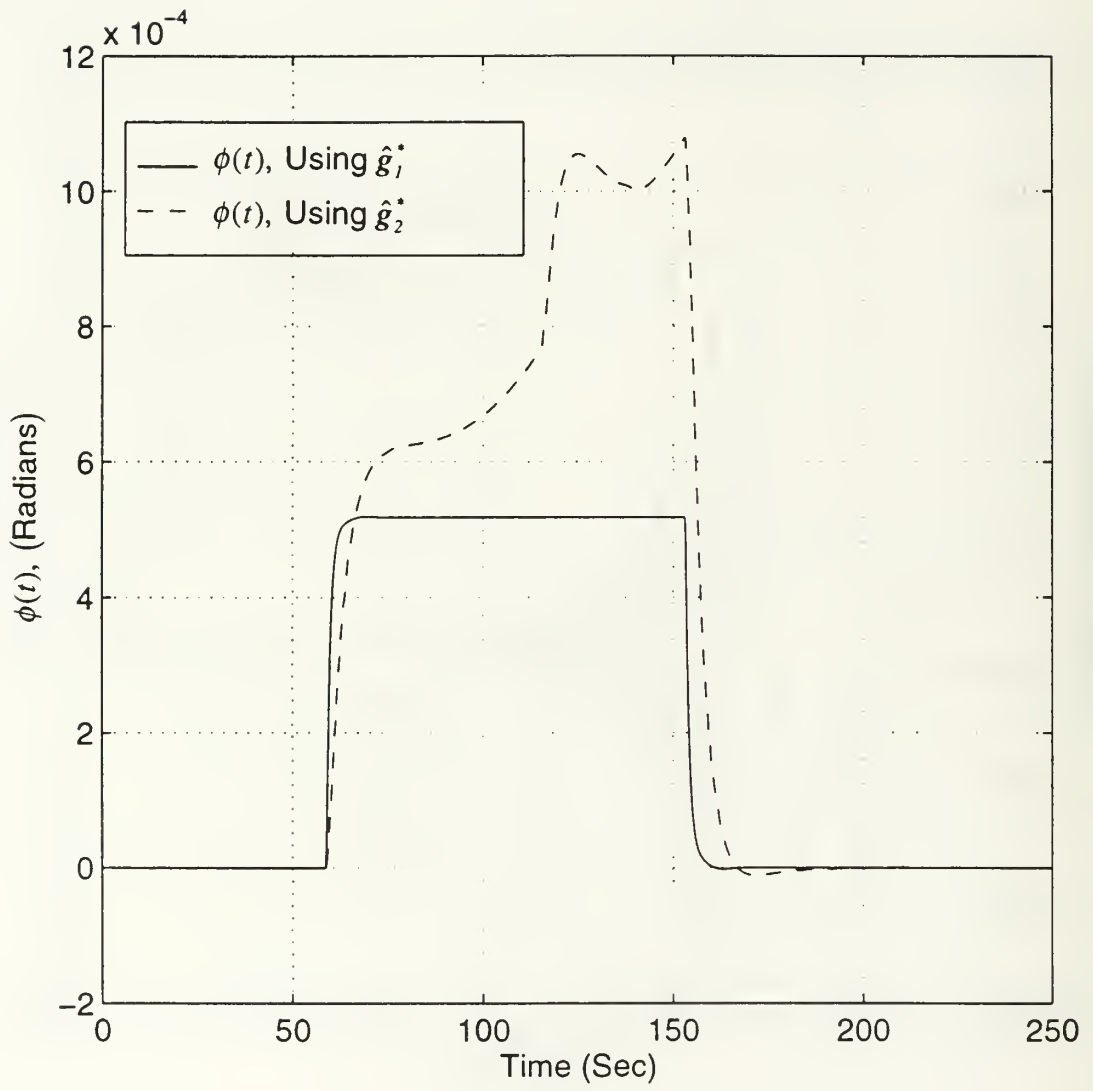
2.18 Roll Angle vs. Time for Case 1, Using $\hat{g}_1^*$ and $\hat{g}_2^*$.
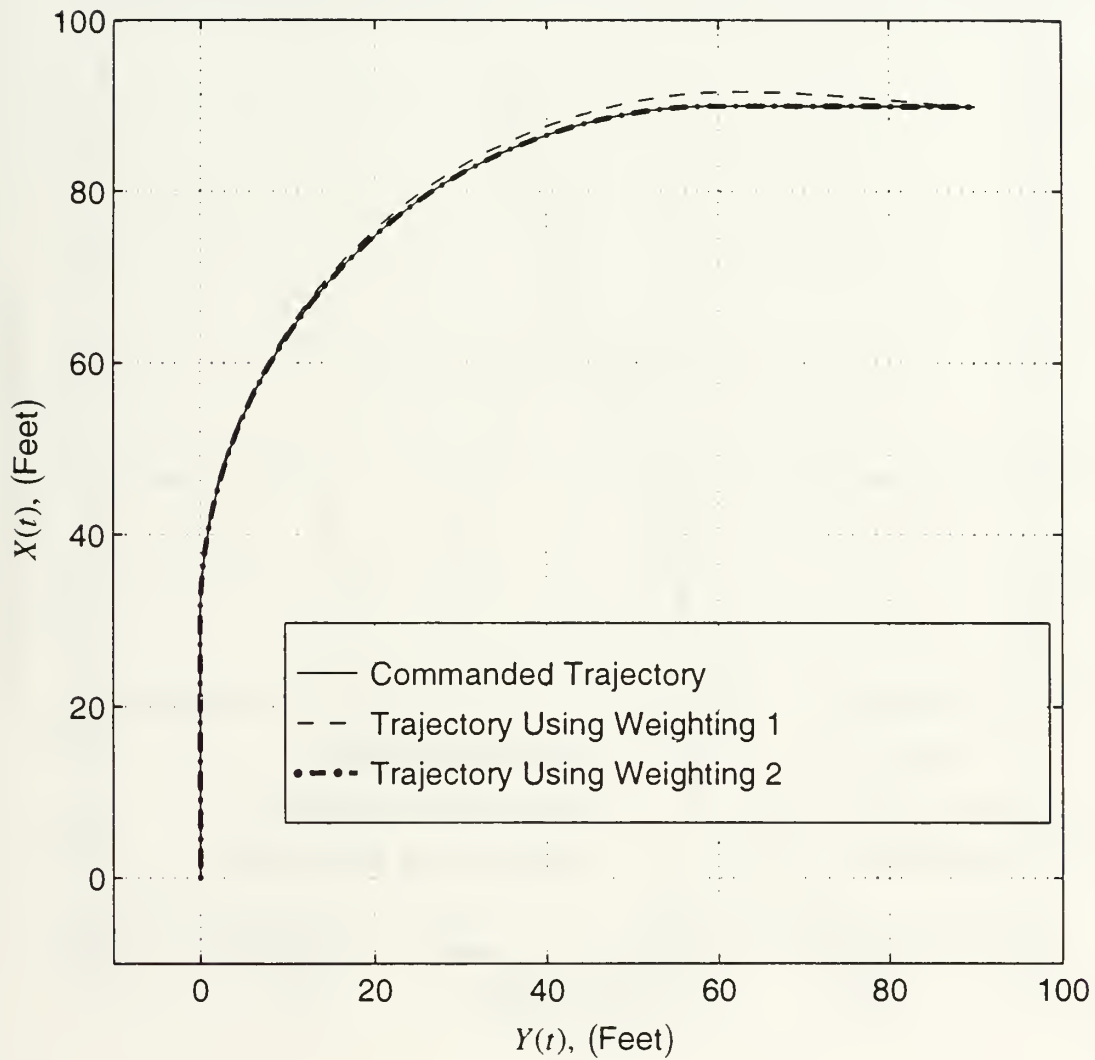
2.19 Position Response for Case 2, Weighting 1: (all $w_i = 1.0$, except $w_{F_{bh}} = w_{F_{slt}} = 0.0$), and Weighting 2: (all $w_i = 1.0$, except $w_{F_{bh}} = w_{F_{slt}} = 0.0$, and $w_{\delta br} = w_{\delta sr} = 0.1$).
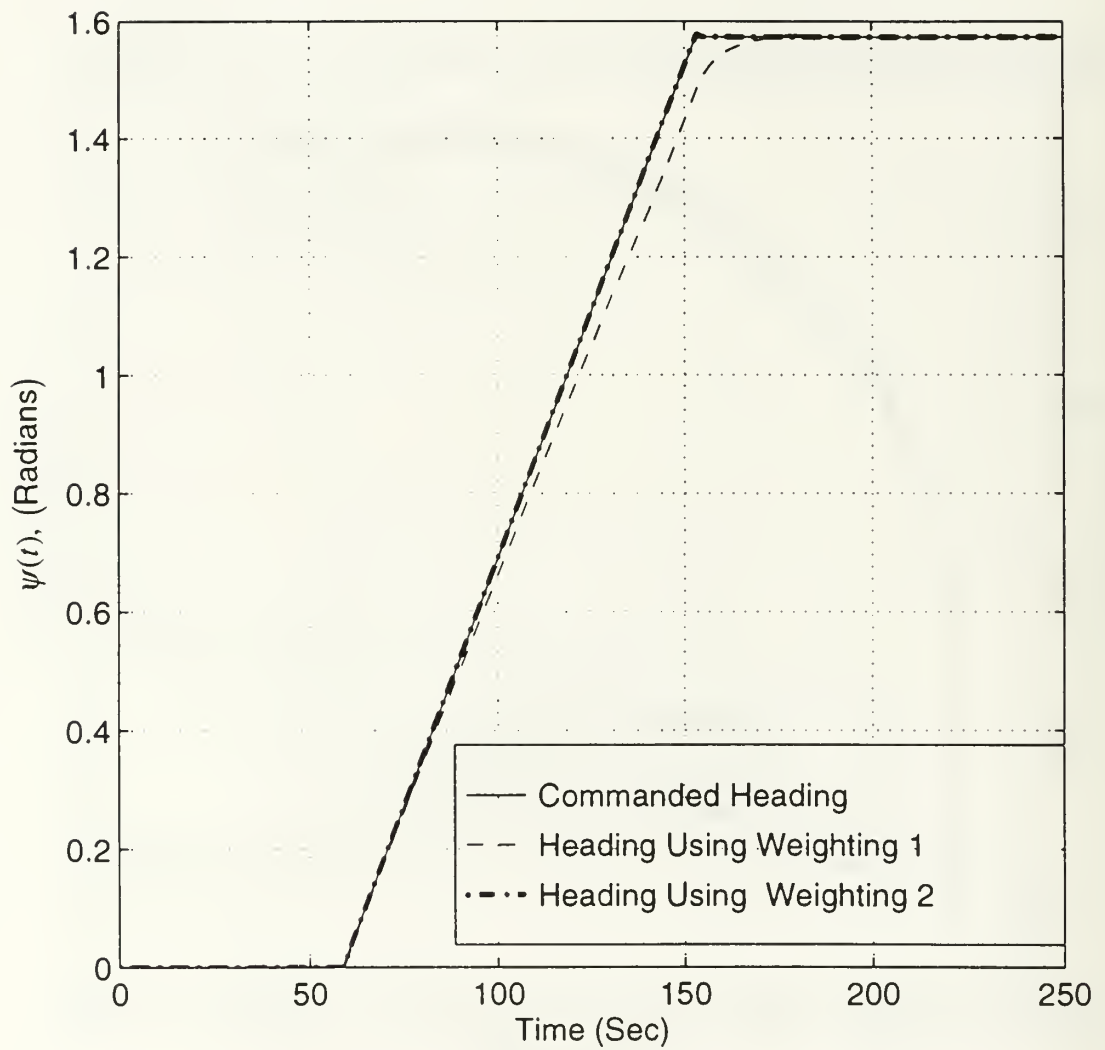
2.20 Heading Response vs. Time for Case 2, Weighting 1: (all $w_i = 1.0$, except $w_{F_{blt}} = w_{F_{slt}} = 0.0$), and Weighting 2: (all $w_i = 1.0$, except $w_{F_{blt}} = w_{F_{slt}} = 0.0$, and $w_{\delta_{br}} = w_{\delta_{sr}} = 0.1$).
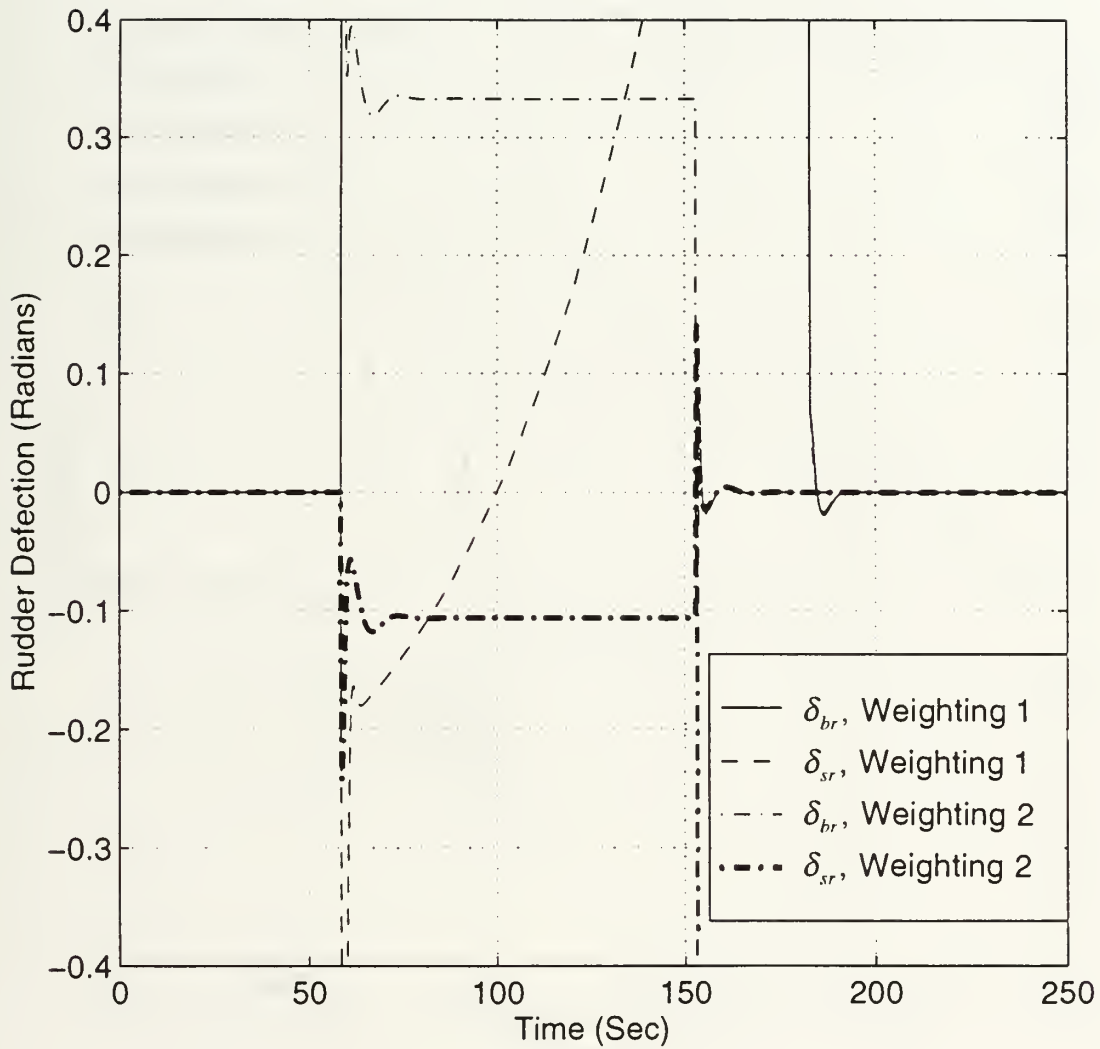
2.21 Rudder Deflection vs. Time for Case 2, Weighting 1: (all $w_i = 1.0$, except $w_{F_{blt}} = w_{F_{slt}} = 0.0$), and Weighting 2: (all $w_i = 1.0$, except $w_{F_{blt}} = w_{F_{slt}} = 0.0$, and $w_{\delta br} = w_{\delta sr} = 0.1$).
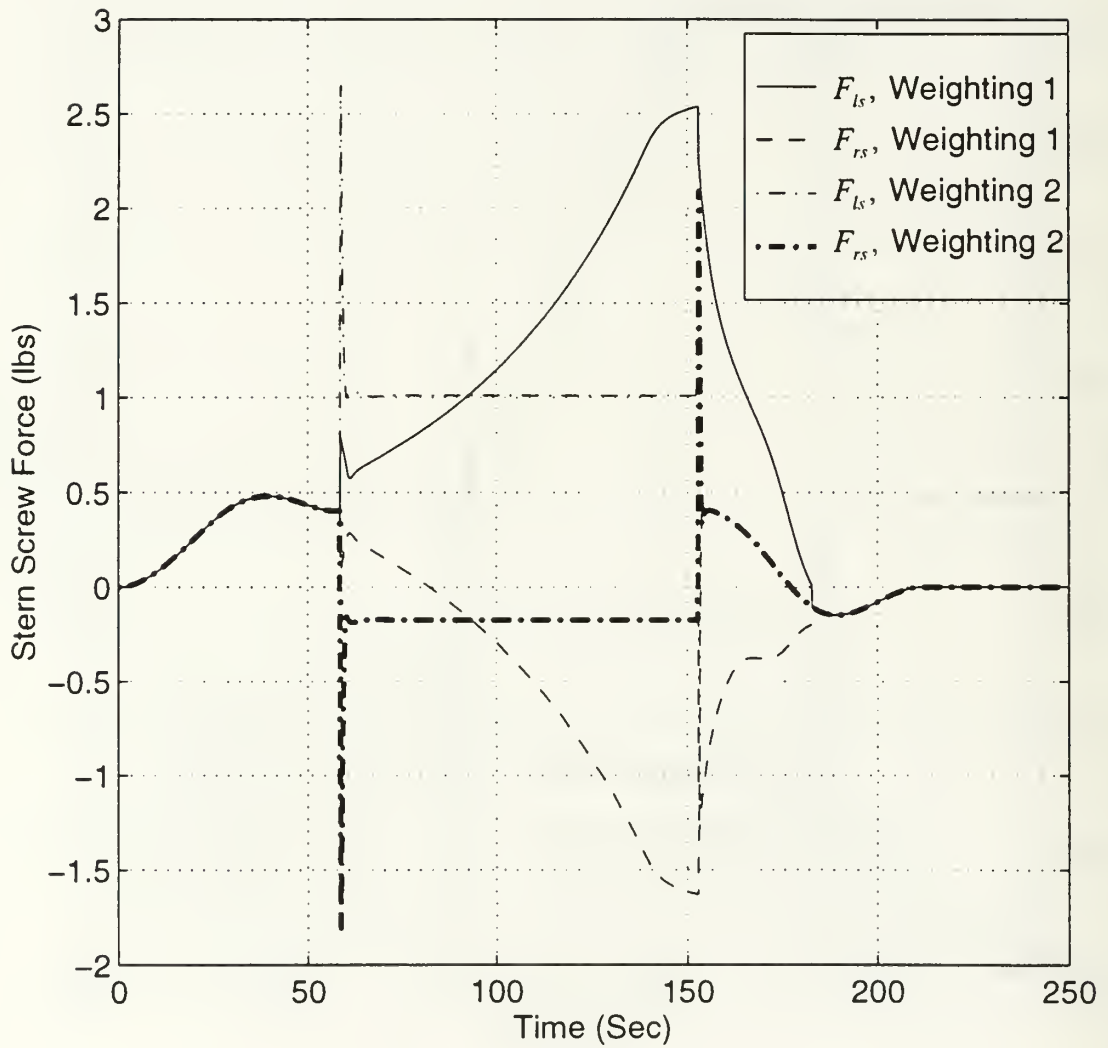
2.22 Rear Screw Control Force vs. Time for Case 2, Weighting 1: (all $w_i = 1.0$, except $w_{F_{blt}} = w_{F_{slt}} = 0.0$), and Weighting 2: (all $w_i = 1.0$, except $w_{F_{blt}} = w_{F_{slt}} = 0.0$, and $w_{\delta br} = w_{\delta sr} = 0.1$).
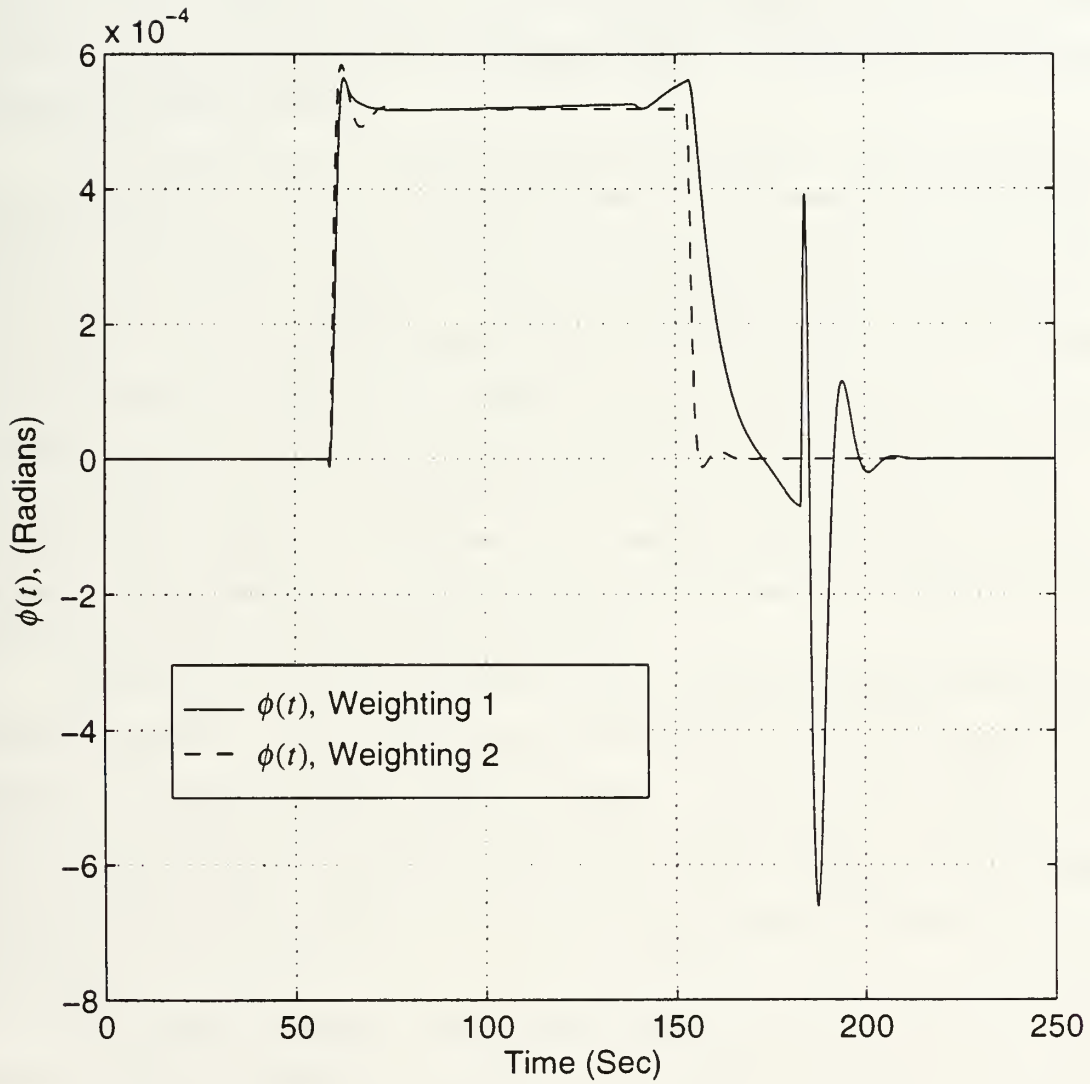
2.23 Roll Angle Response vs. Time for Case 2, Weighting 1: (all $w_i = 1.0$, except $w_{F_{blt}} = w_{F_{slt}} = 0.0$), and Weighting 2: (all $w_i = 1.0$, except $w_{F_{blt}} = w_{F_{slt}} = 0.0$, and $w_{\delta br} = w_{\delta sr} = 0.1$).

III. ROBOT CONTROL AND THE NPS PHOENIX VEHICLE

In the previous chapter, a general theory for continuous control of vehicle motion was presented. This alone is not sufficient to enable an autonomous vehicle to execute a complex mission involving multiple phases requiring different control modes. Vehicle motion is continuous, while the sequencing and coordination of different control modes is represented by discrete events.

The development of autonomous underwater vehicle control technology for underwater robots lies at the intersection of Discrete Event Systems (DES) and Dynamic Control of Continuous Systems (DCS) where system theory is well developed for each alone but not both acting together. It is not well understood how to formally evaluate the performance of these combined systems that are now being referred to as "Hybrid" control systems (Antsaklis and Passino, 1993). Saridis (Saridas, 1989) introduced the concept of "entropy" for a multidimensional performance index that could possibly be optimized for hybrid systems. Computer Aided Design of these systems has been proposed (Simon, et. al., 1993) using a rigid robot manipulator as an example, overcoming the lack of formal methods by using ORCCAD, a CAD package and the synchronous language "Esterel" developed especially for handling DES as automata.

Software architectures for underwater vehicles - a distinctly different problem from robotic manipulators - involve vehicle stabilization issues, and have been described and discussed in previous literature (Hall and Adams, 1992, Albus, 1988, Sousa, et. al., 1994), but without any experimental validation. Few detailed results have been quantified for the Odyssey class of vehicle (Smith and Dunn, 1994, Bellingham, et. al., 1994) although the Odyssey has performed under ice and demonstrated homing behaviors into a capture net.

Some Hybrid systems are predominantly DES and can be designed using state tables and finite state machines, or recently, Petri net methodologies (Cassandras, 1993). Others are predominantly continuous DCS with only a small component of discrete state logic for which stability theory, established optimal control techniques, and sliding modes are well suited (Friedland, 1986). "Hybrid", in the context of this dissertation, deals with the underwater robot control problem which is a true mix of DES and DCS for which new design techniques and evaluation methods are needed. In order to separate the functionality of the system we note that the control of the sequencing of a mission is a discrete event

system (DES) problem with the state transitions driven by conditions arising from the completion of robot tasks or by sensor based events, while the stabilization and control of vehicle motion to mission derived trajectories and or set points, is a traditional problem in dynamic control.

There is currently a very strong interest among researchers in the fields of artificial intelligence and robotics in finding a more effective means of linking high level symbolic computations relating to mission planning and control for autonomous vehicles to low level vehicle control software. Such research typically results in a proposal for a general software architecture, intended to solve a wide class of such problems. One of the first such proposals due to Saridis, who defined *intelligent control* as a research area lying in the intersection of artificial intelligence, control theory, and operations research (Saridis, 1983). He then further explicitly recognized three "basic levels" of control called the *organizational*, *coordination*, and *hardware control* levels, respectively. A more in depth discussion of this problem is given in (Healey, et. al., 1993).

The NASREM architecture (Albus, 1990) is at one end of the spectrum of Hybrid controllers and relies on a hierarchical system of planning. At the other end is the layered control with subsumption (Brooks, 1986) modified with discrete state coordination as in (Bellingham and Consi, 1991). The state transitions arise from completion of robot tasks while the specifications of a mission phase generates plans for vehicle motions in terms of either set points and control mode activations. It is the later that forms the basis for linking the mission control (DES) at the top (Strategic Level) to the vehicle control (DCS) at the bottom, (Execution Level) and is embodied in a middle (Tactical Level) set of control software functions.

This defines a tri-level software control architecture (- the Rational Behavior Model - (Byrnes, 1992, Byrnes, et. al., 1993(b))) comprising Strategic, Tactical, and Execution Levels. The three levels separate the software into easily modularized functions encompassing everything from logically intense discrete state transitioning through the interfacing of *asynchronous* data updates with the real time *synchronized* controller functions that stabilize the vehicle motion to set points or trajectory commands.

The distinguishing features that identify each level are

1. Strategic Level:

This level in the architecture uses a rule based language and is entirely Boolean - dealing only with the management of the discrete state transitioning required to perform the mission control. No numerical computations are done at this level and no memory is required except for the phase of the mission. In principle, it determines what needs to be done next.

2. Execution Level:

This level contains all the code functions that are required to stabilize the motion control of the vehicle to a set of commands that could be modes to be activated and servo set points where the servo control functions can be complex, even including command overrides for reflexive behavior and adaptive control features. Many robot controllers operate at this level only.

3. Tactical Level:

This level is a set of functions that are compiled as predicates in the Strategic Level Rules which open and close lines of communications between the Strategic Level and the Execution Level functions. They include the functions that gather data from the servo level and perform the necessary computations to determine if the robot tasks are completed, perform the navigational planning and replanning functions, the sonar computations, state of health diagnostic functions, and evaluates and sends appropriate set points and servo mode activation flags to the Execution Level. In this level, the computations are <u>numerical</u> but <u>asynchronous</u> with respect to time. The distinguishing feature between the Tactical and Execution Level software is that of the need for hard real time completion in the Execution Level and asynchronous completion in the Tactical Level.

In the controller architecture, developed in this work, the Strategic Level uses Prolog as a rule based mission control specification language. Other DES control system design techniques and implementation methods could be used, although, through the work of this dissertation so far it has been found that none is more convenient than using this existing available language. Prolog has the advantage of being an executable specification language which can run in real time as is demonstrated herein. The DES represented by the

mission specification could have been be translated in to C, Ada, C++, or other languages such as Esterel (Simon, et. al., 1993) and Coral (Silva, et. al., 1994). However, in our use of Prolog, the Prolog inference engine cycles through the predicate rules and in doing so, manages the state transition aspects of mission control so the need for formal, logical verification of the control specification disappears. It transitions the states in real time, and generally develops the commands (activations) that drive the vehicle through its mission. Error recovery procedures from failures in the mission tasks or the vehicle subsystems are handled as transitions to "error" states that ultimately provide commands to the servo level control for appropriate recovery action.

The Tactical Level, currently written in C is set of functions that are linked at compile time with the Prolog predicates and are designed to either return TRUE / FALSE in response to queries - these are distinguished by the prefix "Ask" in the Prolog rules - or to activate commands, distinguished by the prefix "Exec". These Tactical Level functions are also interfaced to the real time Execution Level controller using asynchronous communications and script type messages passed through an ethernet socket with TCP/IP protocol. Mission planning using this system provides a "complete plan". Success is guaranteed for every mission phase, either by proper completion, or by an abort.

The Execution Level controller is designed to command the vehicle subsystems appropriately for the mode flags and set points sent on the socket and to activate robot behaviors that correspond to those commanded. Communication from the Tactical Level to the Execution Level takes place through a single socket. By the design of this hierarchical control system, the Tactical Level runs asynchronously and retains the mission data file and the mission log file in global memory. It sends the command scripts to the Execution Level and requests data for the evaluation of state transitions. The architecture is a hybrid between the true hierarchical control of NASREM (Albus, 1990) and purely reactive of subsumption (Brooks, 1986) schemes. In this way, control of mission is retained, while reacting to unanticipated events is also enabled.

While earlier results at NPS were obtained by workstation simulations (Byrnes, 1993(a)), the major contribution of this section of the dissertation is that the tri-level software architecture has been implemented and experimentally verified with real time hybrid control tests using the NPS Phoenix vehicle. The experiments used all three levels of the control software active in real time. In what follows, the details of the vehicle hardware, as well as the controller hardware and software configurations are provided with discussion of the experimental evaluation of the controller through example. The example is

with respect to a simple mission using the hovering mode capabilities of the vehicle including coordination of the activation of a high frequency profiling sonar as part of the mission plan.

A.   THE NPS PHOENIX AUTONOMOUS UNDERWATER VEHICLE

The Phoenix is the third generation of Autonomous Underwater Vehicles at the Naval Postgraduate School. The first was a two foot long craft which had twin screws and rudder was used for parameter identification and control research. There was no onboard computer or batteries but was attached to a tether which supplied power and control commands from an external PC (Healey, et. al., 1989, Brunner, 1988). This is now referred to as the AUV I and is no longer in operation at this time.

In order to experiment with autonomous control a new vehicle was built named the AUV II (Healey and Good, 1992) and was much larger, 6.5 feet in length, weighing 435 lbs and carried it's own batteries and computer. Many untethered tests of this vehicle were carried out in the NPS swimming pool during 1990 and 1991 (Healey, et. al., 1991), in which waypoint control, steering, diving, and speed control experiments were performed and control laws verified. A single process in the onboard computer controlled all maneuvers and was written in the C programming language. All data collected from the onboard sensors after a run were uploaded using a serial link connected through the access hatch.

In February 1993, the vehicle interior was accidentally flooded and an extensive rebuild was undertaken. The vehicle was again operational by October of that year and renamed the "Phoenix". Although Phoenix is the same size and weight, and possesses the same propulsors as the AUV II, it is a far superior system in the area of computational power and data communications. The internal components of the Phoenix are shown in Figures 3.1, 3.2 and external views are shown in Figure 3.3.

The new vehicle contains two computer systems, one the original Gespac with the OS9 real-time operating system and the second a Sun Microsystems Voyager SPARC station using the Solaris (Unix) operating system. Both computers communicate using thinwire ethernet which also extends outside of the vehicle to workstations on the shore either through a hard wire connection or using a radio ethernet link. The radio link can be either configured to have the antenna mounted to a float for uninterrupted communications

while submerged or mounted directly to the hull for better streamlining but only able to upload/down load information while surfaced.

Most of the experimentation done for this dissertation was with a Sun Microsystems Britelite portable SPARC workstation, in place of the Sun Voyager, operating outside the hull. It was connected to the vehicle with a thinwire ethernet cable using a removable water-proof connector.

1. Propulsion Systems and Sensors

Six propellers are used for maneuvering the vehicle: two open screws located at the stern, two vertical and two lateral cross-body thrusters. All are powered from 24 Vdc electric motors, computer controlled using pulse width modulated servo amplifiers. Located at the back of each motor is an optical encoder which is used to measure angular speed The encoder generates a square wave pulse which is counted by the controlling computer.

a. Stern Screws

Each stern screw is a brass four blade propeller measuring 4.25 inches in diameter. These are connected directly to the motor shafts with no reduction and are each capable of delivering up to 5 pounds of bollard pull force (Saunders, 1990, Cody, 1992). Both may be independently controlled and are able to spin in either direction.

b. Cross-Body Thrusters

The cross-body thrusters consist of a 3 in. ID aluminum tube with a centrally located 4 blade brass propeller shown in Figure 3.4. A spur gear is mounted around a 3 inch diameter propeller and driven by a pinion connected to a 24 Vdc motor giving a 2.5:1 gear reduction. The only seal is located around the motor shaft which leaves the propeller, ring gear, and pinion "wet". The propeller shaft is supported by three equally spaced struts at each end. These devices were designed and constructed at NPS since there were no commercially available cross-body thrusters this small. The twist of the propeller blade is symmetric enabling bi-directional operation delivering approximately 1.0 pound of bollard pull force in either direction (Healey, et. al., 1995).

All propulsors are powered by 24 Vdc Pitman PITMO DC Model 14202 series electric motors. Each is equipped with Hewlett-Packard Model HEDS-5000 series optical

66

encoders attached to the shaft at the back of the motor to measure angular rate. The encoders use a 500 aperture code wheel which is between an LED emitter and integrated circuit detector. The detector converts the signals to a square wave output which is measured by the computer.

c. Control Surfaces

Eight control surfaces are present on the vehicle, four rudders and four vertical control planes. Each surface is powered by servo motors used by radio controlled model aircraft (Airtronix Model 94501). Since the surfaces are independently powered, the rudders are electronically coupled so that when a command to turn is given, the upper and lower planes will rotate together. The stern rudders will rotate one direction, and the bow rudders in the opposite direction which provides higher turning performance. This approach is also employed with the vertical control planes as shown in Figure 3.5.

d. Sensors

A full description of all vehicle sensors are given in Appendix D. It covers the gyroscopes, speed sensor, short baseline navigation system, and GPS components. An expanded discussion of the Tritech ST725 and ST1000 design and operation is given in Chapter V.

2. Computer System

A Gespac 68030 microprocessor is used for running the vehicle hardware control software. It uses the OS9 operating system which enables real time execution of the software. Along with the processor are eleven other Gespac boards interfaced with the sensors, motors, control surfaces, networks, etc. A brief description of each board and it's primary interfacing function is given in Appendix D.

The Phoenix has been designed to operate for at least 3 hours on a single battery charge. The following section describes The major components of the electrical power system are described in Appendix D.

3. Hull Integrity

The vehicle has four access plates located on top of the hull, which are held in place by machine screws and are sealed against leakage using a marine grade putty. To ensure against leakage through the plate seams, or any other point, the interior is pressurized with air to approximately 0.5 lbs above atmospheric. This procedure serves two purposes, one is that if the hull is not leak-proof, the pressurized air will escape through the openings and can be detected by applying soapy water around the seams to check for bubbles. The second reason for pressurization is that if a small opening develops while submerged, the air will seep out before water enters the hull. The escaping air will form bubbles that will provide a visual queue that a leak is immanent once the internal air pressure drops below the hydrostatic pressure. Detection of the bubbles allows for corrective action to be taken.

B.   THE CONTROL NETWORK EXPERIMENTAL SETUP

The control system, illustrated in its simplest form in Figure 3.6, is currently implemented in hardware using three networked processors. All Execution Level software is written in "C" and runs on a Gespac M68030 processor in a separate card cage inside the boat. Connected in the same card cage is an ethernet card and an array of real time interfacing devices for communications to sensors and actuators indicated in the details of Figure 3.7. The Execution Level control code containing a set of functions in a compiled module called **"exec"** is downloaded first, opening the communications socket on the Gespac side. This process block sleeps until a network connection is requested from the higher level controller in the Sun SPARC.

The network configuration for the results presented is shown in Figure 3.8. It consists of five nodes connected by thinwire ethernet. The Gespac computer inside the vehicle is connected to the other systems using a water-proof through-hull connector. The DOS PC is used only for OS9 cross-compilation of C code usually developed on an SGI Elan or equivalent system. The Elan is also used for displaying in real time the sonar data obtained in the Execution Level for missions using the ST1000 sonar. To further facilitate software development and transfer, a wireless ethernet unit is available for Internet connections outside of the laboratory.

C. THE TRI-LEVEL CONTROL SYSTEM

1. Vehicle Primitives

In previous work (Healey and Marco, 1992(a)), waypoint following in a transit phase of a mission was demonstrated in a swimming pool test area where stable behaviors of the vehicle were demonstrated including

a) Forward Speed Control,
b) Fin_Steering
c) Fin_Depth_Control
d) Waypoint_Following
e) Bottom_Following, and
f) Obstacle_Avoidance.

These control functions were implemented with a)-c) and f) running simultaneously, but subsumed by the guidance laws implemented in d); and, with c) subsumed by e). The control laws implemented have been based on PD, and Sliding Mode methods as explained in (Healey and Marco, 1992(b)).

Control laws for these functions are readily implemented entirely in the Execution Level with digital control algorithms running at 0.1 sec. update rate. Now, more complex functions have been enabled using active control of the cross-body thrusters and sonar. These are,

g) Submerge_and_Pitch_Control
h) Heading_Control
i) Longitudinal_Positional_Control
j) Speed_Control using command generators
k) Lateral_Positional_Control
l) Center_Sonar
m) Ping Sonar (Mode 0)
n) Ping and Rotate Sonar Clockwise (Mode +1)
o) Ping and Rotate Sonar Counter-Clockwise (Mode -1)
p) Ping and Rotate Sonar Through a Sector (Mode 2)

q) Initiate_Filter_For_Sonar_Range

(Needed For Smoothed Range and Range Rate Estimation)

r) Reinitialize_Filter

s) Track_Target

t) XY_Psi_Control

Most of these functions need a given subset of the actuator system to be active under the operation of either an open loop command or a feedback control law. Some of the functions use orthogonal sets of actuators and may be activated without conflict. Some use the same actuators to control different functions and thus control laws may be additive. This means, for example, that vertical thrusters may be used via control laws to control depth as well as pitch, and lateral thrusters to control heading as well as lateral position and side slip speed. In combination with propulsion motors, most functions including Submerge_and_Pitch_Control, Longitudinal_Speed_Control and Longitudinal_Position_Control, as well as Heading_Control, may now be commanded reliably. Heading_Control, Submerge_and_Pitch_Control, and virtually any multiple combination of a) to t) above are available to the extent that they do not cause a conflict of actuator control or sensor usage.

2. Orthogonal Behaviors

Orthogonal behaviors are defined as those control behaviors that use non-interacting subsets of actuators. Even though each may use some common sensory data, orthogonal behavior control functions may be activated simultaneously without conflict. An example would be Heading_Control (using lateral thrusters), Longitudinal_Position_Control (using the propulsion motors) and Center_Sonar. Non-orthogonal behaviors use intersecting sets of actuators for control of different error functions and thus control laws can be built up from linear combinations of individual control laws - as used for combined heave and pitch control using vertical thrusters.

Activation of orthogonal behaviors are instituted using a script composed of flags and set points that are a way of communicating between Tactical Level C functions and the real time control loop of the Execution Level. At each pass through the loop, a read is made from the communications socket and an if-else structure is used to determine which set of sensors, actuators and control laws are to be activated during the computation cycle. The

70

same technique is used to flag the activation of sonars, and filtering actions, and similarly for flags to indicate which data streams are to be written in return.

3. Reactive Behavior Implementation

Reactive behavior in the controller is handled in three ways similarly to that done in the ORCCAD design (Simon, et. al., 1993).

1. In the Execution Level control loop through <u>command overrides</u> following a sensor read, as, for instance, a new obstacle detection requiring an emergency surface or obstacle avoidance (flinch) response,

2. at the Tactical Level, reactive error recovery can be handled by resetting key parameters associated with control signal evaluations. An example is the resetting of a control gain or the inclusion of integral control if a particular error function cannot be stabilized,

3. reactive behavior is also handled at the Strategic Level for catastrophic faults by transitioning to states that command fatal error recovery procedures. An example is to surface if, for example, a particular mission phase is not completed after a pre-specified time out and all other techniques have been exhausted.

In the results described here, reactive behavior is built in at the Strategic Level by time and space traps using time out calls. If an allocated time is exceeded, the mission phase fails and the vehicle is commanded to surface. Control overrides are built into the Execution Level to surface the vehicle if battery power is too low or if a leak is detected.

4. Strategic Level

The Strategic Level Prolog rules are compiled and linked together with the supporting Tactical Level "C" language functions into a single executable process called **"Mission_Control"**, that is run in a Sun Microsystems SPARC 4 laptop computer and linked through an ethernet socket to the Gespac processor (socket "A" in Figure 3.6).

71

Starting "**Mission_Control**" enables communications between both SPARC and Gespac processes. All vehicle control functions, with the exception of the transmission of sonar imaging data, communicate by message passing through that socket. The Strategic Level Prolog is divided into two parts - first the generic mission controller in Figure 3.9 (Marco, et. al., 1996), and secondly, the phase level detail in Figures 3.10-3.12. For clarity, the higher level rules are shown in bold type, the C functions in italics, and any user defined or built-in Prolog predicates are in plain text. The rule set is first compiled and then run in the interpreter by entering the query "execute_mission.". The example mission outlined in the figures below consists of three phases: *vehicle initialization; submerging to a specified depth while maintaining a heading command; and sweeping the profiling sonar head 360 degrees while still controlling to depth and heading.* This is a deliberately simplified mission for clarity of explanation. Many more complex missions have been performed utilizing the principles outlined here.

A complete listing and description of each C function callable by Prolog is in Appendix E.

```
done :- current_phase(mission_abort).

done :- current_phase(mission_complete).

execute_mission :- initialize_mission, repeat, execute_phase, done.

initialize_mission :- ood('start_networks',X), asserta(current_phase(1)), asserta(complete(0)),
                      asserta(abort(0)).

 execute_phase :- current_phase(X), execute_phase(X), next_phase(X),!.
```

Figure 3.9 Prolog for the Generic Mission Controller.

```
execute_phase(1)  :-  exec_init_vehicle(X),  exec_start_timer(X),  repeat,
                          phase_completed(1).

phase_completed(1) :- ask_init_vehicle_done(X),  X==1,asserta(complete(1)).

phase_completed(1) :- ask_time_out(X), X==1, asserta(abort(1)).


next_phase(1) :- complete(1), retract(current_phase(1)), asserta(current_phase(2)).

next_phase(1) :- abort(1), retract(current_phase(1)), asserta(current_phase(mission_abort)).
```

Figure 3.10 Prolog for Phase 1 (Initialize Vehicle Systems).

```
execute_phase(2) :- exec_get_setpoints(X),   exec_submerge(X), exec_rotate(X),
                    exec_start_timer(X),   repeat,   phase_completed(2).

phase_completed(2) :- ask_depth_reached(X), X==1, ask_heading_reached(X), X==1, asserta(complete(2)).

phase_completed(2) :- ask_time_out(X), X==1, exec_surface(X), repeat, ask_surface_reached(X), X==1,
                    asserta(abort(2)).

phase_completed(2) :- ask_sys_problem(X), X==1, exec_surface(X), repeat, ask_surf_reached(X), X==1,
                    asserta(abort(2)).


next_phase(2) :- complete(2), retract(current_phase(2)), asserta(current_phase(3)).

next_phase(2) :- abort(2), retract(current_phase(2)), asserta(current_phase(mission_abort)).
```

Figure 3.11 Prolog for Phase 2 (Submerge to Depth and Rotate to Heading).

```
execute_phase(3) :- exec_get_setpoints(X),    exec_submerge(X), exec_rotate(X),
                    exec_set_sonar_mode(X), exec_start_timer(X),  repeat,
                    phase_completed(3).

phase_completed(3) :- ask_sonar_sweep_complete(X), X==1, asserta(complete(3)).

phase_completed(3) :- ask_time_out(X), X==1, exec_surface(X), repeat, ask_surface_reached(X), X==1,
                    asserta(abort(3)).

phase_completed(3) :- ask_sys_problem(X), X==1, exec_surface(X), repeat, ask_surface_reached(X), X==1,
                    asserta(abort(3)).


next_phase(3) :- complete(3), retract(current_phase(3)), asserta(current_phase(mission_complete)).

 next_phase(3) :- abort(3), retract(current_phase(3)), asserta(current_phase(mission_abort)).
```

Figure 3.12 Prolog for Phase 3 (Sweep Sonar).

Referring to the Prolog code in Figure 3.9, the rule head "execute_mission" will be satisfied, and hence the mission completed if the predicates "initialize_mission", "execute_phase", and "done" all evaluate TRUE. Once "ood('start_networks',X)" completes, phase 1 is asserted to be the current phase (i.e. set to TRUE), then the entire rule body of "initialize_mission" is evaluated as TRUE. This action enables the control to enter a repeat loop which executes the predicate "execute_phase" attempting to evaluate each predicate current_phase(X), execute_phase(X), and next_phase(X), as X assumes the values 1 through 3 in sequential order. This particular mission has only three phases, but is expandable to include as many phases as desired.

Each phase includes a "repeat" predicate so that the rules for phase completion are evaluated repetitively until one of the rules is TRUE. With the exception of vehicle initialization, each phase can terminate in one of three ways:

      1. Normal Completion.   Next Action: (Execute Next Phase)

      2. Abort Due to Time Out.   Next Action: (Surface Immediately)

      3. Abort Due to System Problem. Next Action: (Surface Immediately)

If phase X completes normally, "complete(X)" is asserted and X is incremented by one to execute the next phase. Normal completion usually indicates that a commanded set point or

task has been accomplished and the vehicle is ready to start the next phase. In the case of phase 2, this means that the commanded depth and heading has been achieved. If a time out or a system problem occurs, the vehicle is commanded to surface immediately and a mission abort for phase X is asserted after the surface is reached. A time out indicates that a set point or task is taking too much time to complete and with our current version of error recovery, the mission phase is aborted and also the entire mission. System problems can cover a variety malfunctions, sensor failures, thruster failures, or any type of hardware problem, which for this work, are assumed to be catastrophic requiring an entire mission abort.

After each phase executes, the predicate "done" is evaluated. If the next phase is commanded, "done" fails and the cycle continues, if however a mission abort is asserted or the mission completes, "done" is satisfied and "execute_mission" evaluates TRUE and the entire mission is finished.

Tasks that are required to be performed in successive phases are commanded again as shown by the calls "..*exec_submerge(X)*, *exec_rotate(X)*, .." which appear in Phase 2 and 3 (Figures 3.11 and 3.12). In this way new set points can be entered for each phase. Generally control mode commands are left active until changed although "kill" rules can be used to stop actions such as filters etc.

5 . Tactical Level

The Tactical Level of the control system contains all the C functions that are compiled as predicates in the Strategic Level rules, and performs the computations upon which the vehicle commands and transitions are based. Additionally, a second SPARC process called the "**Sonar Manager**" is opened which runs asynchronously in the SPARC and with equal priority to the "**Mission_Control**". This process is linked through a separate socket ("B" in Figure 3.6) to the Gespac for the purpose of the reception and handling of sonar imaging data. The "**Sonar Manager**" captures data that is sent out from the Execution Level as soon as it has been acquired, and then processes and passes the data to be displayed on the IRIS Graphics workstation for visualization purposes.

The introduction of the additional process called "**Sonar Manager**" and it's separation from the "**Mission_Control**" Tactical Level functions has been found to be important and a necessary first step toward a more general Concurrent Tactical Level that

was foreseen by the earlier RBM architecture (Byrnes, et. al., 1993(b)) and explained recently by (Kwak and Thornton, 1994). The need for concurrency of multiple processes lies fundamentally with the fact that sonar data is obtained *asynchronously* with bounded but unknown latency and the servo control functions cannot wait for the sonar port data to arrive. While it is perfectly normal to send control set point commands *asynchronously* to stable control loops, waiting for sonar returns could hold up the servicing of the inner servo loop commands to actuators. Thus in this solution to the Hybrid control problem, the additional "**Sonar Manager**" process always reads the socket onto which sonar data is written so that it is immediately free for another sonar write without delay, and the servo loop is made independent of direct involvement with the sonar. As an unpleasant alternative, this research has shown that without the "**Sonar Manager**", all the Tactical Level functions would have to be modified to include a check to read sonar data. This would have been a cumbersome addition of much unnecessary code writing. One possible alternative, yet to be fully implemented, is to allow the primary acquisition of sonar to communicate directly and asynchronously in the Tactical Level. However, the drawback of that approach would be the difficulty of obtaining high rate sonar updates in servo control functions.

a. Transition Criteria

Most control phase transitions of the Phoenix are event based, meaning that a certain set of criteria must be met in order for a transition to occur. A common example of this is when a position set point is sent to the vehicle controllers and reached. A method of determining whether the vehicle has indeed reached this point must be programmed into the control logic. Measuring the position error alone and declaring the maneuver complete when this error is small is not sufficient. This is because the vehicle could be overshooting the commanded position and simply passing through the set point. Not only must the position error be small but the rate error must also be small. This dual criteria can be expressed mathematically as a positive definite, linear combination of the position error e and the position rate error $\dot{e}$. We use,

$$\kappa_k = w_e |e_k| + w_{\dot{e}} |\dot{e}_k| \qquad (3.1)$$

where w_e and w_e are positive weights for the position and rate errors respectively. Eqn. (3.1) may be quantized, which allows a minimum value of κ, denoted κ_0, to be specified defining a threshold for the combination of errors which can be set relatively large when precision control is not required or low for extremely precise positioning. Once κ drops below κ_0, the maneuver is declared complete and a transition to the next control phase may occur.

When noisy sensors are used, the noise prevents κ from settling enough to determine an accurate measurement for the transition, and the use of Eqn. (3.1) alone has shown to be unsatisfactory. The signal can be smoothed by filtering κ through a first order digital filter of the form

$$\kappa_{f(k+1)} = e^{-T/\tau}\kappa_{f(k)} + (1 - e^{-T/\tau})\kappa_k \tag{3.2}$$

where κ_f is the filtered form of κ, τ is the time constant of the filter, and T is the sampling time. The condition for transition can be shown in Figure 3.13, which indicates that the signal for transition, s, is 1 (TRUE) for $\kappa_f < \kappa_{f_0}$ or 0 (FALSE) for $\kappa_f > \kappa_{f_0}$. As an example, the function "ask_depth_reached(X)", performs the calculations above and returns s. Other dynamic error and time based signals are computed similarly.

b. Mission Data File

Each phase with the exception of initialization has associated with it particular mission parameters. These tend to be set points, phase time-outs, etc. This information is obtained by the Tactical Level through the use of a mission data file which contains numerical values for each phase. The file is named "mission.d" and has the following format for each phase.

Timeout (Sec)	X_{com}(Ft)	Y_{com}(Ft)	Z_{com}(Ft)	ϕ_{com} (Rad)	θ_{com}(Rad)	ψ_{com}(Rad)
	κ_{f0_X}(Ft)	κ_{f0_Y}(Ft)	κ_{f0_Z}(Ft)	κ_{f0_ϕ}(Rad)	κ_{f0_θ}(Rad)	κ_{f0_ψ}(Rad)
Sweep_Time (Sec)		Sweep_Mode	ψ_{sd}(Deg)	ψ_{sw}(Deg)		
	CM_X	CM_Y	CM_Z	CM_ϕ	CM_θ	CM_ψ
	ΔT_X(Sec)	ΔT_Y(Sec)	ΔT_Z(Sec)	ΔT_ϕ(Sec)	ΔT_θ(Sec)	ΔT_ψ(Sec)

$$\bullet$$
$$\bullet$$
$$\bullet$$

At time of initialization the entire file is read and each time the predicate *exec_get_setpoints(X)* is executed, the data for that phase is obtained. For each phase, totaling "N_phases", a vector of position and angular set points is defined along with the minimum value for the filtered error, κ_{f0}, for each control direction. The values Sweep_Mode, ψ_{sd} (Deg) and ψ_{sw} (Deg) are initial sonar control parameters which may or may not change during the phase depending on the mission specification. The "CM" (Control Mode) values are either 0, for step input control or 1, which indicates a command generator should be used. If command generators are used, the time to complete the maneuver may be specified using the ΔT's in the last row (see Appendix B). With this file, many combinations of control can be selected. For example, any phase may use a command generator for depth control but not for heading, or this control mode may be specified for both. Any combination is possible provided the particular degree-of-freedom is controllable. For the case of submergence only, with no provision for the positions X and Y to be measured or controlled in the mission specification, the values of X_{com} and Y_{com} are meaningless and any value may be entered. Common practice is to simply enter 0.0 for the terms not pertinent to a particular phase.

6. Execution Level

The structure of the Execution Level software illustrated by Figure 3.7 indicates that it is composed of software at the hardware interface (software drivers) as well as software for vehicle control. It is the primary set of software that performs real time data acquisition and servo control. After initialization of power systems and sonars, and the basic driver settings, the PIA card pins that control the on/off features of power supplies, thruster

power, screw power, and sonar power, a simple timing loop is entered and reentered at a fixed update rate (in our case 0.1 sec.). During each loop the following takes place and is shown in Figure 3.14.

1. Read socket "A" for behavior based mode command flags and control set points (SOCKET READ BLOCK).

2. Read all sensors (READ SENSORS BLOCK).

3. Select appropriate C code control functions for computing and sending control values to actuators, using an if-else structure for distinguishing the commands and send signals to the sonars to ping and rotate, etc. (CONTROL BLOCK).

4. Write selected data to local ramdisk/memory or to sockets "A" or "B" as appropriate (DATA RECORDING).

5. Check time for any time based events and wait for the next timing interrupt to maintain integrity of the digital control loop (TIMER WAIT).

Specific control laws as built into callable modules of code are easily selected according to the communication flags, provided that they exist in the first place.

The hardware components of the Phoenix are controlled and interrogated using Execution Level C language functions. Each either reads information from the sensors or writes commands to the vehicle actuators. The function descriptions are described in detail in Appendix F.

An example of the 3 levels of communication interactions can be seen using pseudo-code in Figures 3.15 and 3.16 and from the following code fragments.

STRATEGIC LEVEL (PROLOG)

```
        .
        .
...exec_get_setpoints(X)...
...exec_submerge(X), ...
        .
...repeat, ask_depth_reached(X),...
        .
        .
```

TACTICAL LEVEL (C)

79

```c
          .
          .

int exec_get_setpoints()
{
     ++current_setpt_index;  /* Increment the Set Point Index */
     return(TRUE);
}

          .
          .

int exec_submerge()
{
     sprintf(command_sent,"%s %f %f","SUBMERGE",z_setpt[current_setpt_index],
          theta_setpt[current_setpt_index]); /* Command vehicle to Submerge to z_setpt */
     write_to_execution(command_sent);
     return(TRUE);
}

          .
          .

int ask_depth_reached()
{
     sprintf(command_sent,"%s","GET_DEPTH_INFO");
     write_to_execution(command_sent); /* Request Depth Information from Execution Level */
     read_from_execution(&command_read[0]);     /* Read Reply from Execution Level , Blocking
                                                        Socket */
     sscanf(command_read,"%F %F",&z_est,&kappa_zf);

     if( kappa_zf < kappa_zf_min[current_setpt_index] )
     {
          return(TRUE);   /* Within Minimum Error  */
     }
     else
     {
          return(FALSE);  /* Outside Minimum Error */
     }
}

          .
          .
```

EXECUTION LEVEL (C)

```c
          .
          .

{
     /* INITIALIZATION BLOCK CODE */
}

          .
          .

/* CONTROL LOOP */
while (shutdown_signal_received == FALSE)
{
     /* SOCKET READ BLOCK */
```

```c
/* Read Command (If Any) From Tactical Level */
read_status = read_from_tactical(&command_read[0]);        /* Non-Blocking Socket Read */
if(read_status > 0)
{
    sscanf(command_read,"%s",&command[0]); /* Extract the Command Only! */
    /* Switch to the Appropriate Command Parser */
    if(!strcmp(command,"SUBMERGE"))
    {
        sscanf(command_read,"%s %F %F %F %d %d",
            &command[0],&z_com,&T_z_f,&theta_com,&T_theta_f,
            &submerge_mode,&pitch_mode);

        DEPTH_AND_PITCH_CONTROL = TRUE;
    }
    else if(!strcmp(command,"GET_DEPTH_INFO"))
    {
        sprintf(command_sent,"%f %f",z_est,kappa_zf);
        write_to_sun(command_sent);
    }
}

    .

    .


/* CONTROL BLOCK */

if(DEPTH_AND_PITCH_CONTROL)
{
    /* Use Com_gen Otherwise use z_com as a step input */
    if(submerge_mode == 1)
    {
        /* Command Generator */
        com_gen_z(z0,z_f,T_z_f,&z_com,&zdot_com,&zddot_com,&z_cg_init);
    }
    if(pitch_mode == 1)
    {
        /* Command Generator */
        com_gen_theta(theta0,theta_f,T_theta_f,&theta_com,&thetadot_com,
                    &thetaddot_com,&theta_cg_init);
    }

    depth_and_pitch_control(z_com,zdot_com,zddot_com,submerge_mode,
                        theta_com,thetadot_com,thetaddot_com,pitch_mode);
}
    .

    .

    .

    .

}
```

This shows how the Strategic Level communicates with the Tactical Level which in turn sends command strings to the Execution Level for submerging control. When the Prolog predicate "exec_submerge(X)" is executed, the "C" function in the Tactical Level is

called, and writes the command SUBMERGE along with the depth and pitch angle set point for the particular phase to the Execution Level. This function then completes and having sent the command to the Execution Level, returns a value of TRUE. At this time the Execution Level extracts which command has been sent and program control is switched to the appropriate command parser block. Since SUBMERGE was sent, the command parser expects two set points, depth and pitch angle. Once this command has been received, a flag DEPTH_AND_PITCH_CONTROL is set TRUE which activates the associated function in the control block and will remain in effect until commanded otherwise.

7. Socket Communications (Tactical / Execution Level)

Careful attention must be paid to both sides of a communications socket when dealing with synchronous and asynchronous processes. Reading from a "blocking" socket causes execution to pause until data is received. In contrast to that, a "non-blocking" socket allows execution to proceed if no data is waiting to be read. For synchronous real time execution of dynamic processes attempting to make a read every time step, a "non-blocking" socket is a requirement. Since the Tactical Level sends commands and receives data asynchronously, while the Execution Level must run synchronously at 10 Hz., the UNIX side of socket A is configured to be "blocking", while the OS-9 side is "non-blocking". For instance, eight different types of socket communications are used by the Esterel language mentioned previously (Simon, et. al., 1993).

Execution of the predicate "ask_depth_reached(X)" sends a request to the Execution Level for depth information (GET_DEPTH_INFO). The command is parsed in exactly the same way as before except that the Tactical Level function waits ("blocking" socket) for the Execution Level to return the values of depth and filtered depth error. A comparison is then made between the current filtered error, κ_{zf}, and the pre-specified minimum, κ_{zf_0}. and the function returns TRUE or FALSE as appropriate.

8. Human Supervision

Human supervisory control has not been built into the control system to date. This does not mean that it is impossible to do. In fact, user inquiry for the state of the vehicle can easily be incorporated into a Tactical Level function that reads an acoustic modem and waits for messages to be received. The Strategic Level can include predicates that ask if a

user message has been received, and the Tactical Level message can be parsed into commands that could call any of the vehicle primitives directly - or specifically - request data to be changed. While the architecture supports supervisory control, that is not the main focus of this dissertation.

D.   RESULTS FROM AN EXPERIMENTAL MISSION

The mission described in this work is one of many used to verify this design of a Hybrid controller, and was performed in the NPS hover tank. During execution all pertinent data was collected, including depth, heading, thruster motor speed, etc. During phases where the sonar is active, the range and heading angle of the sonar head was recorded. A log file of mission status messages, a time history of the depth response of all three phases, and a plot of the profiling sonar image of the tank was obtained.

While the mission executes, the process running the Strategic Level displays status messages to the screen, while others are written by the Tactical Level C functions. Stored in a log file for each mission, the following was obtained with messages from the Strategic Level in upper case and in lower case for the Tactical Level.

```
?- execute_mission.

INITIALIZE MISSION!
START NETWORK!
READ MISSION FILE!
2
60.0  0.0  0.0  2.5  0.0  0.0  0.0  0.0  0.0  0.1  0.0  0.0  0.1  0.0    0  0.0  0.0
0 0    0 0  0 0  0.0  0.0  0.0  0.0  0.0  0.0
60.0  0.0  0.0  2.5  0.0  0.0  0.0  0.0  0.0  0.1  0.0  0.0  0.1  60.0  1  0.0  0.0 : mission file
0 0    0 0  0 0  0.0  0.0  0.0  0.0  0.0  0.0

Mission File opened successfully.

START PHASE 1!
INITIALIZE VEHICLE!
INITIALIZE BOARDS!
TURN ON PROP POWER!
TURN ON SONAR POWER!
UNCAGE DIRECTIONAL GYROSCOPE!
DIRECTIONAL GYROSCOPE UNCAGED.
ZEROING SENSORS.
INITIALIZE ST1000 SONAR!
INITIALIZATION DONE.
PHASE 1  COMPLETE.

START PHASE 2!
current_setpt_index = 0
```

```
            SUBMERGE!
            z_setpt = 2.5 theta_setpt = 0.0

            ROTATE!
            psi_setpt = 0.0

            START_TIMER!

            DEPTH TRANSITION.
            Depth @ transition = 2.401626
            z_dot @ switch = 0.004895
            kappa_zf @ transition = 0.093

            HEADING TRANSITION.
            Heading @ switch = -0.033422
            r @ switch = 0.003719
            kappa_psif @ transition = 0.084
            PHASE 2  COMPLETE.

            START PHASE 3!
            START SWEEP TIMER!
            SET SONAR MODE!
            START TIMER!
            SONAR SWEEP COMPLETE.
            PHASE 3 COMPLETE.

            DIS-CONNECT NETWORKS!
            MISSION COMPLETE.

    yes
    | ?-
```

The commanded depth was 2.5 feet with a filtered error threshold 0.1 feet. The set points for both pitch and heading angle were 0.0 radians, and the sonar was set to continuously sweep clockwise (Mode +1) for 60.0 seconds in phase 3. After the network connections to the various processes were established, the mission file was read by the Tactical Level. Although this was a three phase mission, only two rows of set points were required, since vehicle initialization does not require set point data.

The first column of the mission data file is the time out for a phase (seconds), the next six columns are the set points for longitudinal, lateral, depth, roll, pitch, and heading. The second set of six columns are their respective filtered error thresholds, κ_{f_o}, and the last four columns contain the duration of the sonar sweep, the sweep mode, scan direction, and sweep width. A log file is used to show the status of the various transitions and numerical values for certain variables of interest. Upon completion of phase 3, the network connections are terminated and the mission completes.

84

Figure 3.17 shows the time history of the depth and depth rate response. The lower trace shows the behavior of the filtered depth error, κ_{zf}, and the threshold for the filtered error, κ_{zf_0}. The time axis includes a short time for initialization, and in phase 2 it can be seen that κ_{zf} starts to reduce as the vehicle begins to submerge at T_1. The transition to phase 3 is triggered as κ_{zf} reaches κ_{zf_0} (T_2), when the sonar is activated and an image of the test tank walls and a cylindrical object is recorded as shown by Figure 3.18. While this phase is active, the depth controller continues to operate and reduces the error beyond the threshold of 0.1 feet to nearly zero. Once the sonar sweep time is over, the mission terminates at time T_3.

E. ARCHITECTURE EVALUATION

It is not an easy task to evaluate a given control system architecture. The theoretical design for stability and robustness leads to selection of parameters that are used in the control functions of the Execution Level. While stability and robustness of the mission control is not easily proven to the same degree as Execution Level functions, the design presented here provides the following positive features.

1) Evaluation of the controller response is provided from the mission data files, and all control parameters are in variable form which can be easily changed to tune the low level servos.

2) In this design of the mission control, every phase can be transitioned, by either normal completion, a time-out, or by abort. Therefore, no deadlocks are possible since the three termination states are all reachable.

3) When new sensors or actuators are added to the vehicle, the associated data input/output can be integrated into the control software by simply creating new functions in the Tactical and Execution Levels. Inclusion of the new sensor/actuator will not affect the functionality of the existing code, and the input/output corresponding the device is obtained by a single point modification of the data recording function.

85

4) Only the Strategic Level code requires modification if the mission is altered to eliminate, or add, a new phase. Since the phases are essentially generic in structure, and can be described by combinations of existing mission primitives, the code changes are usually minor.

5) It is a simple matter to test the performance of a single sensor or actuator set by using the corresponding mission primitive in the Strategic Level, while omitting the ones that are not of interest.

6) Conditions that define the transition signals can be easily adjusted by changing the entries in the Tactical Level mission data file. As with the control parameters, the transition thresholds are in variable form that can be tailored for particular operating conditions.

F. RELATIONS BETWEEN PROLOG AND PETRI NETS FOR STRATEGIC LEVEL IMPLEMENTATION

So far the mission controller has been represented with Prolog rules. While this is the actual language that is used to drive the Strategic Level in real time, there also exists a method to graphically model discrete mission events. These models are referred to as Petri nets (Murata, 1989), and can sometimes give a more clear and intuitive representation for a Strategic Level mission controller. It should be pointed out that using a Petri net graph is not intended to replace the Prolog code, but rather provides a different representation of the mission controller. The following sections show how Prolog code may be written given a Petri net graph, followed by a Petri net analog of the Prolog mission code outlined before. It should be noted that our use of Prolog is greatly simplified and so are the Petri net representations, being limited to single token places corresponding to the TRUE/FALSE evaluations of the predicate rules.

Three example Petri net graphs and their Prolog representations are shown in Figures 3.19-3.21. The circles denoted by P_1, P_2, ..., P_n are the states or places of the Petri net graph and the bars labeled t_1, t_2,..., t_n are the transitions, where a token enters a place whenever a transition is fired. The basic technique for writing Prolog code from an existing Petri net is manually done and not optimized, but to start at the terminal state(s) and work back towards the initial state of the graph (following the backward chaining nature of Prolog). While doing this we define "place" and "transition" predicates, where "place"

predicates evaluate TRUE if a token resides there, and a "transition" predicate is TRUE if it is enabled and the transition has fired.

Using this approach, Prolog code has been generated for three example Petri net graphs. Each Prolog example is driven by executing "mission_complete" which is the rule to be satisfied for completion with the predicate "done" repeatedly queried until TRUE. The "place" and "transition" predicates are denoted by p(n) and t(n) respectively. The "ask" predicate shown in these examples allows the user to interactively activate the firing of a transition by typing a 1 for TRUE or 0 for FALSE.

The Petri net of Example 1 shows two terminal states, P_2 and P_3, which implies that for completion, two instances of the Prolog "done" predicate must be defined, (done :- p2. or done :- p3.). Starting with the terminal states and working back, we must determine what conditions must be satisfied to reach these states. The precondition for a transition to either P_2 or P_3 requires a token in place P_1. This is assured since the rule for transition t_1 is declared to be TRUE. At this point both transitions t_2 and t_3 are "enabled" and either may fire to move the token to terminal states P_2 or P_3.

Example 2 is an OR structure with a single terminal state P_2 (done :- p2.) which may be reached one of four ways. The first three through transitions t_2, t_3, or t_4, and a fourth, directly through t_5. This requires the Prolog to have four instances of the rule P_2, which reflects the OR nature of the Petri net.

Example 3 is an AND structure with a single terminal place, P_3 (done :- p3.), where both places P_1 and P_2 must be occupied to enable transition t_3. This is described in Prolog by the AND (p1, p2) in the rule body of p3.

In the example mission, completing a phase normally, a phase time out, or having a system problem are all examples of a discrete event. In a Petri net graph, these are the transitions, the places represent the execution of mission commands such as "submerge", "rotate", etc. A Prolog "repeat" loop is also considered to be a place with it's transition predicates continually evaluated. Figure 3.22 shows the Petri net graph for the mission, which starts with a transition at t_1 (equivalent to querying the rule "execute_mission"). The places P_1, P_2 and P_3 represent the Prolog phases 1, 2 and 3 and are expanded in detail in Figures 3.23, 3.24, and 3.25 respectively. The transitions t_2, and t_5 through t_{10} denoted with a thick line are only evaluated if enabled, and fired when the associated predicate becomes TRUE, (X==1). A transition drawn with a thin line fires as soon it is enabled. In Petri net notation, we are using a "timed" graph since there are definite time delays between the enabling and subsequent firing of some - but not all - transitions. Referring to the

expanded Petri net graph of phase 2 in Figure 3.24, the transition t_{21} will fire when the rule "execute_phase(2)" is executed. Once this has occurred, the predicates represented by P_{22}, P_{23}, and P_{24} are all executed and upon completion, the transition t_{22} fires immediately. and the place P_{25} becomes active. At this point the transitions t_6, t_7, t_{23}, and t_{24} are enabled and the predicates associated with them are evaluated repetitively until one of them is TRUE, at which time the respective transition will fire. If a time out or system problem does not occur and the desired depth is reached, t_{23} will fire, and the predicate "ask_heading_reached" must also be evaluated repetitively. Since the Prolog repeat continues to evaluate transitions, t_{23} must return a token to P_{25} causing t_6, t_7, t_{23}, and t_{24} to remain enabled and evaluated. When the heading is reached, t_{24} is fired, t_{25} is enabled, and fires immediately completing phase 2 normally. A similar structure of phase completion and error recovery is used in phase 3 as well, which can serve as a template for most any mission phase.

If a time out or system problem occurs in either phase, it is clearly shown that one of the transitions t_6 through t_9 will fire and the "exec_surface" predicate (place P_5) will be executed. In this state the predicate "ask_surf_reached" is continually queried until the surface is reached at which time the mission is aborted (P_6). If all goes well and the objectives of phases 2 and 3 are met, the place P_4 (mission_complete) is reached and the mission terminates normally.

G.   CONCLUSIONS

The conclusion of the work in this chapter has indicated that complex behavior can be readily coordinated through Strategic Level rules, that are easily modified. These act as state transitioning mechanisms and the communication through Tactical Level software to the Execution Level controllers is a simple but convenient way of commanding stable responses of the vehicle. The design of well behaved control laws and functions at the Execution Level is essential as a primary part of the design and is affected through careful attention to the digital control loop design. Reactivity, failure recovery, and even human interfacing within the controller can take place at any level.

Although the example mission here was simplified so that the details of the code and results could be more clearly presented, other more complex missions have been performed successfully.

H. CHAPTER III FIGURES
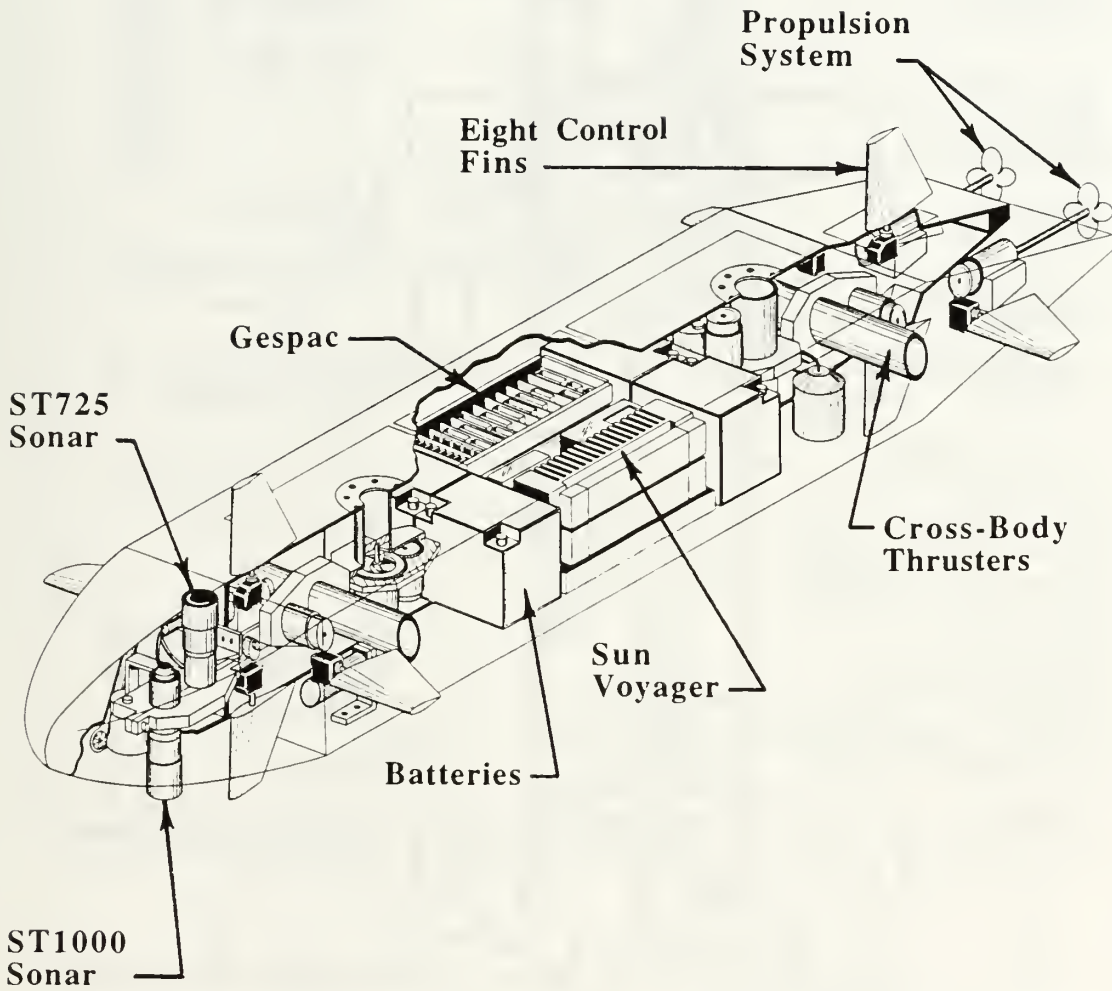
Figure 3.1 Perspective View of the NPS Phoenix.

91

ST725 SONAR

DEPTH CELL
TRANSDUCER

RADIO
ETHERNET
BOARD

BOW LEAK
DETECTOR

BOW LATERAL
THRUSTER

VERTICAL
GYRO

BOW VERTICAL
THRUSTER

COMPUTER POWER
SUPPLY (2)

MOTOR SERVO
CONTROLLER (6)

SUN VOYAGER
COMPUTER

STERN VERTICAL
THRUSTER

FREE GYRO
POWER SUPPLY

STERN LATERAL
THRUSTER

PSA 900
UNIT

CONTROL FINS (8)

ST1000 SONAR

PSA 900 SONAR
TRANSDUCER

TURBO PROBE

FIN SERVO (8)

3 AXIS RATE GYRO

12 VOLT BATTERY (2)
FOR COMPUTER

GESPAC CARD CAGE

DIVE TRACKER

12 VOLT BATTERY (2)
FOR GYROS/MOTORS

FREE GYRO

GPS UNIT

REAR LEAK DETECTOR

REAR SCREW MOTOR (2)

REAR SCREW (2)

Drawn By D. Marco '96

Figure 3.2 Major Internal Components of the NPS Phoenix.

92

RADIO ETHERNET
ANTENNA (FIXED
TO HULL OR FLOATING)

GPS ANTENNA

ST725 SONAR

ST1000 SONAR

TURBO PROBE

FIN

DIVE TRACKER
TRANSDUCER

POWER PLUG

DRAIN
PLUG

SIDE VIEW

THIN WIRE / RADIO
ETHERNET PORT

ACCESS HATCH

DIFFERENTIAL
GPS ANTENNA

THRUSTER

REAR SCREWS
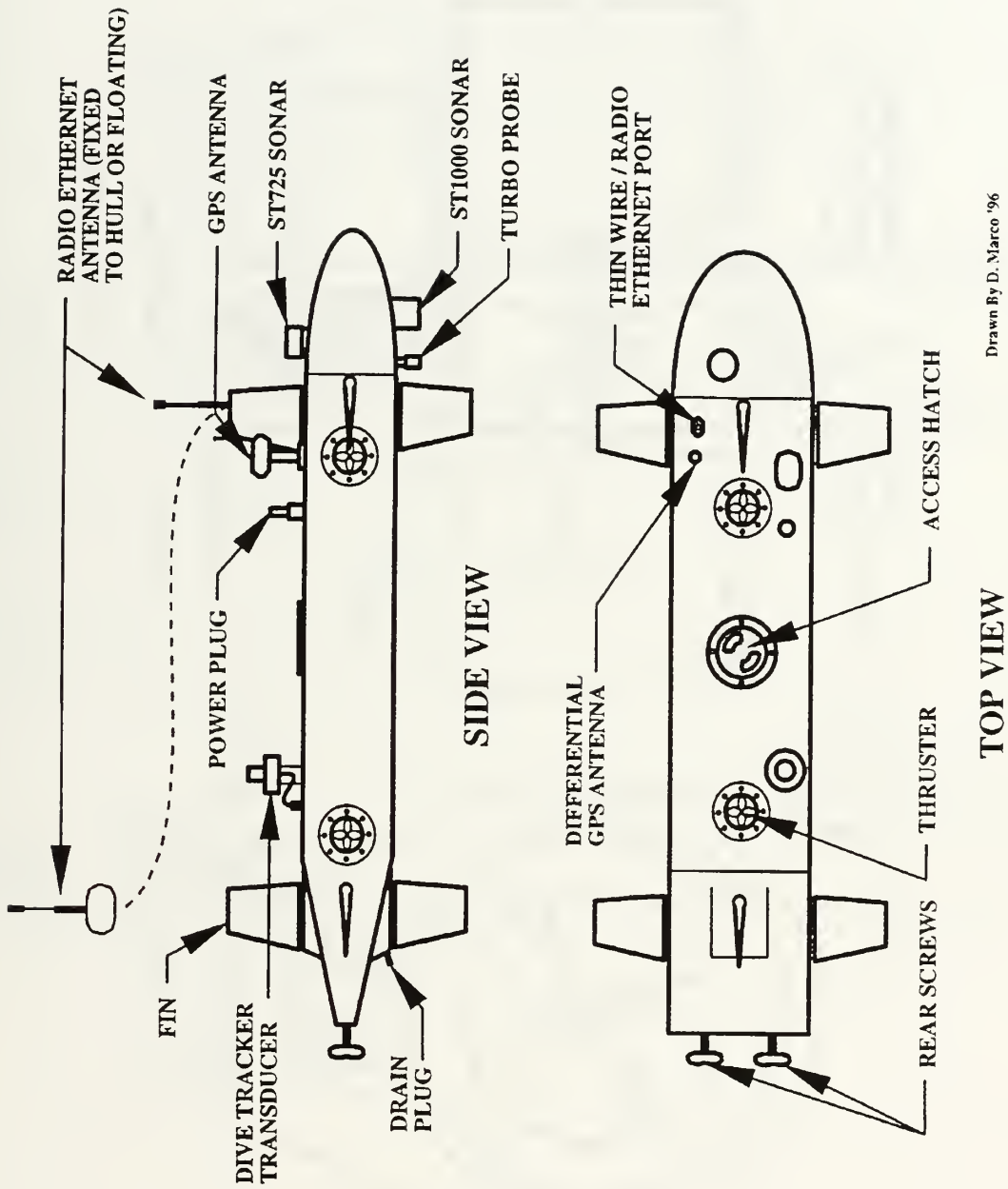
TOP VIEW

Drawn By D. Marco '96

Figure 3.3 External Components of the NPS Phoenix.

93
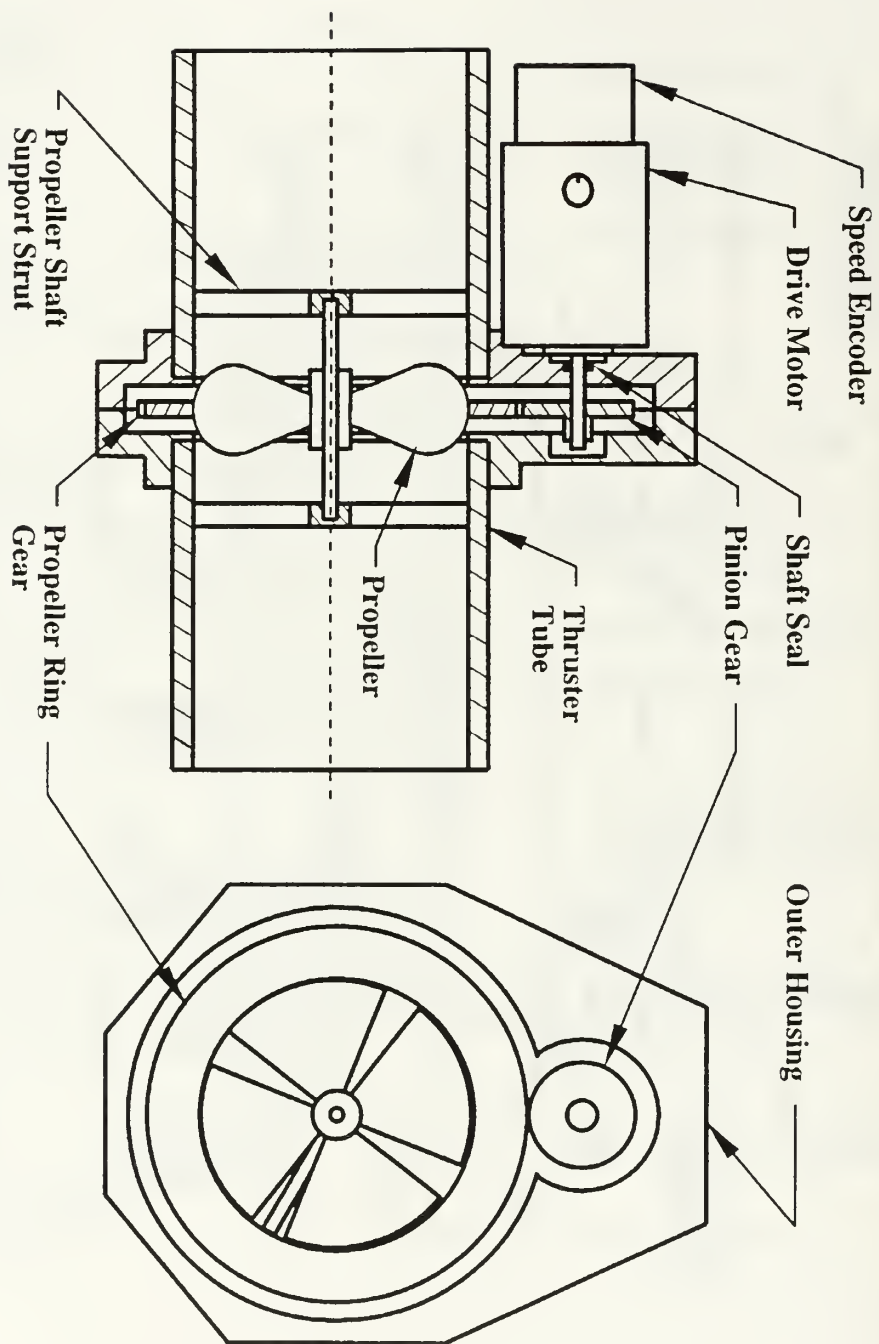
Figure 3.4 Section and Front View of the Cross-Body Thruster.

Labels on the figure:
- Propeller Shaft Support Strut
- Propeller Ring Gear
- Propeller
- Thruster Tube
- Pinion Gear
- Shaft Seal
- Drive Motor
- Speed Encoder
- Outer Housing
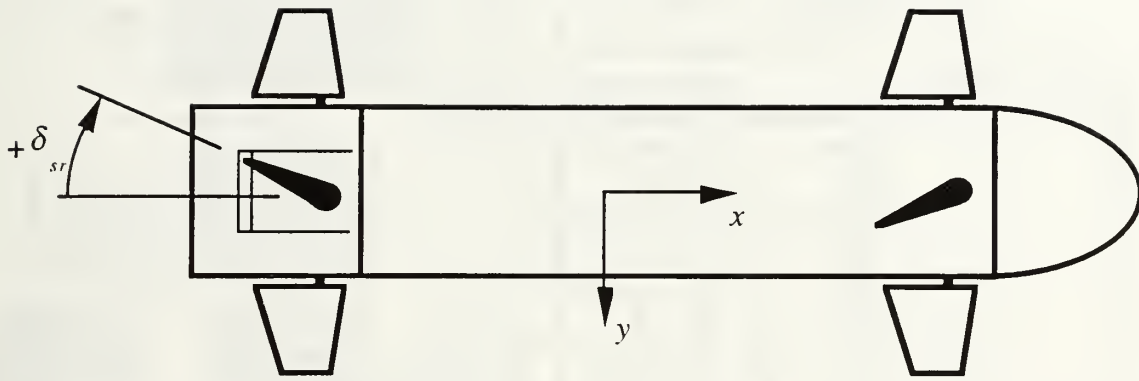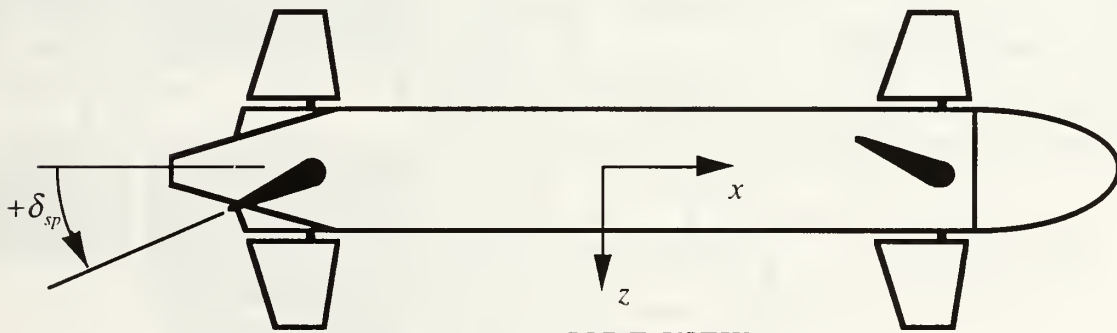
Figure 3.5 Rotation Convention for the Control Surfaces.
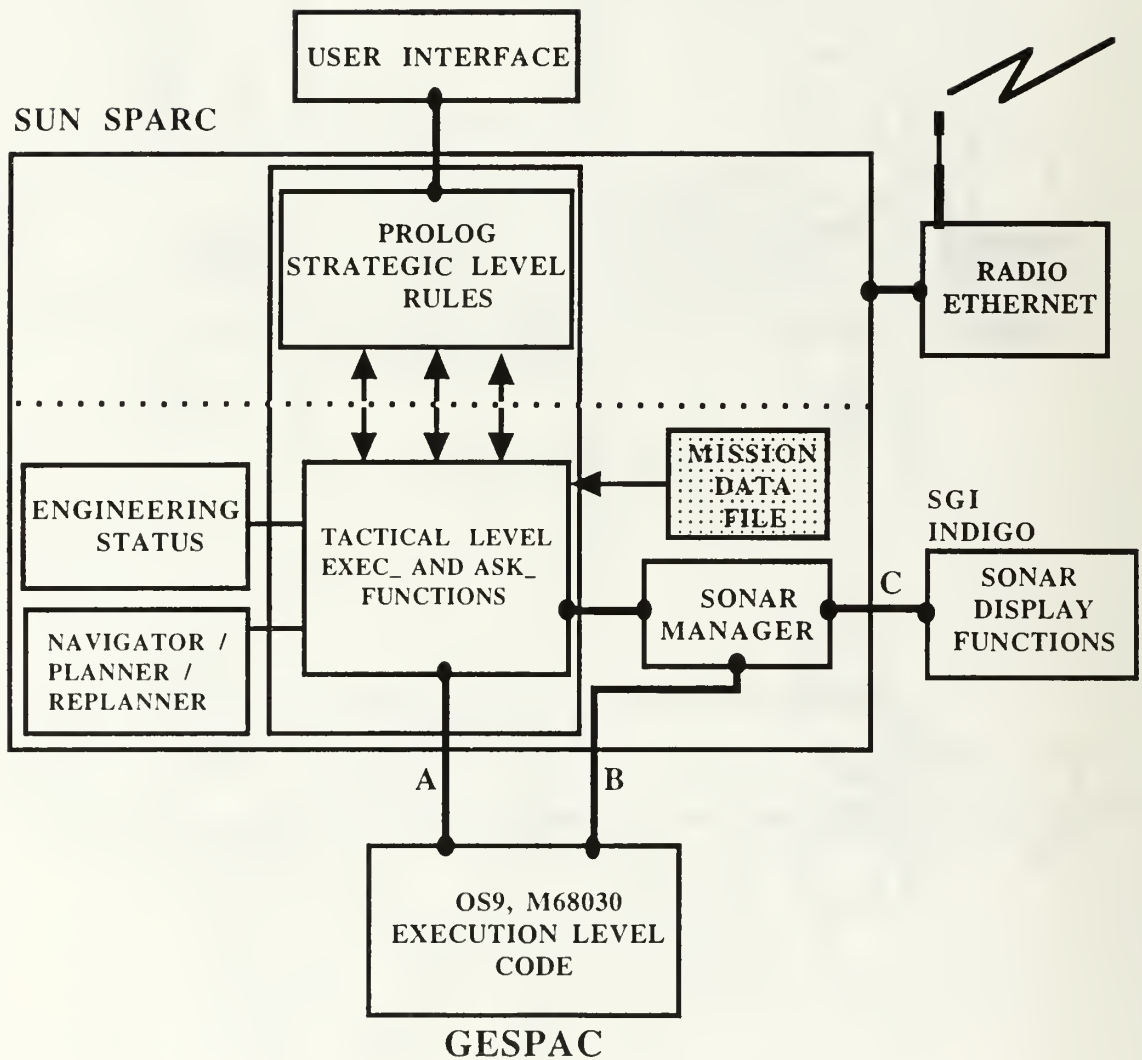
Figure 3.6 Diagram of the Phoenix Networked Controller.
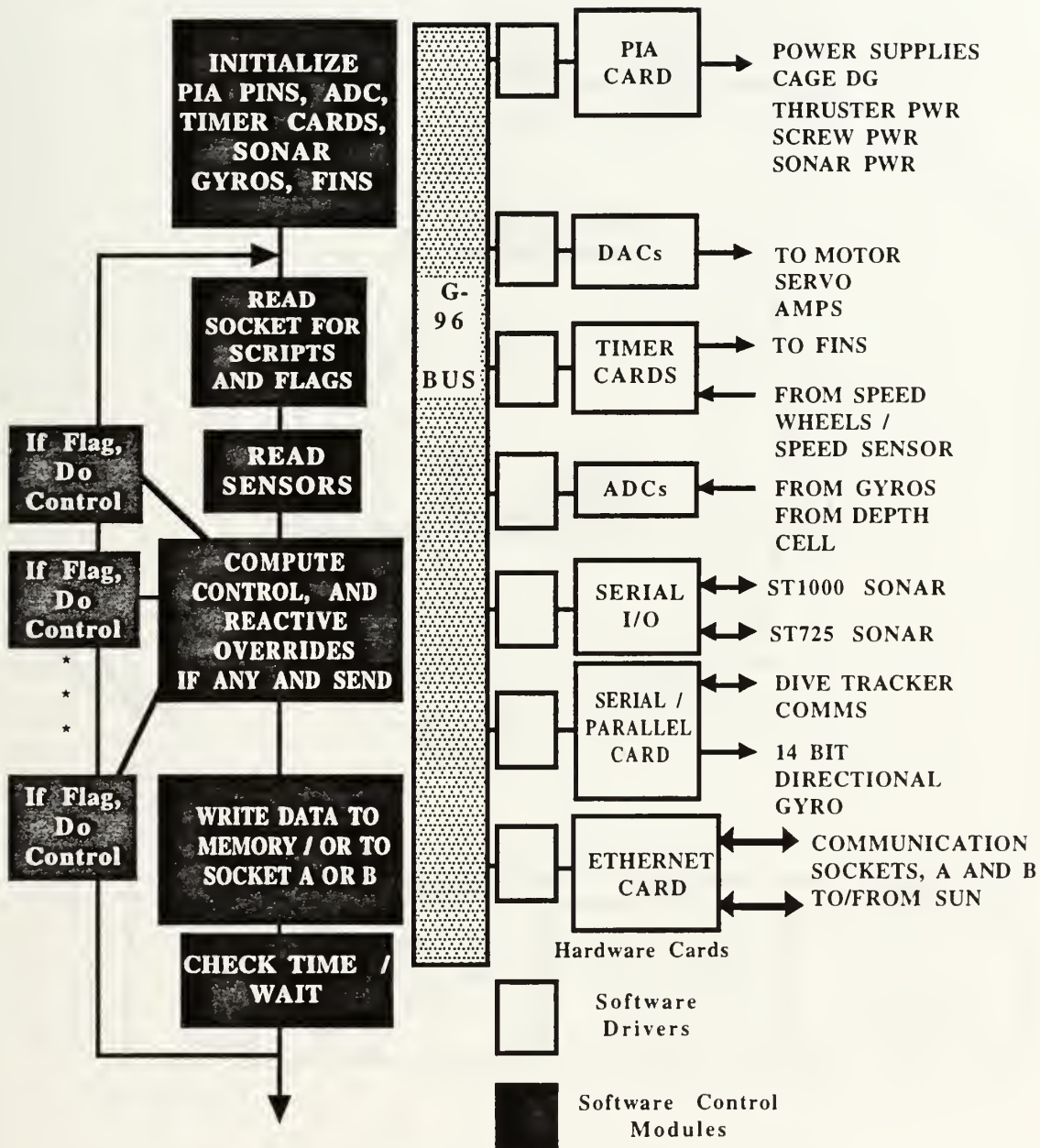
Figure 3.7 Structure of the Execution Level Software/Hardware.
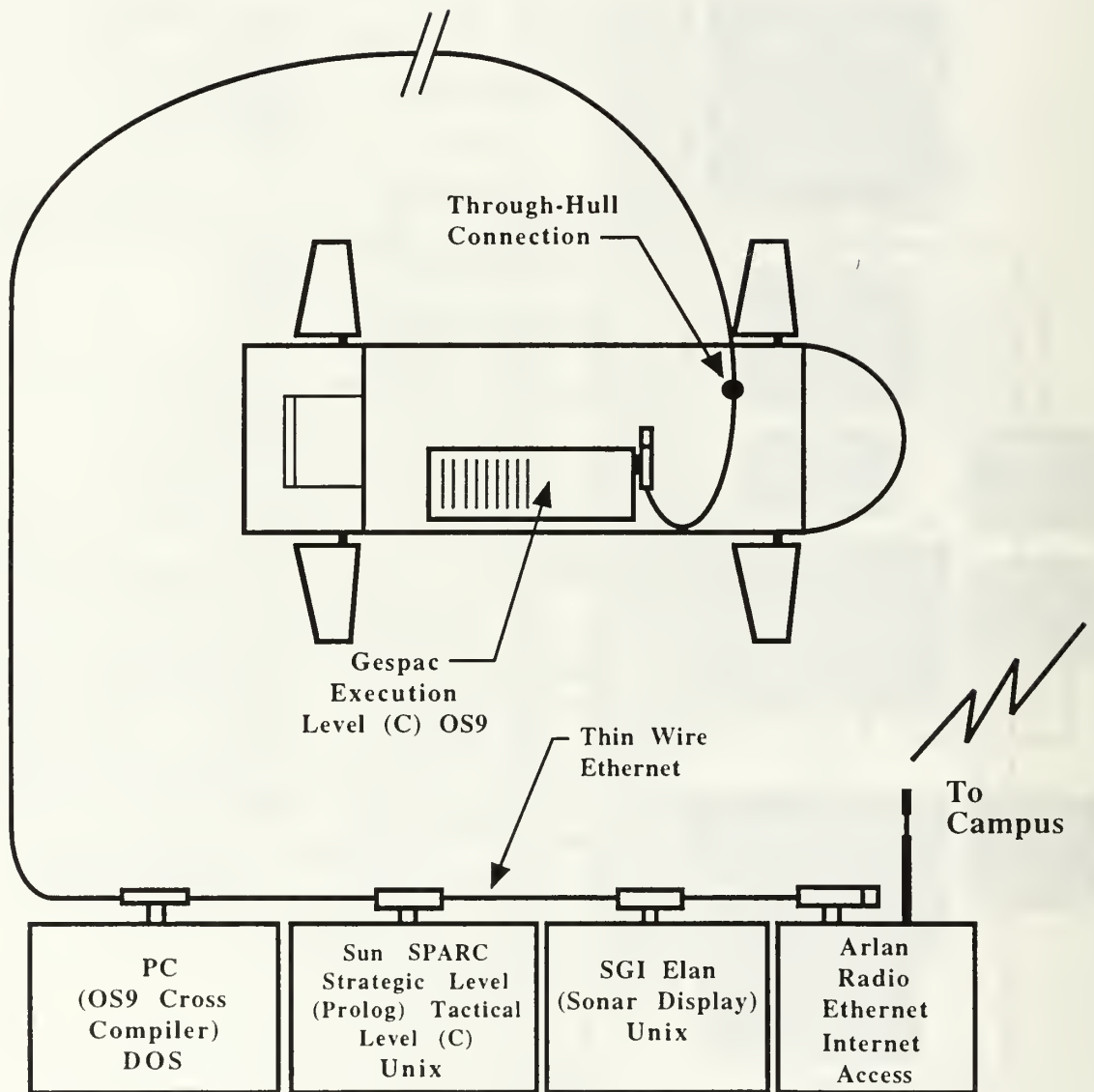
Figure 3.8 Network Configuration for Experiments in the NPS Test Tank.
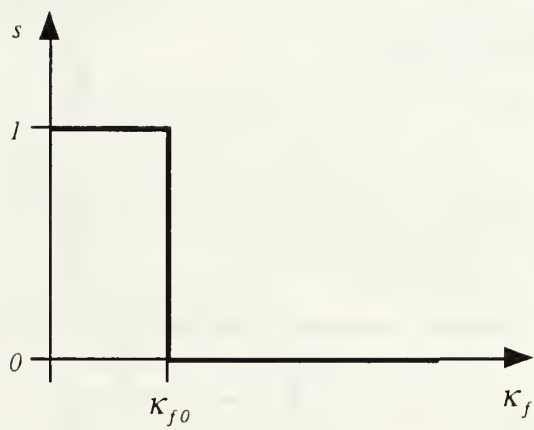
Figure 3.13 The Transition Signal Condition.

**INITIALIZATION
BLOCK**

**SOCKET READ
BLOCK**

**READ SENSORS
BLOCK**

**CONTROL
BLOCK**

**DATA RECORDING
BLOCK**

TIMER WAIT

CONTROL LOOP
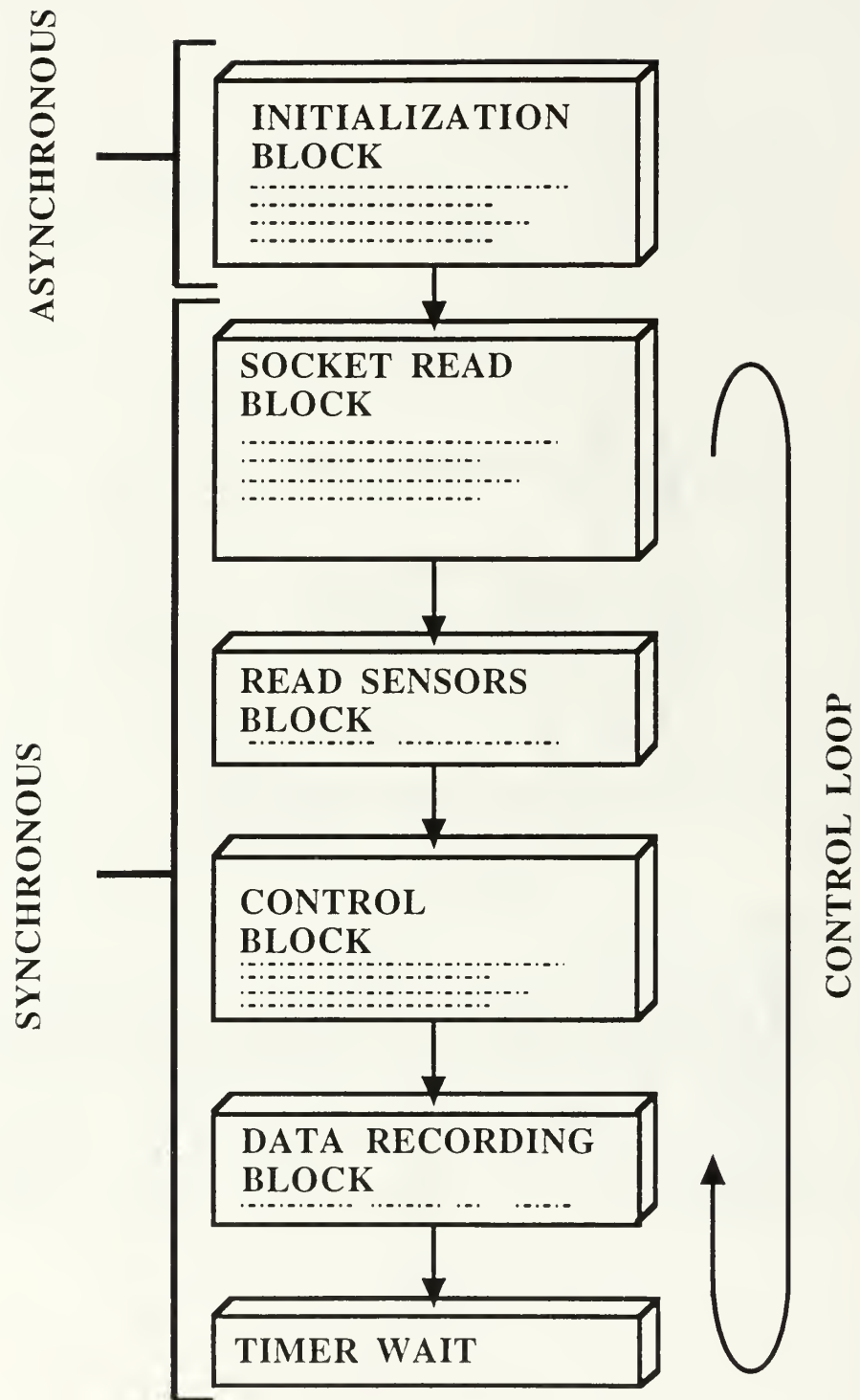
Figure 3.14 Structure of the Execution Level Software.

EXECUTE FUNCTION PROCESS

STRATEGIC LEVEL

MISSION DATA FILE

..., | exec_submerge(), | ...
.
.

. . . .
. . . .
. . . .

LINKED THROUGH
FUNCTION CALL

LINKED THROUGH
FILE READ

TACTICAL LEVEL

| exec_submerge() |

Command = SUBMERGE + | setpoint + mode |

| write_to_execution(Command); |
.
.

LINKED THROUGH
ETHERNET SOCKET

EXECUTION LEVEL

CONTROL LOOP

| read_from_tactical(Command) | -> NON-BLOCKING SOCKET

if(SUBMERGE)

ACTIVATE SUBMERGE BEHAVIOR USING setpoint + mode

Figure 3.15 Communication Interactions Between the Three Levels using
"exec_submerge".

ASK FUNCTION PROCESS

STRATEGIC LEVEL

..., | ask_depth_reached() | ...

LINKED THROUGH
FUNCTION CALL

LINKED THROUGH
FUNCTION CALL

TACTICAL LEVEL

| ask_depth_reached() |

Command = GET_DEPTH_INFO

| write_to_execution(Command); |

| read_from_execution(Command_Read); |

Analyze Data Returned

is $K_f \leq K_{f0}$?

| RETURN TRUE OR FALSE |

BLOCK UNTIL DATA

ARRIVES

LINKED THROUGH
ETHERNET SOCKET

EXECUTION LEVEL

CONTROL LOOP

| read_from_tactical(Command) | -> NON-BLOCKING SOCKET

if(Command == GET_DEPTH_INFO)

| write_to_tactical(z, K_f); |

LINKED THROUGH
ETHERNET SOCKET

Figure 3.16 Communication Interactions Between the Three Levels using
"ask_depth_reached".

102
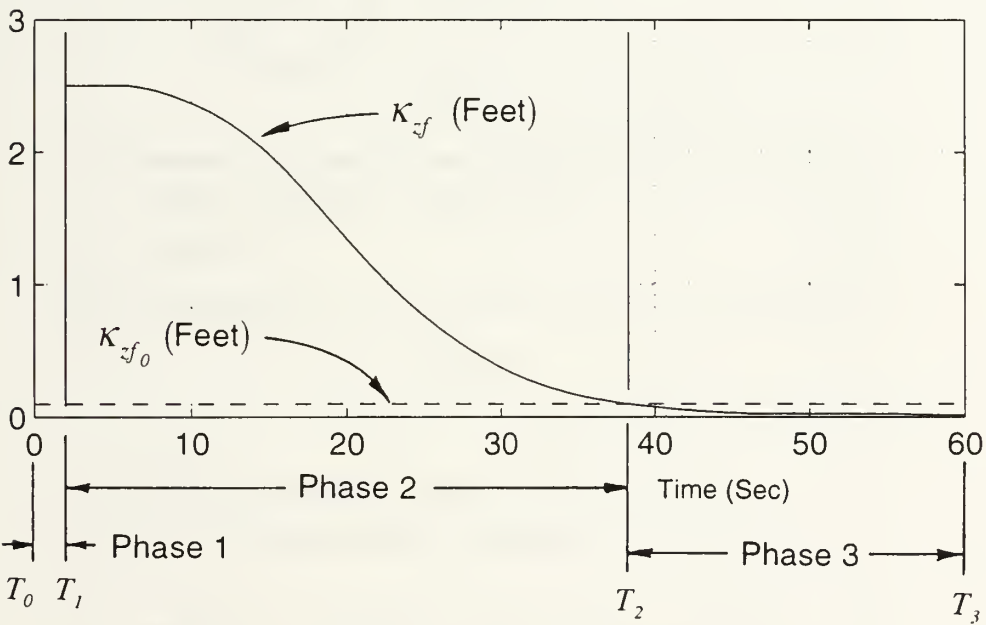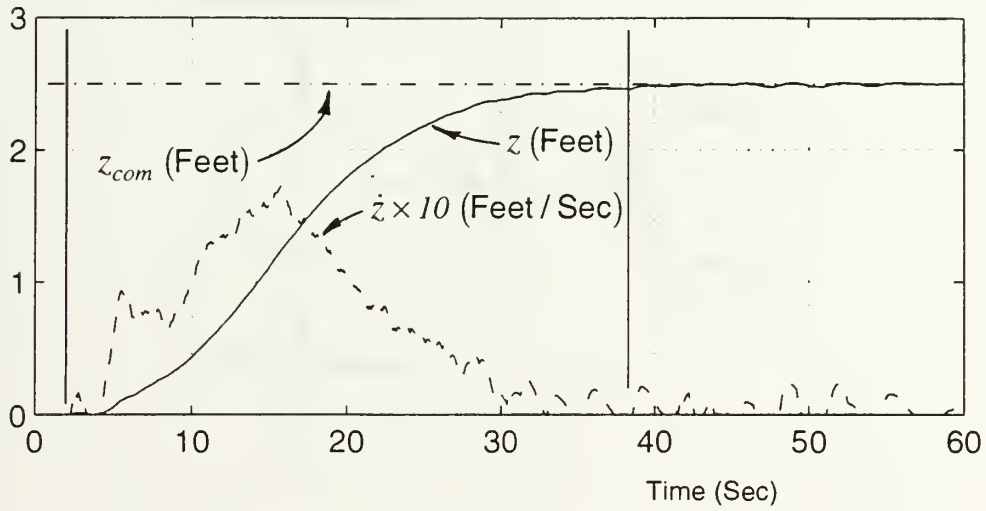
Figure 3.17 Vehicle Depth, Depth Rate, and Filtered Error vs. Time Response for All Three Phases.

103
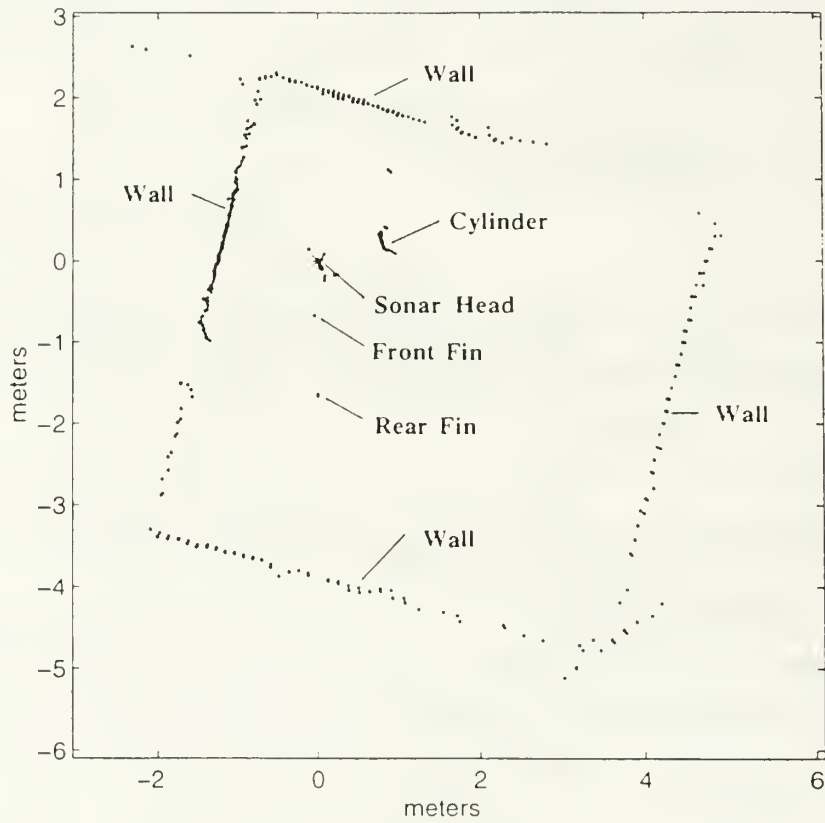
Figure 3.18 ST1000 Sonar Image of the NPS Test Tank showing a Cylinder, the Tank Walls, and the Vehicle Fins.
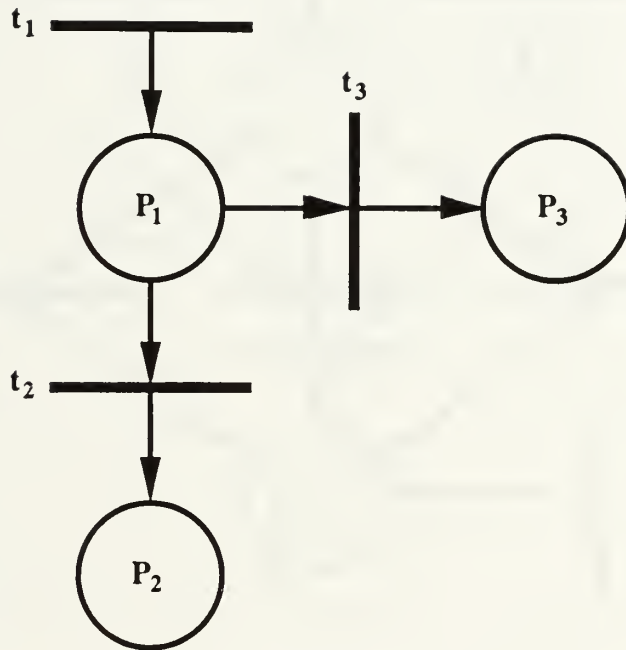
```
mission_complete :- repeat, done.
done :- p2.
done :- p3.


t1.
p1 :- t1.
p2 :- p1, ask(' t2(X)', X), X == 1.
p3 :- p1, ask(' t3(X)', X), X == 1.


ask(Q,A) :- write(Q), write(' ?'), nl, read(A), nl.
```

Figure 3.19 Petri Net Graph Representation and Prolog Code for Example 1.

105
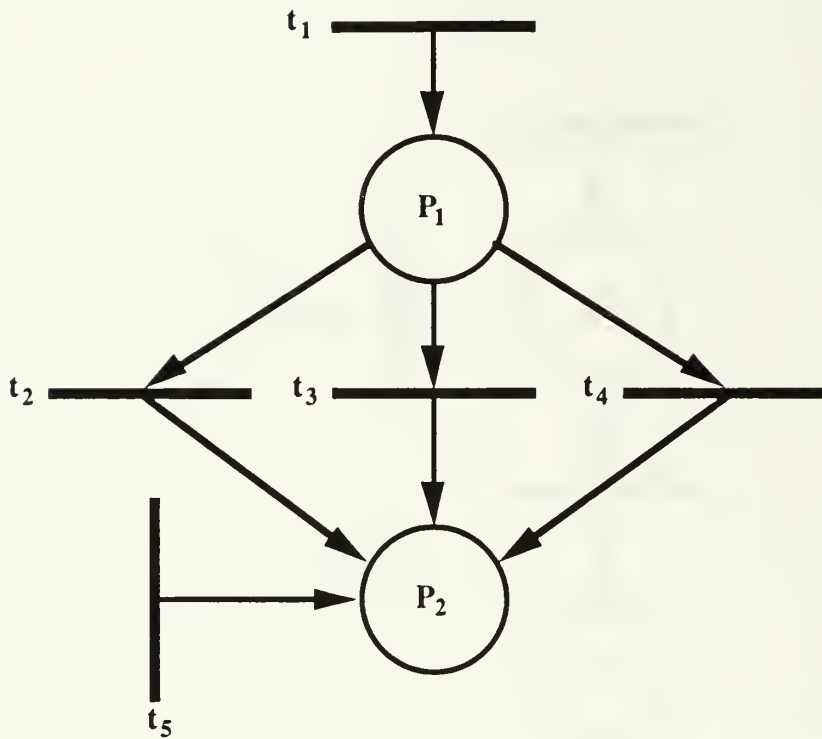
```
mission_complete :- repeat, done.
done :- p2.


tl.
pl :- tl.
p2 :- pl, ask(' t2(X)', X), X == 1.
p2 :- pl, ask(' t3(X)', X), X == 1.
p2 :- pl, ask(' t4(X)', X), X == 1.
p2 :- ask(' t5(X)', X), X == 1.


ask(Q,A) :- write(Q), write(' ? ), nl, read(A), nl.
```

Figure 3.20 Petri Net Graph Representation and Prolog Code for Example 2.
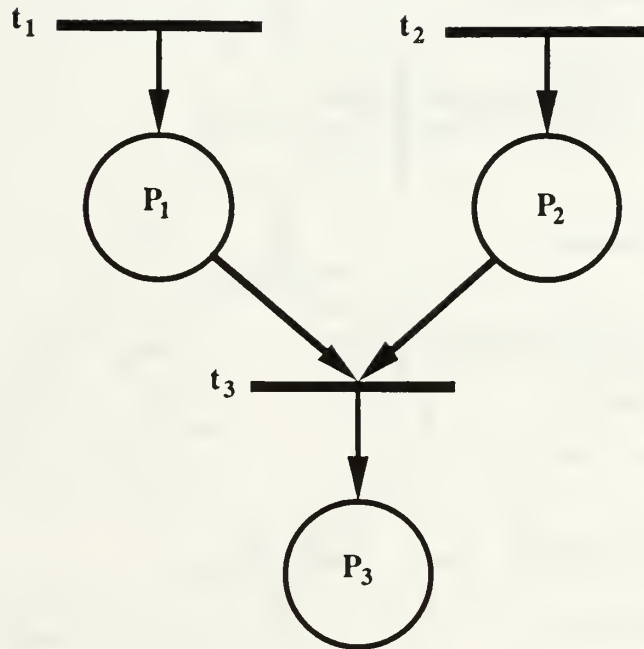
```
mission_complete :- repeat, done.
done :- p3.

p1 :- ask('t1(X)', X), X == 1.
p2 :- ask('t2(X)', X), X == 1.
p3 :- p1, p2, ask('t3(X)', X), X == 1.

ask(Q, A) :- write(Q), write('?'), nl, read(A), nl.
```

Figure 3.21 Petri Net Graph Representation and Prolog Code for Example 3.

107
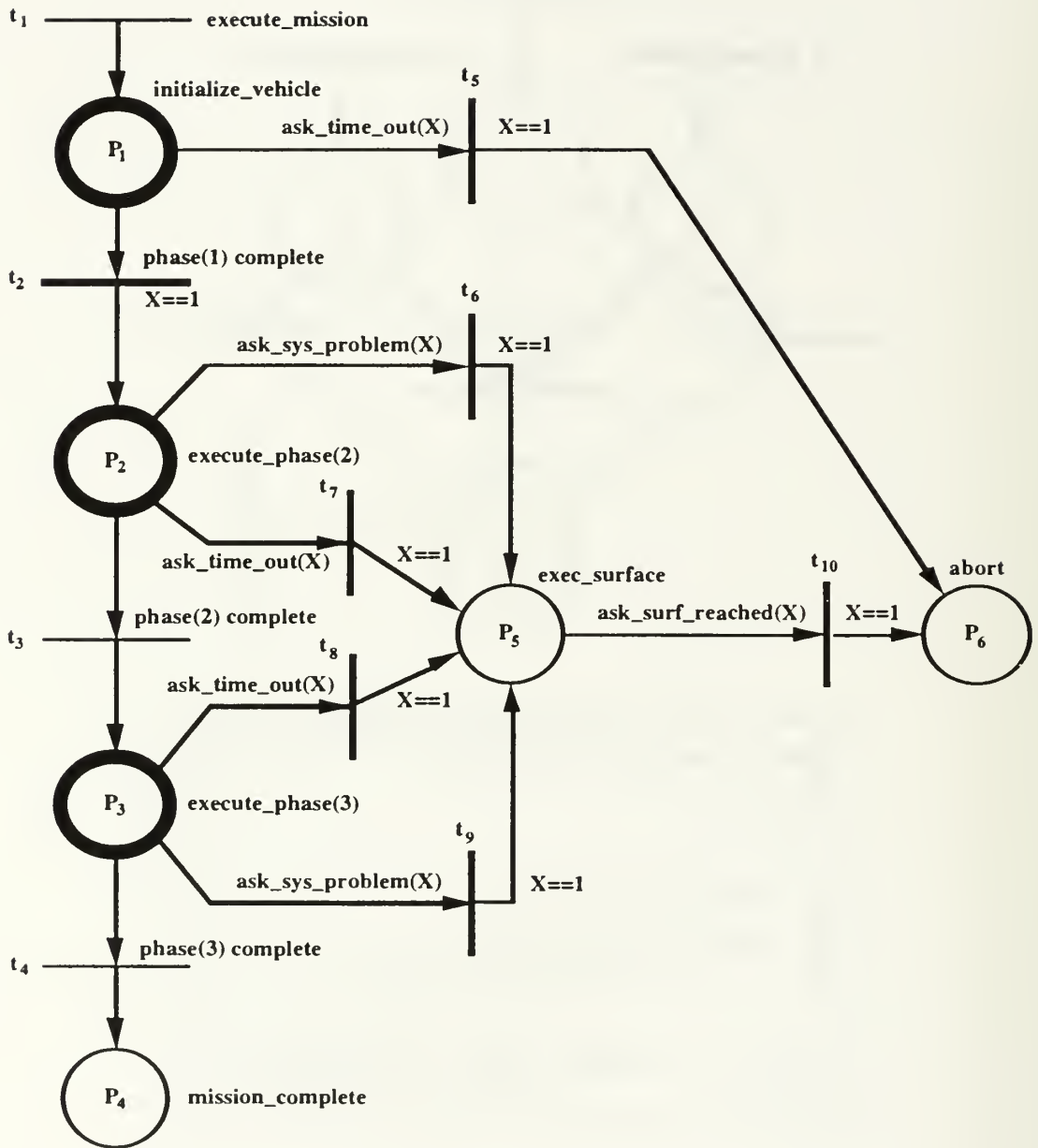
Figure 3.22 Petri Net Graph Representation of the Prolog Code for the Generic Mission.
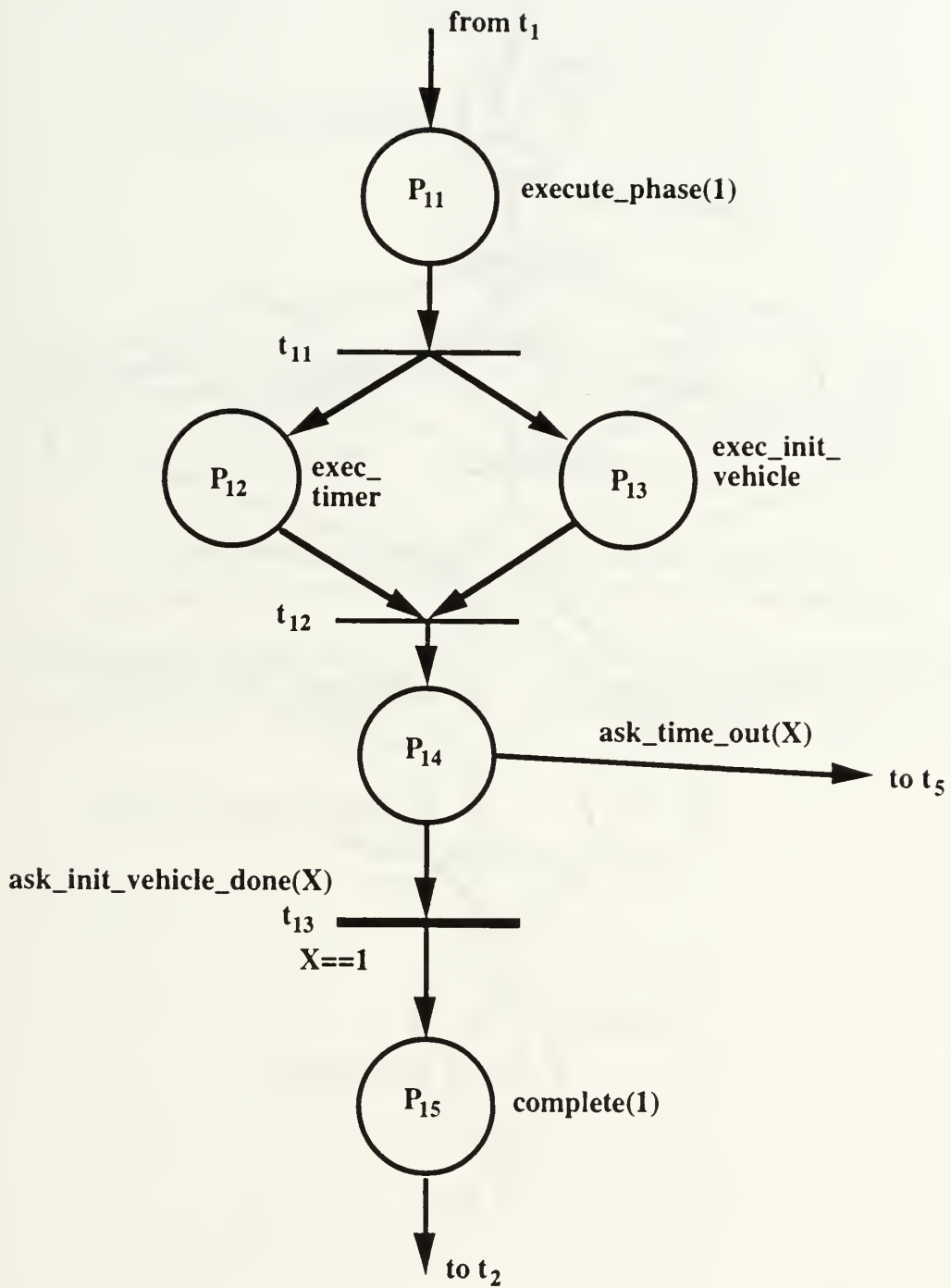
Figure 3.23 Petri Net Graph Representation of the Prolog Code for Phase 1.
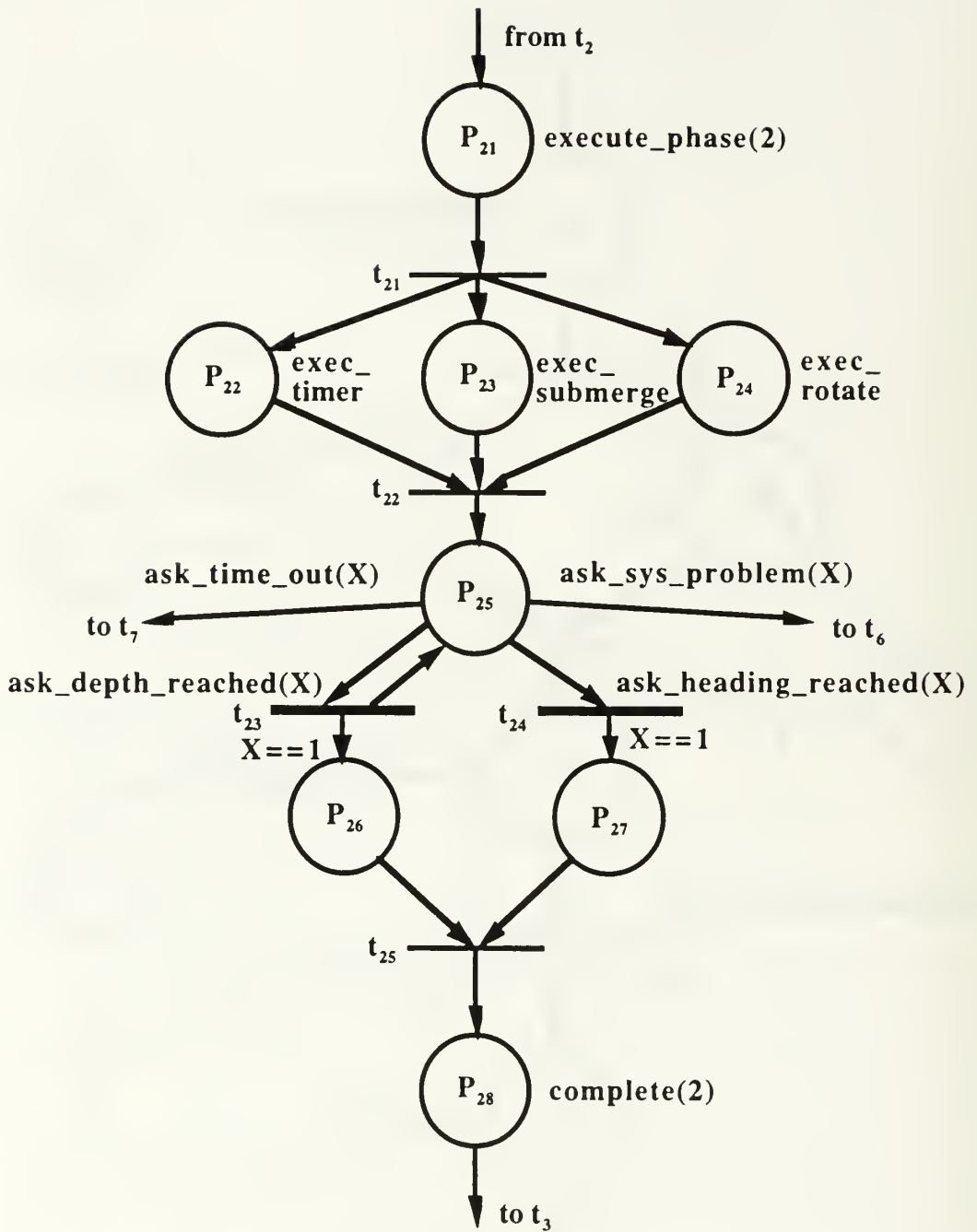
Figure 3.24 Petri Net Graph Representation of the Prolog Code for Phase 2.
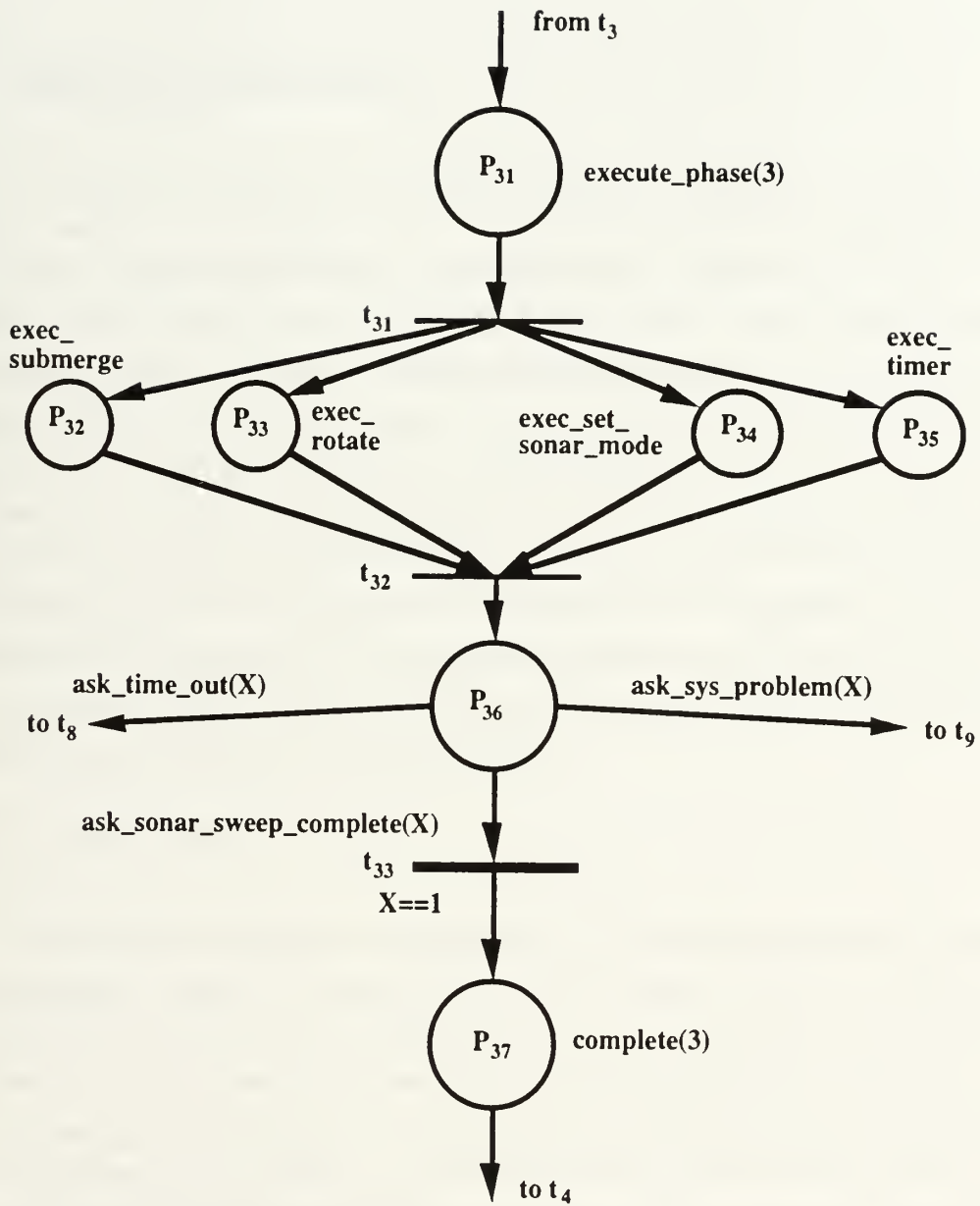
Figure 3.25 Petri Net Graph Representation of the Prolog Code for Phase 3.

IV. TRITECH MECHANICALLY SCANNABLE SONARS

A. GENERAL DESCRIPTION

In order to investigate the potential for using commercially available, high frequency sonar for controlling position of an AUV, the vehicle was equipped with two mechanically scannable high frequency sonar heads built by Tritech Corp. One is an ST725 scanning sonar and the other an ST1000 profiler sonar. These sonars have been successfully tested in the NPS test tank, the NPS swimming pool, and in the harbors at Monterey and Moss Landing, CA. Figure 4.1 shows where the sonar units are located on the vehicle with the heads located inside a rubber boot at the upper end of the housings. Each head can be scanned continuously through 360 degrees of rotation or swept through any defined angular sector, around the central axis of the unit. During normal operation the head will ping, wait for return echoes to process, and then rotate a specified angular width and repeat. The ST1000 and ST725 sonars measure approximately 9 inches long by 2.75 inches in diameter. Both heads are powered by 24 Vdc and communicate with the host computer using an RS-232 serial link at 9600 Baud.

1. ST725 Sonar

The ST725 sonar operates at a frequency of 725 kilohertz and emits an acoustic beam 2.5 degrees wide in the horizontal plane by 24.0 degrees wide in the vertical plane, 12.0 degrees above and below the horizontal plane as shown in Figure 4.2. This device is described as a scanning sonar due to the nature of the range information returned for each ping. A scanning sonar operates by placing the intensities of the echoes from each ping into discrete segments of range called range bins. For this sonar, the number of range bins is nominally 128 but for operating ranges of 10 meters or less, the number of range bins is reduced to 64. The maximum operating range of the ST725 is 100 meters with a minimum operating range of 6 meters, and provides a resolution of (Maximum Range)/128 or (Maximum Range)/64 depending on the range setting used. At the minimum range setting used for the majority of the experiments conducted in the NPS hover tank, the smallest range resolution was approximately 9 cm.

The range bins and associated intensities define a vector of intensities called a scanline for a given bearing angle of the head, and is sent through the serial port to the controlling computer as a 32 or 64 byte data stream, 64 or 128 bins as appropriate. Each byte (8 bit binary) contains the intensity information for two bins, where the first byte, (byte0), holds the intensities for bin0 and bin1, the second byte holds intensities for bin2 and bin3, etc., as shown in Figure 4.3. Since the intensity for each bin is 4 bits wide, the intensity value can range only from 0 - 15 where 15 denotes the strongest return and zero, no return.

In order to more clearly analyze the returns, the data can be thresholded to display only returns above a certain intensity value so that significant objects/structures can be shown, while other less significant entities (e.g. suspended particles in the water column, weak multi-path echoes), are excluded from the display. Combining thresholding with an appropriate power setting for the transducer, can usually provide very clear displays. Figures 4.4, 4.5, and 4.6 show an ST725 sonar display of the NPS AUV test tank. The head was suspended from the vertical catwalk and an aluminum cylinder was placed in the tank. Clearly visible are the sides of the tank, with the echoes from the cylinder shown near the upper left hand corner of the tank display. A power gain of 13 (0 < gain < 100), and an angular step size of 0.9 degrees was used for all three displays, and thresholds 1, 10, and 15 were used for Figures 4.4, 4.5, and 4.6 respectively. Adjustment of the threshold shows the filtering ability of this technique. In Figure 4.4, it is clear that the tank walls are highly reflective since there is an apparent continuation of each wall beyond the tank boundaries. Also, the effect of the finite resolution is indicated by the short segments at constant radius emanating from the sonar head location.

2. ST1000 Sonar

The ST1000 sonar operates at a frequency of 1024 kilohertz and emits a 1.5 degree conical acoustic beam as shown in Figure 4.2, and is described as a profiler sonar since only the range to the nearest return echo is recorded. With the use of a high frequency ping, imaging of objects to a high degree of detail is possible. Operating ranges for the ST1000 sonar are from about 1 meter to a maximum of 50 meters, and the unit is pressure rated to full ocean depth.

For each ping, the sonar head returns to the controlling computer a 2 byte data stream, which contains the distance to the first return in units of millimeters. Figure 4.7

shows an image of the tank sides, along with two objects clearly visible inside the tank. The semi-circular shape in the upper part of the display is an aluminum cylinder standing vertically in the water column. In the lower part of the display is a line, which is another aluminum cylinder lying in the horizontal plane showing the high degree of detail which can be obtained from this sonar. Results from work done by (Ingold, 1992), has shown that objects can be imaged to a resolution of less than 2 cm.

B. COMPUTER CONTROL OF THE SONAR HEAD

Both sonar heads are rotated with a stepper motor. With a maximum of 400 steps/rev, the highest angular resolution is 0.9 degrees, although double and triple stepping can be enabled resulting in 1.8 or 3.6 degree increments for faster rotation rate. Around the stepper motor shaft is a position encoder which returns an ASCII "F" or "T" to the serial port each time the motor is stepped. Figure 4.8 shows the locations about the encoder where the values of "F" or "T" are returned. Output from the encoder allows the sonar head to be centered at the same location upon initialization, and the count of steps either clockwise (CW) or counter-clockwise (CCW) is maintained by the controlling software. The exact angular position of the head can then be determined at any subsequent time.

A procedure for centering the sonar head to it's "ahead position" (i.e. the middle of the region of T's) is outlined in Figure 4.9. The procedure shown assumes high resolution 0.9 degree steps but the control software has been generalized to also accommodate 1.8 and 3.6 degree step size settings and is accomplished with a control function called "center_sonar". Once the head is centered, the angular position of the sonar head ψ_s, is defined to be 0.0, ($-180.0 < \psi_s < 179.1$ degrees), and the angular position count ψ_{sc}, is set to 0, ($-200 < \psi_{sc} < 199$). Each time the head is stepped, ψ_{sc} is updated by either +ssiz or -ssiz, depending on the direction of rotation. Three step sizes can be programmed, ssiz = 1, 2, or 4 corresponds to 0.9, 1.8, or 3.6 degrees respectively, which leads to a simple relation for calculating the head position in degrees

$$\psi_s = 0.9 \psi_{sc} \tag{4.1}$$

There is no external feedback of step count so this method of realizing the sonar heading assumes that the motor does in fact step on open loop command. The control software developed in this dissertation allows three positioning modes of operation for the sonar head, which are:

Mode 0: Ping only mode   -   Ping sonar at present heading only (do not step).

Mode 1: Continuous sweep mode   -   Ping and step continually sweeping in a constant clockwise direction.

Mode -1: Continuous sweep mode   -   Ping and step continually sweeping in a constant counter-clockwise direction.

Mode 2: Sector sweep mode   -   Ping each step within a defined angular sector.

Parameter definitions for Mode 2 are shown in Figure 4.10, where ψ_{sd} is defined as the scan direction, and is the angle that the head position will oscillate with an angular amplitude of ψ_{sw}, the sweep width. Mode 1 can be achieved, as a subset of Mode 2 by simply specifying ψ_{sd} with ψ_{sw} set to 0.0. It thus follows that the values of ψ_{sd}, ψ_{sw} and the positioning mode can be dynamically changed at any time as needed according to the context of the mission control.

1. Sonar Control Functions in Detail

Both the ST725 and ST1000 sonar heads can be controlled using a small set of ASCII character commands sent over the serial line. The sonar(s) do not have automatic modes to continuously ping and rotate given a single command. Therefore, the head must be commanded through the serial port to ping and/or rotate for each time. Below is a list of the commands available for position and pinging control for both heads.

ST1000 Commands:

    "+"   Rotate 1 Step Clockwise without Pinging. Returns 1 Byte Encoder Character, "T" or "F".

    "-"   Rotate 1 Step Counter-Clockwise without Pinging. Returns 1 Byte Encoder Character, "T" or "F".

    "Z"   Ping Once Without Stepping. Returns 2 Byte Range.

    ")"   Ping Once Then Rotate 1 Step Clockwise. Returns 2 Byte Range followed by a 1 Byte Encoder Character, "T" or "F".

"("   Ping Once Then Rotate 1 Step Counter-Clockwise. Returns 2 Byte Range followed by a 1 Byte Encoder Character, "T" or "F".

ST725 Commands:

"+"   Rotate 1 Step Clockwise without Pinging. Returns 1 Byte Encoder Character, "T" or "F".

"-"   Rotate 1 Step Counter-Clockwise without Pinging. Returns 1 Byte Encoder Character, "T" or "F".

"S"   Ping Once Without Stepping. Returns 32 or 64 Byte Range Bins.

"R"   Ping Once Then Rotate 1 Step Clockwise. Returns 32 or 64 Byte Range Bins followed by a 1 Byte Encoder Character, "T" or "F".

"L"   Ping Once Then Rotate 1 Step Counter-Clockwise. Returns 32 or 64 Byte Range Bins followed by a 1 Byte Encoder Character, "T" or "F".

In order to ease operation, several "C" language functions have been written to coordinate control of the heads. The following is a list of the functions and parameter lists for use in the vehicle Gespac computer. Using the function

$$center\_sonar(path);$$

commands the sonar head rotate to the ahead position, where **path** = Serial port number, and can be used to center either the ST725 or ST1000. The function

$$st1000\_sonar\_command = set\_st1000\_mode($$
$$st1000\_sweep\_mode,$$
$$st1000\_scan\_direction,$$
$$st1000\_scan\_width,$$
$$\&st1000\_psi\_sonar\_min,$$
$$\&st1000\_psi\_sonar\_max);$$

is used to determine which command to send to the ST1000 head based on the parameters given by:

st1000_sweep_mode	=	Sweep Mode 0,1,-1,2
st1000_scan_direction	=	Scan Direction, ψ_{sd}, 0.0...360.0 Degrees
st1000_scan_width	=	Scan Width, ψ_{sw}, 0.0...360.0 Degrees

If the sweep mode is 2, the values of *st1000_psi_sonar_min* and *st1000_psi_sonar_max* are calculated and returned based on *scan_direction* and *scan_width* . If mode 0 is used, *st1000_sonar_command* is set to "Z", if mode 1 or -1, the commands ")" or "(" are returned. The character "Z" is returned for mode 2 and the following function must be called each time step to update the command to the head.

$$st1000\_sonar\_command = \text{st1000\_sweep\_control(}$$
$$st1000\_psi\_sonar,$$
$$st1000\_psi\_sonar\_min,$$
$$st1000\_psi\_sonar\_max);$$

where *st1000_psi_sonar* = Current Angular Position of the head.

update_st1000_position(*st1000_sonar_command*,
&*st1000_psi_sonar_count*,
st1000_ssiz);

updates the head position *st1000_psi_sonar_count* based on *st1000_sonar_command* and *st1000_ssiz*, where

st1000_psi_sonar_count = Integer count of head position 0...400, incremented by +*st1000_ssiz* if *st1000_sonar_command* is ")", -*st1000_ssiz* if "(", and 0 if the command is "Z". *st1000_ssiz* is 1, 2, or 4 for step sizes 0.9, 1.8, and 3.6 degrees respectively.

ping_st1000_sonar(*st1000_sonar_command*);

sends command *st1000_sonar_command* to the ST1000 head.

st1000_range = **read_st1000_sonar();**

reads the range from ST1000 and returns it in *st1000_range*. The pinging and reading functions have been separated since other calculations and sensor reading may be performed during the waiting time for the sonar return.

The ST725 uses basically the same functions outlined above but since some of the ASCII commands used for this head differ from the ST1000, different function names must be used. A call to the function

$$st725\_sonar\_command \quad = set\_st725\_mode($$
$$st725\_sweep\_mode,$$
$$st725\_scan\_direction,$$
$$st725\_scan\_width,$$
$$\&st725\_psi\_sonar\_min,$$
$$\&st725\_psi\_sonar\_max \quad);$$

determines which command to send to the ST725 head based on the parameters given by:

$st725\_sweep\_mode$ = Sweep Mode 0,1,-1,2
$st725\_scan\_direction$ = Scan Direction, ψ_{sd}, 0.0...360.0 Degrees
$st725\_scan\_width$ = Scan Width, ψ_{sw}, 0.0...360.0 Degrees

(NOTE: Same as with the ST1000)

If the sweep mode is 2, the values of $st725\_psi\_sonar\_min$ and $st725\_psi\_sonar\_max$ are calculated and returned based on *scan_direction* and *scan_width* . If mode 0 is used, $st725\_sonar\_command$ is set to "S", if mode 1 or -1, the commands "R" or "L" are returned. The character "S" is returned for mode 2 and as with the ST1000, the following function must be called each time step to update the command to the head.

$$st725\_sonar\_command = st725\_sweep\_control($$
$$st725\_psi\_sonar,$$
$$st725\_psi\_sonar\_min,$$
$$st725\_psi\_sonar\_max);$$

where

$st725\_psi\_sonar$ = Current Angular Position of the Head.

The function

$$\textbf{update\_st725\_position}(st725\_sonar\_command,$$
$$\&st725\_psi\_sonar\_count,$$
$$st725\_ssiz \quad);$$

updates the head position *st725_psi_sonar_count* based on *st725_sonar_command* and *st725_ssiz*, where *st725_psi_sonar_count* = Integer count of the head position 0...400, incremented by +*st725_ssiz* if *st725_sonar_command* is "R", -*st725_ssiz*

if "L", and 0 if the command is "S". *st725_ssiz* is 1,2, or 4 for step sizes 0.9, 1.8, and 3.6 degrees respectively. The function call

<div align="center">

ping_st725_sonar(*st725_sonar_command*);

</div>

sends command *st725_sonar_command* to the ST725 head. Calling

<div align="center">

read_st725_sonar(&*st725_range [0]*);

</div>

reads the range bins from the ST725 and returns them in array *st725_range* .

2 . Sonar Initialization

Before any sonar data may be collected, the head must be initialized with the appropriate parameters for a given operating range. Range settings for the scanning sonar (ST725) are from 6 to 100 meters, while for the profiler sonar (ST1000), the settings range from 1 to 50 meters. Tables 4.1, 4.2, and 4.3 list the initialization parameters for these ranges, and the sequence of initialization is outlined below using a function, **send_st725()** and **send_st1000()**, which is used to send characters or numbers to the head. Comments and the data type of values to send are to the right. Three different data types are used, Char (1 byte ASCII), Byte (1 Byte Binary), and Word (2 Byte Binary). The reader can refer to Appendix G for a detailed listing of the standard sonar commands.

ST725 Head Initialization Sequence:

1	**send_st725('P');**	Send Sonar Parameters (Char)
2	**send_st725(TxPulse);**	Send TxPulse Length (Word)
3	**send_st725(NSAMPL);**	Send NSAMPL (Byte)
4	**send_st725(NBINS);**	Send NBINS (Byte)
5	**send_st725(Range_Code);**	Send Range_Code (Byte)
6	**send_st725(checksum);**	Send checksum (Byte)
7	**send_st725('X');**	Enable Time Varying Gain Mode (Char)
8	**send_st725('K');**	Set mode return Range bin Peak (Char)

9	**send_st725('E');**	Set Final Gain for TVG (Char)
10	**send_st725(212);**	Gain Based on 0..255 (Byte)
11	**send_st725('C');**	Set Initial Gain for TVG (Char)
12	**send_st725(gain);**	Gain Based on 0..255 (Byte)

ST1000 Head Initialization Sequence:

1	**send_st1000('P');**	Send Sonar Parameters (Char)
2	**send_st1000(TxPulse);**	Send TxPulse Length (Word)
3	**send_st1000(NSAMPL);**	Send NSAMPL (Byte)
4	**send_st1000(NBINS);**	Send NBINS (Byte) (If in Scanning Mode)
5	**send_st1000(Range_Code);**	Send Range_Code (Byte)
6	**send_st1000(checksum);**	Send checksum (Byte)
7	**send_st1000('X');**	Enable Time Varying Gain Mode (TVG) (Char)
8	**send_st1000('K');**	Set mode return Range bin Peak (Char)
9	**send_st1000('E');**	Set Final Gain for TVG (Char)
10	**send_st1000(212);**	Gain Based on 0..255 (Byte)
11	**send_st1000('J');**	Send Profiler Sonar Parameters (Char)
12	**send_st1000(ECPULS);**	Send ECPULS (Word)
13	**send_st1000(TIMOUT);**	Send TIMOUT (Word)
14	**send_st1000(LOKOUT);**	Send LOKOUT (Word)
15	**send_st1000(ESWAIT);**	Send ESWAIT (Word)
16	**send_st1000(GECMIN);**	Send GECMIN (Byte)
17	**send_st1000(GAINDT);**	Send GAINDT (Word)
18	**send_st1000(ECSCLX);**	Send ECSCLX (Word)
19	**send_st1000(ECSCLY);**	Send ECSCLY (Word)
20	**send_st1000(Maxdst);**	Send Maxdst (Word)
21	**send_st1000(DACSCX);**	Send DACSCX (Word)
22	**send_st1000(DACSCY);**	Send DACSCY (Word)
23	**send_st1000(Rng_unit);**	Send Rng_unit (Byte)
24	**send_st1000(checksum);**	Send checksum (Byte)

The duration of the acoustic transmission pulse is defined by TxPulse length (μ sec) as shown in Figure 4.11. TIMOUT (μ sec) is the maximum time to wait for returns after the pulse has been sent, and is approximately equal to 2*Max Range/c. LOKOUT (μ sec) is the amount of time for the pulse to travel before meaningful results are obtained, which is typically set to 400 μ sec. Using this setting, returns from less than about 0.6 meters from the head will be ignored. The value LOKOUT = (minimum distance)/c. ESWAIT (μ sec) is the wait time for the head to index in autoechosounder mode (Not Applicable). GECMIN is the output power gain equal to 2.55*gain, where 0 < gain < 100. Checksum is the lower 8 bits of the sum of the initialization values previously sent. For the ST725 and the ST1000, the values sent in steps 2 through 5 are used, and for the ST1000, an additional checksum is calculated based on the values sent in steps 12 through 23.

Table 4.1 Initialization Parameters for Scanning Mode (ST725 and ST1000 Heads)

Operating Range (m)	TxPulse (1.96 μ sec)	NSAMPL	NBINS	Range Code
6	30	1	64	0
10	30	3	64	1
20	100	3	128	2
25	125	4	128	3
30	150	6	128	4
50	250	12	128	5
100	475	26	128	7

Table 4.2 Initialization Parameters for Profiling Mode (ST1000 Head)

Operating Range (m)	TxPulse 1.96 μ sec units	NSAMPL	NBINS	Range Code	TIMOUT (μ sec)	Maxdst (mm)
1	30	1	64	00	1500	1500
2	30	1	64	01	3000	3000
4	30	1	128	02	6000	6000
6	30	1	128	03	9000	9000
10	40	1	128	04	15000	15000
20	50	3	128	05	30000	30000
30	75	6	128	06	45000	45000
50	100	12	128	07	65535	65535

Table 4.3 Values Common to all Range Settings for ST1000 Sonar

ECPULS	1.96 μ sec units	30
LOKOUT	(μ sec)	200
ESWAIT	1.96 μ sec units	25600
GECMIN		2.55*gain
GAINDT		64
ECSCLX		16383
ECSCLY		11374
DACSCX		256
DACSCY		3125
Rng_unt		0

The initialization sequence previously outlined is performed for both heads using the following function:

initialize_sonar(*port,mode,max_range,gain,&Nbins*)

Input Values:

    port   -   RS232 communications port number (int)

    mode   -   Scanning "S" or Profiling "P" (char)

    max_range -   Maximum range setting (int)

    gain   -   Sonar power gain 0..100 (int)

Output Values:

    Nbins   -   Number of scan bins to collect 64 or 128
                           for the particular initialization (int)

3. Implementation of Sonar Functions

The functions previously described are designed to operate in a control loop where the sonar control functions are called once per time step. Experience has shown that the fastest the sonar can be operated within 6 meters is 0.1 seconds, which allows enough time for the head to ping, receive and process a return, then step. Therefore, if both sonars are to be used and the time step of the control loop is only 0.1 seconds, only one sonar may be operated per time step. A limitation that can be easily remedied by operating each head once

every other time step or simply increasing the time step value. The latter solution may not be appropriate if the sonar is used for position feedback in a vehicle control system due to stability requirements based on update rates. It has then been suggested that one sonar be used during one phase of a mission and then switching to another for a different phase. An example of this would be to use the ST725 scanning sonar for large area search and once a target of interest has been found, the ST1000 would take over for more detailed imaging/servoing once in close proximity of the target. Another solution is ping the sonar(s) at the beginning of the control loop, perform other calculations and then read the returns at the end. It is this method that is used with the NPS Phoenix, and the structure is outlined below with the following C code fragments:

```
st725_port = open("/t2",S_IWRITE+S_IREAD);
st1000_port = open("/t3",S_IWRITE+S_IREAD);


center_sonar(st725_port);
center_sonar(st1000_port);


initialize_sonar(st725_port,'S',st725_max_range,st725_gain,
                 &st725_Nbins);
initialize_sonar(st1000_port,'P',st1000_max_range,
                 st1000_gain,&st1000_Nbins);


START OF MISSION LOOP:


st1000_sonar_command = set_st1000_mode(
                        st1000_sweep_mode,
                        st1000_scan_direction,
                        st1000_scan_width,
                        &st1000_psi_sonar_min,
                        &st1000_psi_sonar_max);

st725_sonar_command = st725_sweep_control(
                        st725_psi_sonar,
                        st725_psi_sonar_min,
```

124

```
                                     st725_psi_sonar_max);

        update_st725_position(st725_sonar_command,
                              &st725_psi_sonar_count,
                              st725_ssiz );

        update_st1000_position(st1000_sonar_command,
                               &st1000_psi_sonar_count,
                               st1000_ssiz);

        ping_st725_sonar(st725_sonar_command);
        ping_st1000_sonar(st1000_sonar_command);
                              .

                              .

        (Perform other calculations while waiting for sonar return)

                              .
                              .
        read_st725_sonar(&st725_range [0]);
        read_st1000_sonar(&st1000_range [0]);

        st725_psi_sonar  = 0.9*st725_psi_sonar_count;
        st1000_psi_sonar = 0.9*st1000_psi_sonar_count;


END MISSION LOOP:

        close(st725_port);
        close(st1000_port);
```

Experience with both the sonars on the Gespac computer has shown that the heads do not always return the correct number of bytes when requested. Sometimes no, less than, or more bytes are returned for a given ping command, and can lead to "locking up" of the controlling process if the serial read function attempts to read more bytes than are present in the buffer. For example, the program can not simply assume that 3 bytes of data will always arrive when the ST1000 head is issued the command "L", which should return 2 bytes of range data and 1 encoder byte. Occasionally only 2 bytes will arrive for the

command, and if the read function attempts to read 3 bytes, the process will hang indefinitely since the third, expected byte never arrives. The following code fragment of the function **read_profile_sonar**() outlines the steps taken to ensure robust operation of the ST1000 sonar head.

```
read_profile_sonar()
{
    n_bytes = _gs_rdy(st1000_path);    /* Check How Many Bytes are on The Port */
    timeout = 0;       /* Initialize the Timeout Counter */
    RESET_PORT   =   FALSE;      /* Assume the Port Does Not Need to be Reset */

    while(n_bytes < profile_sonar_bytes_expected)   /* Loop Until Data is There */
    {
        tsleep(2);      /* Wait for 0.02 seconds */
        n_bytes = _gs_rdy(st1000_path);    /* Check How Many Bytes are on The Port */
        /* Reset the Port if an Insufficient Number of Bytes Have Arrived Within the Allotted
        Time */
        if((timeout == 1) && (n_bytes != profile_sonar_bytes_expected))
        {
            RESET_PORT   =   TRUE;
            break;
        }
        timeout = timeout + 1;    /* Increment the Timeout Counter */
    }

    if(RESET_PORT)
    {
        st1000_range = 0.0;   /* Set The Range To Zero By Default Since No Range
                                Available */
        close(st1000_path);   /* Close The Serial Port Path */
        st1000_path = open("/t3",S_IWRITE+S_IREAD);    /* Reopen The Path */
        n_bytes = _gs_rdy(st1000_path);    /* Check The Newly Opened Port And Check For
                                Any Stray Bytes */
        if(n_bytes != -1 ) n=read(st1000_path,x,n_bytes); /* Read Them If There To Ensure a
```

```
    n_bytes = -1;      /* Set n_bytes = -1 To Fail If Statements To Follow */

}


if(n_bytes == profile_sonar_bytes_expected)

{

    /* Read The Range From The Sonar */

}

if(n_bytes > profile_sonar_bytes_expected)

{

    /* An Incorrect Number Of Bytes Has Arrived In The Buffer */

    /* Clear Them Out By Reading Them */

    st1000_range = 0.0;   /* Set The Range To Zero By Default */

}

profile_sonar_pinged = FALSE;      /* Signify that the Head has Been Read */

return(st1000_range); /* Return the Range */

}
```

Extensive tested of this code has been done, and operates extremely well. No port lockups have occurred since it's implementation.

C.   CONCLUSIONS

This chapter has presented a detailed overview and description of the Tritech ST725 and ST1000 sonars. Example sonar images have been given along with the data formats for both units, along with useful computer functions for controlling and initializing each sonar head.
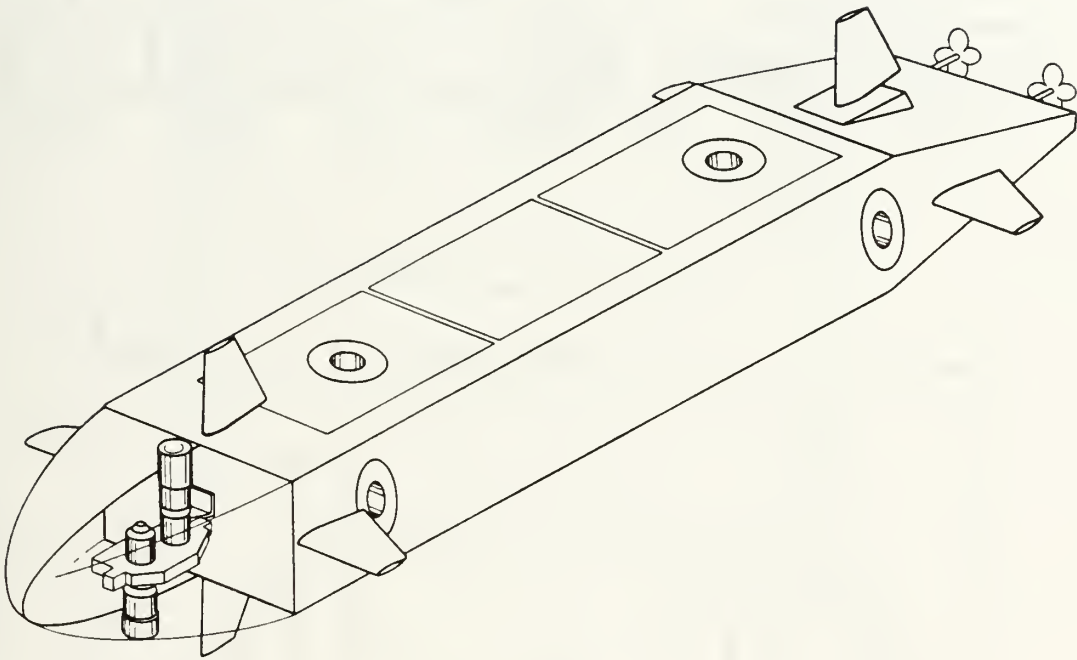
D. **CHAPTER IV FIGURES**

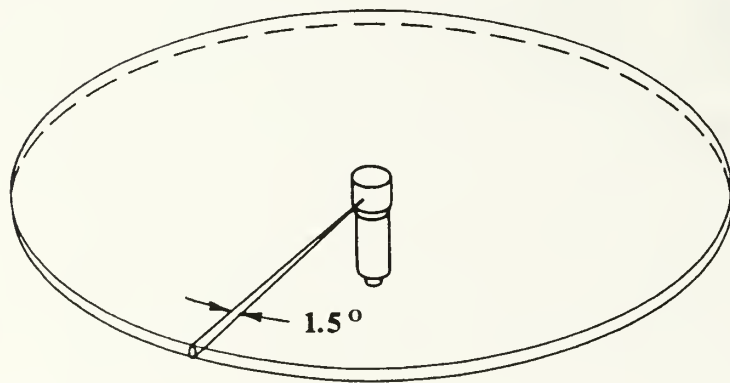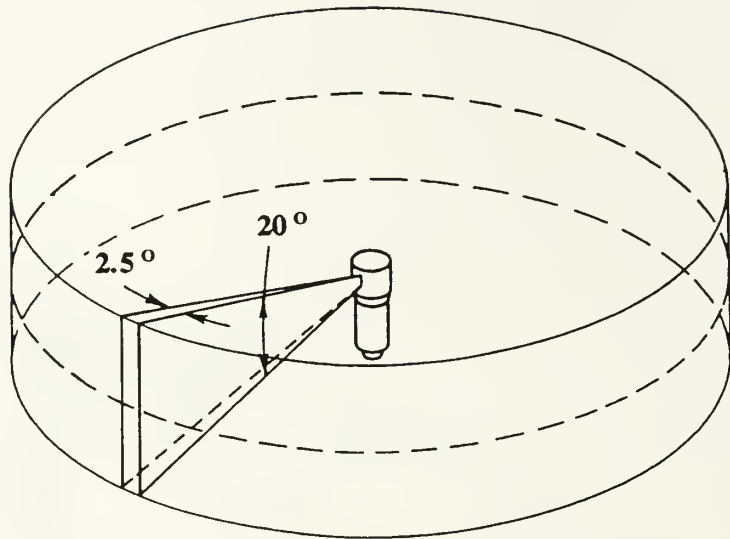Figure 4.1 Location of the Tritech Sonar Heads on the NPS Phoenix.

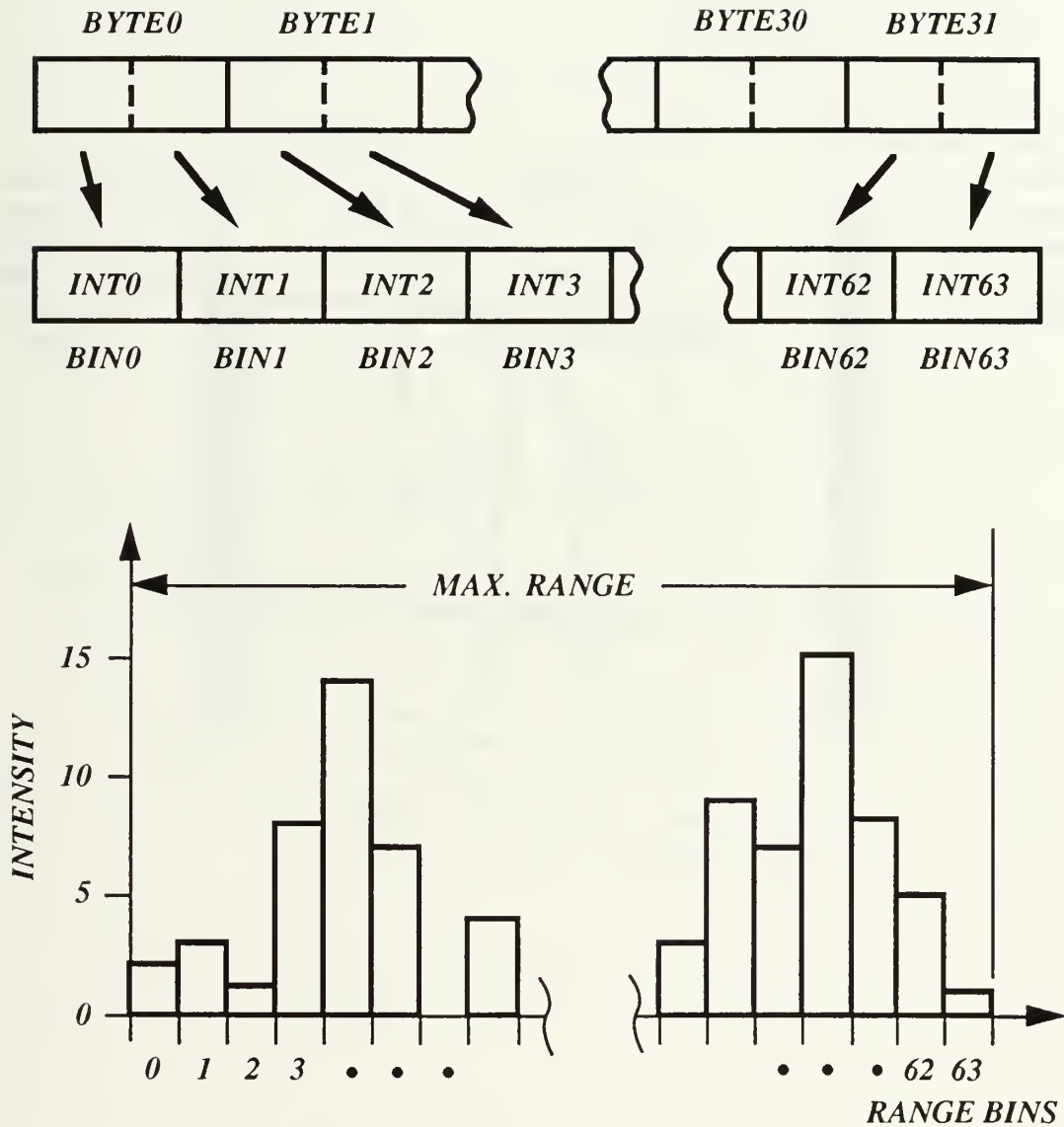Figure 4.2 Sonar Beam Shapes for the ST725 and ST1000 Sonars.

Figure 4.3 Scanline Data Structure of the ST725 Sonar.

ST-725 Sonar Display
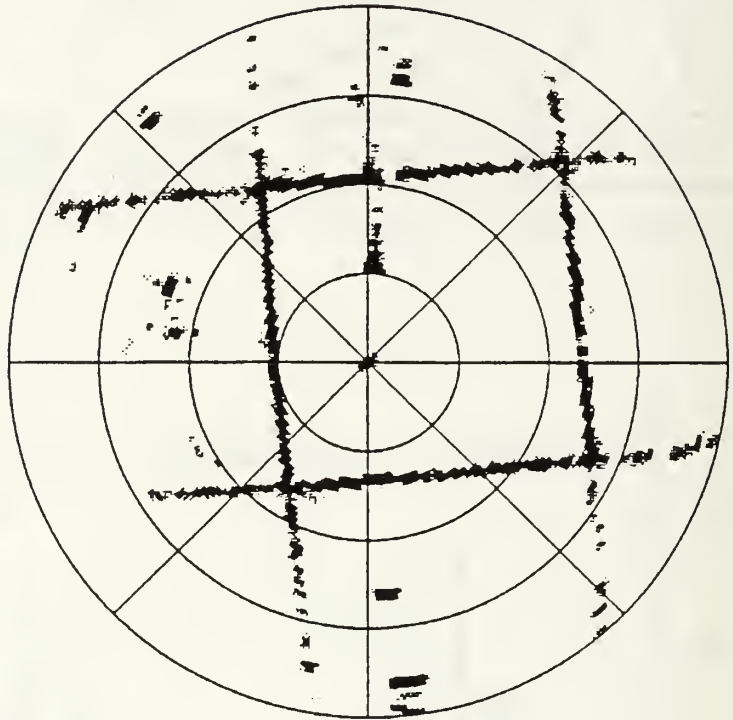Range : 6.0 m
Rings : 1.5 m
Res : High
Gain : 13
Thresh : 1

Figure 4.4 ST725 Sonar Display of the NPS Test Tank for Threshold = 1.

ST-725 Sonar Display
Range : 6.0 m
Rings : 1.5 m
Res : High
Gain : 13
Thresh : 10

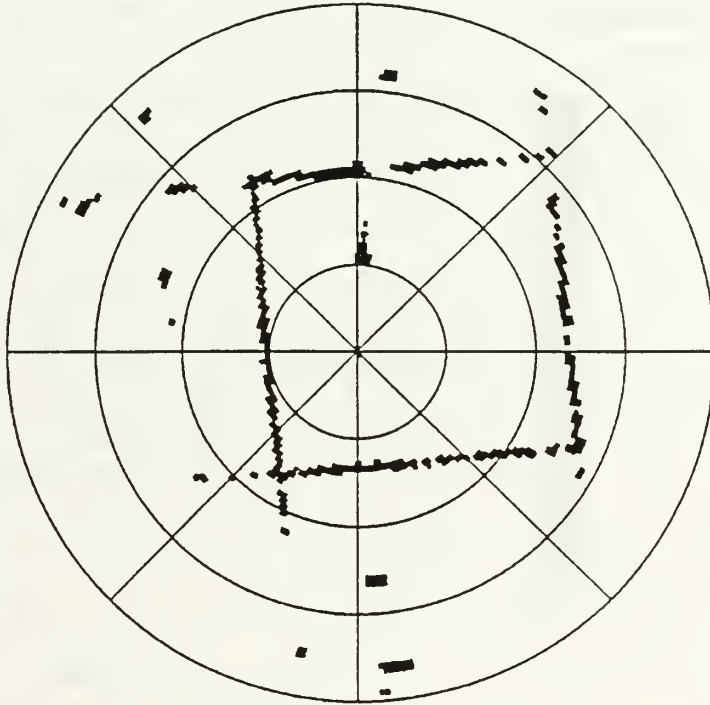Figure 4.5 ST725 Sonar Display of the NPS Test Tank for Threshold = 10.

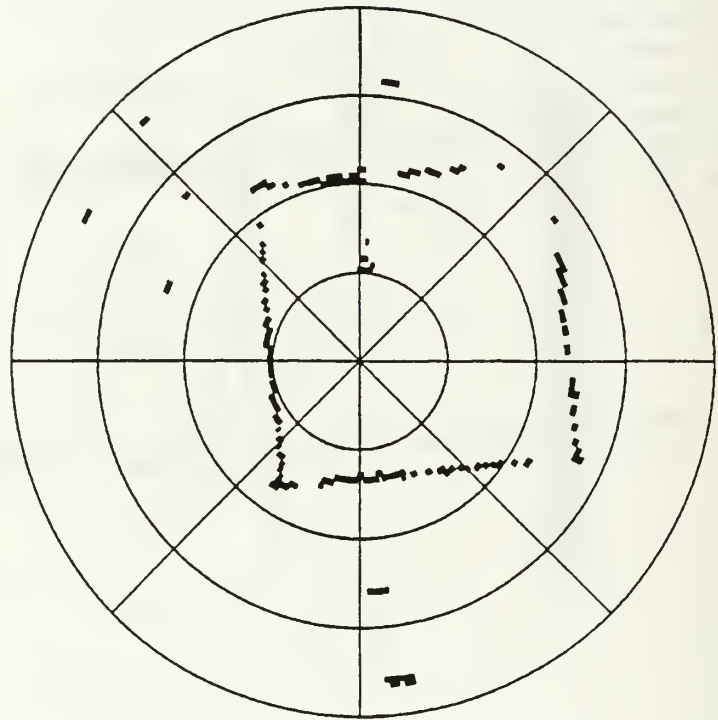Figure 4.6 ST725 Sonar Display of the NPS Test Tank for Threshold = 15.

ST-1000 Sonar Display
Range : 6.0 m
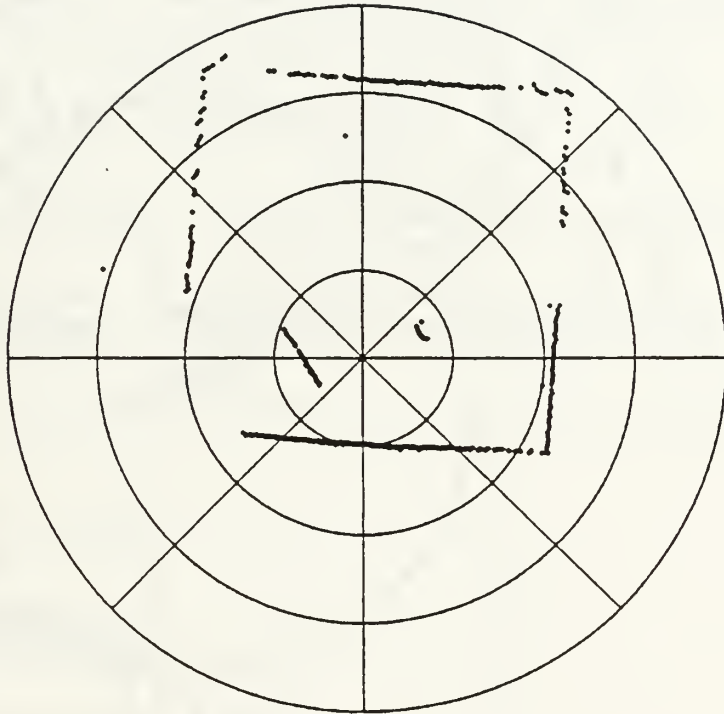Rings : 1.5 m
Res : High
Gain : 13
Thresh : 15

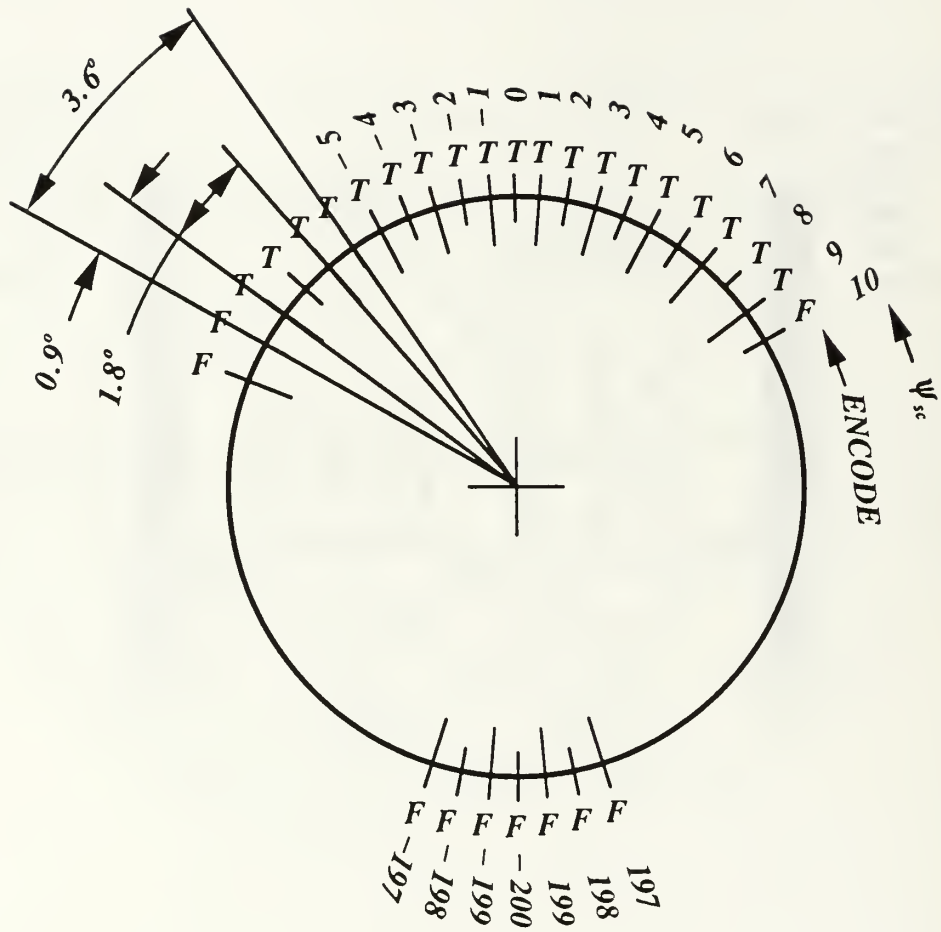Figure 4.7 ST1000 Sonar Display of the NPS Test Tank.

137

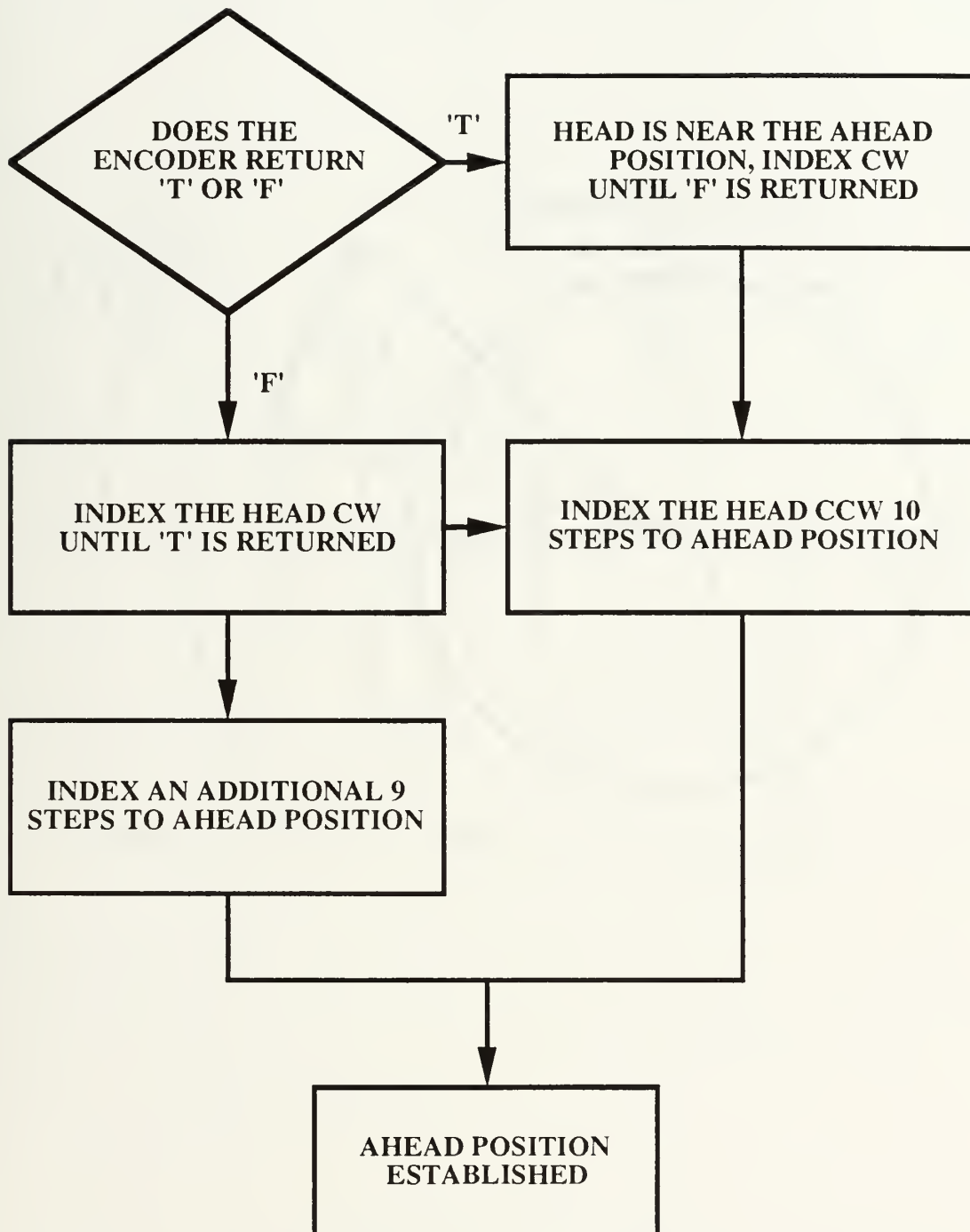Figure 4.8 Sonar Head Encoder Definitions.

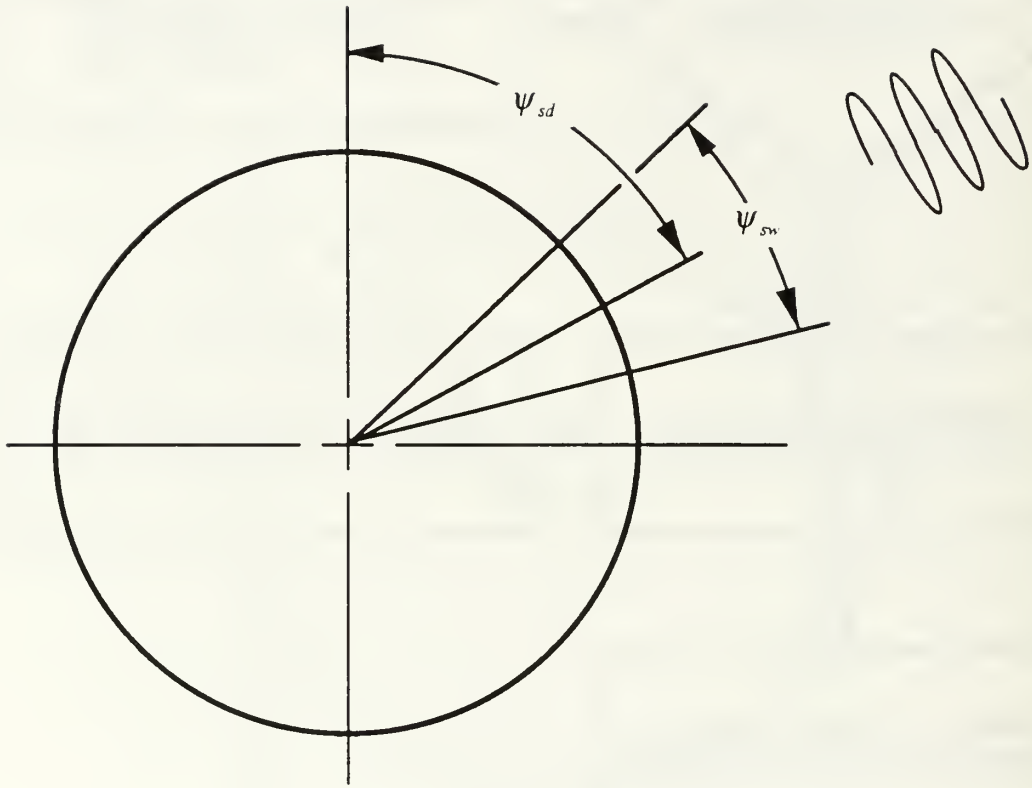Figure 4.9 Method to Align the Sonar Head to the Ahead Position.
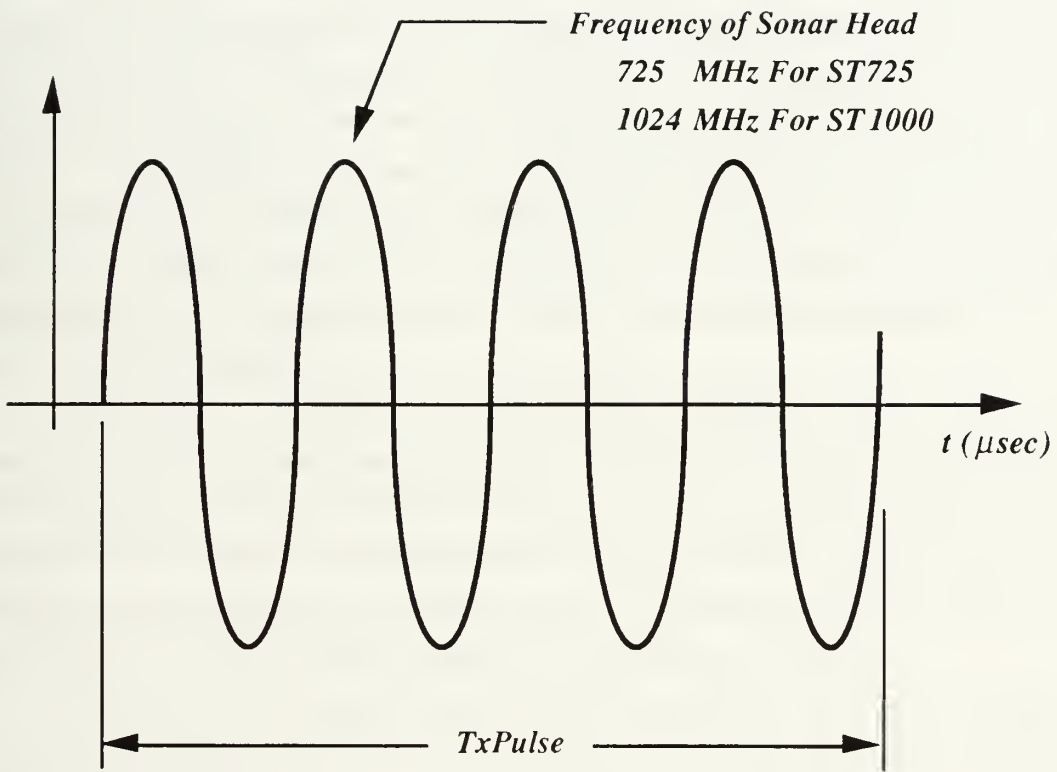
Figure 4.10 Sector Sweep Mode Variables.

Figure 4.11 Transmitted Pulse Description.

V. CONTROL SYSTEM EVALUATION

Results from a series of in-water experiments using the Phoenix underwater vehicle are presented in this chapter. All tests used the tri-level control structure described in Chapter III, and were performed in the NPS test tank. The series was conducted to evaluate the performance of sliding mode control designs for submergence, rotation, and longitudinal positioning of the vehicle to a prescribed stand-off distance from an object. These are typical maneuvers that an AUV will be required to perform for ocean intervention and inspection tasks. Since the high level controller must rely on a stable, and well tuned autopilot to perform maneuvers, the closed-loop servo level controllers must exhibit predictable and well understood behavior. This then frees the mission designer to focus on specific mission objectives.

The chapter opens with a discussion of the experimental setup, which includes descriptions of the test tank, computers and network, and is followed by a brief explanation of the steps taken to prepare the vehicle for testing. The second section addresses the issues surrounding the design of a software filter for depth rate estimation, using signals from the vehicle pressure transducer. This analysis leads to a determination of the best filter design as the basis for "in vehicle" implementation and experimental validation.

The next section contains results from submergence, rotation, and longitudinal motion control experiments. The effects of varying the sliding mode control gains and system parameter estimates for each control mode are presented and the best values to use are determined. The pertinent sections of the Strategic Level Prolog codes used for each experiment are given and explained. Following this are results from simultaneously submerging and rotating the vehicle to a specified depth and heading. The controller for this maneuver utilizes command generators, which will ensure that the desired position and heading will be achieved at the same time.

A. EXPERIMENTAL SETUP

1. NPS AUV Test Tank

All vehicle experiments were conducted in the NPS test tank, which is constructed of 1/4" steel plates, measures 20 feet square with an open top, and is filled with fresh water to a depth of five feet. The water is continuously circulated and chemically treated using readily available swimming pool pumps and filters. The tank stands on the floor and has two Plexiglas windows located on the sides for observation. To enable ease of vehicle manipulation by laboratory personnel, a cat walk is located across the top of the tank. Placement and retrieval of the vehicle from the water is achieved by using an electrically powered hoist that is attached to a movable carriage, which rolls along a beam that extends across the entire length of the tank. Gridlines are marked on the tank bottom and sides to provide a reference background to improve observation of vehicle motions.

2. Computer Network

The computer network configuration used for the experiments is shown in Figure 3.8. It consists of five nodes connected by thinwire ethernet for network communications using TCP/IP. This arrangement allows communications between any of the nodes at any time, even while the experiment is ongoing. Since two computers are needed to implement the tri-level control structure, the Execution Level, residing in the Gespac must communicate with the system running the Strategic and Tactical Levels (Sun SPARC). At the time these experiments were performed, only the Gespac computer was located onboard the vehicle and was connected to the rest of the network using a water-proof, thinwire connector.

For missions using the ST1000 sonar, an SGI Elan workstation is used for displaying, in real time, the sonar data collected in the Execution Level. Installed on the DOS PC is a cross-compiler/linker which converts C language source code files into executable images which will operate on OS9 based systems. It is here that the Execution Level software is compiled and linked. The source code is usually developed on the SGI Elan or equivalent system. A radio ethernet unit is used for network connections to the

144

Internet, and has made data transfer between facilities very fast and convenient, especially if software is developed on remote systems.

3. Pre-Mission Procedures

Before an experiment can be performed, the required software must be present in each computer and the network communications enabled. Since the Gespac computer has no hard disk, and uses a volatile ram disk for data and program storage, each time the system is powered, the Execution Level software must be reloaded. The data transfer is done through the network using ftp (File Transfer Program), usually from the PC, since it is there that the software was cross-compiled. Upon power up, the Gespac boots from a set of EPROM's located on the CPU board which contains the system boot file, operating system, and network software modules. Once the boot process is complete, a ramdisk is created and the network is automatically started allowing connections from remote systems possible. The SPARC station on the other hand, boots from it's internal hard drive, where the Strategic and Tactical Level software already resides, so no program loading is required.

The Execution Level program is not the only software that needs to be loaded into the Gespac. Various modules known as "device descriptors" must be also present to provide an interface between application programs and hardware devices. Descriptors for the sonar and DiveTracker serial ports, and configuration information for the creation of additional ramdisks are required. These modules are transferred along with the Execution Level software using an ftp script known as "exec.ftp", which allows all modules to be downloaded using a single command. Once this is complete, the Gespac system is ready for the Execution Level program to start. At this point, the procedures for mission execution outlined in Chapter III are performed.

B. SUBMERGENCE CONTROL

In this section, vehicle submergence control using the two vertical thrusters is discussed. Here, the term submergence control is used rather than "depth control", to distinguish between the use of vertical thrusters rather than fins, which in a normal flight mode would be used to control depth. No forward motion is assumed, so the control planes are ineffective in this scenario. The sensor used for this mode is a depth cell utilizing

145

a pressure transducer. Since proper positional control of the vehicle requires some form of rate feedback, a digital software filter must be designed to smooth the noise contaminated depth measurement and extract the rate of change of depth from this signal.

1. Filter Design

The purpose of the filter is twofold. Firstly, and most important, an estimate of the depth rate must be extracted from the primary sensory output, and secondly, depending on the circumstances, noise in the primary sensor must be smoothed. In this section, a discussion of the effects of filter bandwidth is given first, and examined through the use of hard experimental data from earlier maneuvers of the NPS Phoenix. Thus realistic noise levels are taken into account.

The filter model is based on a three state kinematic model for depth changing excited by acceleration noise,

$$
\begin{aligned}
\dot{\hat{z}}_1(t) &= \hat{z}_2(t) \\
\dot{\hat{z}}_2(t) &= \hat{z}_3(t) \\
\dot{\hat{z}}_3(t) &= q(t).
\end{aligned}
\tag{5.1}
$$

The states $\hat{z}_1(t)$, $\hat{z}_2(t)$, and $\hat{z}_3(t)$ are estimates of the position, velocity, and acceleration of the depth signal $z(t)$. The measurement equation with $z(t)$ being the depth signal is

$$
z(t) = \hat{z}_1(t) + v(t)
\tag{5.2}
$$

where $q(t)$ is taken to be the covariance of the system noise and $v(t)$ is the covariance of the measurement noise. These equations are the basis for the formulation of a Kalman filter (Gelb, 1988). The values of $q(t)$ and $v(t)$ must be carefully chosen so the bandwidth of the filter is appropriate for the nature of the depth signal. If the system noise was large, with low measurement noise, the filter response would be quite fast. Decreasing the system noise and increasing the measurement noise will create the opposite effect and will result in a sluggish slow responding filter. The depth signal does contain some noise, specifically due to quantization of the analog pressure transducer voltage to it's digital representation in

146

the controlling computer. Therefore the filter must be tuned for the quantization resolution of the analog to digital converter.

The state space representation of Eqns. (5.1) and (5.2) is

$$\dot{\hat{z}}(t) = A\hat{z}(t) + Bq(t) \qquad (5.3)$$

$$z(t) = C\hat{z}(t) + v(t), \qquad (5.4)$$

where

$$\hat{z}(t) \in \Re^{3 \times 1},$$

and the covariances,

$$\begin{Bmatrix} q(t) \\ v(t) \end{Bmatrix},$$

is assumed to be of zero mean, Gaussian white noise, uncorrelated, and

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

For implementation of the execution level control functions, data is sampled at a fixed rate, 10 Hz, and it follows that the discrete time approximation of Eqns. (5.3) and (5.4) is required. The discrete time formulation using a sampling time of 0.1 seconds is

$$\hat{z}_{k+1} = \Phi\hat{z}_k + \Gamma q_k \qquad (5.5)$$

$$z_k = H\hat{z}_k + v_k \qquad (5.6)$$

147

where

$$\Phi = \begin{bmatrix} 1.000 & 0.100 & 0.005 \\ 0.000 & 1.000 & 0.100 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}, \quad \Gamma = \begin{bmatrix} 0.0002 \\ 0.0050 \\ 0.1000 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

With the model defined, an optimal estimate of the state $\hat{z}_k$, conditioned upon data z_k, can be obtained using a Kalman filter as based on a recursive weighted least-squares solution. (Developed in more detail in (Gelb)). The equation for the state estimate is

$$\hat{z}_k = \bar{z}_k + L_k(z_k - H\bar{z}_k) \tag{5.7}$$

where $\bar{z}_k$ is the kinematic model based expected value of $\hat{z}_k$ conditioned upon a prior estimate $\hat{z}_{k-1}$ without correction from the measurement z_k. is obtained from

$$\bar{z}_k = \Phi\hat{z}_{k-1}, \tag{5.8}$$

and the correction is based on the residual $z_k - H\bar{z}_k$, multiplied by the optimal (minimum variance of the estimation error) gain.

The Kalman filter gain is then given by

$$L_k = M_k H^T (H M_k H^T + v_k)^{-1} \tag{5.9}$$

where M_k is the model based propagated state error covariance matrix

$$M_k = \Phi P_{k-1} \Phi^T + \Gamma q_{k-1} \Gamma^T, \tag{5.10}$$

and P_k is it's corrected value after measurement and is given by

$$P_k = M_k - M_k H^T (H M_k H^T + v_k)^{-1} H M_k \tag{5.11}$$

148

Inspection of Eqns. (5.9) and (5.10) shows that one matrix inverse computation involving L_k is required each time step to update the estimate of the state which can consume valuable computing time. Fortunately, experiments with real data have shown (Figure 5.1), that after an initial settling time, the gains reach steady state values and, as such, may be pre-computed and stored as data rather than requiring a step by step update.

Three filter designs have been studied and are presented to illustrate the effects of varying the system and measurement noise. The filters are applied to actual experimental data obtained from the depth cell for a dive of 2.0 feet, as shown in Figure 5.2. The assumed noise levels for the three designs are as follows

$$
\begin{array}{llll}
\text{Filter 1 (Slow)} & q_k = 0.1, & v_k = 10.0 \\
\text{Filter 2 (Medium)} & q_k = 0.1, & v_k = 0.01 \\
\text{Filter 3 (Fast)} & q_k = 10.0, & v_k = 0.001
\end{array}
$$

which lead to bandwidths of approximately 0.2, 0.6, and 1.7 Hz respectively. The associated steady state filter gains are:

$$
\text{Filter 1} \quad L = \begin{Bmatrix} 0.0887 \\ 0.0411 \\ 0.0095 \end{Bmatrix}, \quad
\text{Filter 2} \quad L = \begin{Bmatrix} 0.2544 \\ 0.3727 \\ 0.2731 \end{Bmatrix}, \quad
\text{Filter 3} \quad L = \begin{Bmatrix} 0.6042 \\ 2.7513 \\ 6.2909 \end{Bmatrix}
$$

Results showing the estimated depth and depth rate for Filter 1 is shown in Figure 5.3. Comparing the depth estimate with the raw signal show this filter is too slow as evident from the lags present. The response is slightly oscillatory and does not track the real signal very well, and the rate also shows some oscillatory behavior during the steady state interval. Figure 5.4 shows the response for Filter 2, and displays a marked improvement over the previous filter, because the depth estimate follows the raw signal very closely, although the rate estimate shows some sign of the sensor noise. Filter 3, is shown in Figure 5.5, and again the depth estimate closely follows the raw signal, but the rate estimate is very poor since the quantization noise is strongly imparted to this estimate.

To more fully understand the performance of the three filters, a frequency response analysis has been performed. Figures 5.6 and 5.7 show the log-magnitude and phase angle plots for the depth rate estimate of the three filters. It can be seen from the plots that Filters 1, 2, and 3 exhibit increasing bandwidths. Although Filter 3 has the largest bandwidth it is

so large that it is faithfully reproducing the noise of the sensor and should not be used on these grounds. The Medium filter has a smaller bandwidth and is much less affected by the noise, and is the filter of choice for depth control. Since the filter is relatively slow, it will be an important factor in the sliding mode control design to follow.

2. Submergence Control Simulation

Now that three filters have been designed, each may be evaluated for estimating the depth and depth rate using a dynamic simulation of the Phoenix vehicle under submergence control. A sliding mode controller will be designed which will require the depth and depth rate for feedback. The control inputs are the two vertical thrusters, which, in the simulation, act together as a single input causing the vehicle to be confined to one direction of travel. Motion occurs along the body-fixed z-axis, which is also assumed to remain parallel to the global Z-axis and the gravity vector. With these restrictions, and no vertical current component such that $\dot{w}(t) = \ddot{z}(t)$, the continuous time dynamic model for depth motion becomes

$$M_z \ddot{z}(t) + b_z \dot{z}(t)|\dot{z}(t)| + F_B = 2\alpha_z v(t)|v(t)| \qquad (5.12)$$

where

$$M_z = m + m_{az}.$$

and m_{az} is the heave added mass, and α_z is a coefficient relating the square of the vertical thruster motor voltage, $v(t)$, to the force developed. The thruster effectiveness is doubled since both thrusters are operated simultaneously and receive equal control voltages. F_B is any net force acting on the vehicle caused by a deviation from neutral buoyancy, and b_z is the coefficient of square law drag in the vertical direction.

The position and rate errors for submergence control are defined as

$$\tilde{z}(t) = z_{com}(t) - z(t)$$

$$\dot{\tilde{z}}(t) = \dot{z}_{com}(t) - \dot{z}(t). \qquad (5.13)$$

150

where the subscript "*com*" refers to the commanded depth or depth rate. The sliding surface for a single input/output control design is a scalar equation given by

$$\sigma(t) = \dot{\tilde{z}}(t) + \lambda \tilde{z}(t) \tag{5.14}$$

where λ is scalar quantity analogous to the matrix S_2 described in Chapter II. The control voltage can be expressed as

$$v(t) = \sqrt{|\gamma(t)|} \, sgn(\gamma(t)) \tag{5.15}$$

where

$$\gamma(t) = \frac{\hat{M}_z}{2\hat{\alpha}_z}\left(\ddot{z}_{com}(t) + \frac{\hat{b}_z \dot{z}(t)|\dot{z}(t)|}{\hat{M}_z} + \frac{\hat{F}_B}{\hat{M}_z} + \lambda \dot{\tilde{z}}(t) + \eta sat(\sigma(t)/\phi)\right).$$

Since only the depth signal is available for measurement and we wish to reduce the computation time for the state estimate, the constant gain filter will be used to supply depth rate feedback for the sliding mode controller. With this, the control equation uses the estimates of depth and depth rate as follows:

$$\gamma(t) = \frac{\hat{M}_z}{2\hat{\alpha}_z}\left(\ddot{z}_{com}(t) + \frac{\hat{b}_z \dot{\hat{z}}(t)|\dot{\hat{z}}(t)|}{\hat{M}_z} + \frac{\hat{F}_B}{\hat{M}_z} + \lambda \dot{\hat{\tilde{z}}}(t) + \eta sat(\hat{\sigma}(t)/\phi)\right) \tag{5.16}$$

where

$$\hat{\sigma}(t) = \dot{\hat{\tilde{z}}}(t) + \lambda \hat{\tilde{z}}(t) \tag{5.17}$$

and

$$\hat{\tilde{z}}(t) = \hat{z}_{com}(t) - \hat{z}(t) \tag{5.18}$$

Figure 5.8 shows the response to a step command of 4.0 feet in depth using the three filters previously designed. Depth rate and commanded control voltage levels are shown in Figures 5.9 and 5.10 respectively. The control gains λ, η, and the boundary layer thickness ϕ were varied until the most satisfactory response was observed as a

151

balance between tracking error and control activity. The appropriate gains for each filter were

$$\text{Filter 1} \quad \lambda = 0.1 \ , \ \eta = 0.1 \ , \ \phi = 0.7$$

$$\text{Filter 2} \quad \lambda = 0.2 \ , \ \eta = 0.1 \ , \ \phi = 0.2$$

$$\text{Filter 3} \quad \lambda = 0.5 \ , \ \eta = 0.1 \ , \ \phi = 0.1$$

The slower the filter, the lower the control gains were to compensate for filter lags, and is reflected by the speed of response of the respective systems. The boundary layer thickness also had to be broadened with decreasing filter bandwidth. If high gains and a small boundary layer thickness is used, the system will become unstable for slower filters because of the time lags.

The system was most sensitive to boundary layer thickness. For small errors, $\sigma(t)$ is within the boundary layer, and with a small value of ϕ, the switching function slope is sharp, providing high gain control for small errors. The switching curve is shown in Figure 5.11 contrasting the wide boundary layer, $\phi = 1.0$, with a narrow boundary layer, $\phi = 0.1$. These three cases have shown that the selection of the control gains is highly dependent on the bandwidth of the Kalman filter used. Therefore, the controller must be tuned for each specific filter design.

3. Variable Boundary Layer Formulations

To avoid the task of selecting unique gains for each sliding mode controller, a method to vary the values of the gains can be used. Sensitivity to the boundary layer thickness can be overcome by formulating ϕ as a function of $\sigma(t)$. It is desired that the gain is low near $\sigma(t) = 0$ and high for larger values of $\sigma(t)$. One way to accomplished this is using a linear function, such as

$$\phi(\sigma) = \phi_{max} - \frac{(\phi_{max} - \phi_{min})\sigma(t)}{\sigma_{max}}, \tag{5.19}$$

or a non-linear function

$$\phi(\sigma) = \frac{\phi_{max}}{\frac{(\phi_{max} - \phi_{min})\sigma(t)}{\phi_{min}\sigma_{max}} + 1},$$
(5.20)

where σ_{max} is the maximum value $\sigma(t)$ will assume, or in the case of a step input, the value of $\sigma(t)$ at maximum error, $\sigma(0)$. The curves for the linear and non-linear functions are shown in Figure 5.12. ϕ_{max} was chosen as 1.0 with $\phi_{min} = 0.1$. Returning to Figure 5.11 the two switching curves for the two functions for $\phi(\sigma)$ are shown. From this, it is easily seen that the gain will be equivalent to using a constant value of $\phi(\sigma) = 1.0$ near $\sigma(t) = 0$ but the gain increases faster with increasing $\sigma(t)$ than if a value of $\phi(\sigma) = 1.0$ is used throughout. The increase in gain is even more accentuated using the non-linear function. This method can also be extended to adapt the values of λ and η to reduce the amount of controller tuning required.

To evaluate the effectiveness of the variable boundary layer, the sliding mode controller Eqn. (5.15) was modified to use Eqn. (5.19) for the calculation of ϕ. For purposes of comparison, the response to a step input of 1.0 foot using a constant, $\phi = 0.1$, and variable boundary layer is shown in Figure 5.13. It can be seen that when using a constant value of ϕ, a limit cycle instability appears once the set point is reached. However, the response is stabilized if a variable boundary layer is used, decreasing the gain near the set point were the tracking errors are small.

4. Evaluation of Control Law Robustness Through Parameter Sensitivity (Step Response)

With the computer simulation studies completed, the controller designed above was implemented on the Phoenix in the test tank. Each control gain or vehicle parameter was individually varied from the nominal values listed in Table 5.1, which were obtained from the simulations to determine their effectiveness on vehicle performance. The vertical thrusters were the only actuators used and the forward speed was zero. Each test used a step input of depth error with $\dot{z}_{com} = \ddot{z}_{com} = 0$. The test cases were separated into four distinct series which are shown in Tables 5.2 through 5.5.

153

Table 5.1 Nominal Controller Gains/Vehicle Parameters

$\hat{M}_z$	$\hat{\alpha}_z$	η	λ	ϕ	$\hat{b}_z$
$\dfrac{lb-sec^2}{ft}$	$\dfrac{lb}{V^2}$	$\dfrac{ft}{sec^2}$	$\dfrac{rad}{sec}$	$\dfrac{ft}{sec}$	$\dfrac{lb-sec^2}{ft^2}$
27.0	0.004	0.10	0.2	0.2	28.8

Table 5.2 Controller Gains/Vehicle Parameters for Vertical Damping, $\hat{b}_z$, Test

	$\hat{M}_z$	$\hat{\alpha}_z$	η	λ	ϕ	$\hat{b}_z$	z_{com} (ft)
1	27.0	0.004	0.10	0.2	0.2	**10.0**	3.0
2	27.0	0.004	0.10	0.2	0.2	**28.8**	3.0
3	27.0	0.004	0.10	0.2	0.2	**45.0**	3.0
4	27.0	0.004	0.10	0.2	0.2	**90.0**	3.0

Table 5.3 Controller Gains/Vehicle Parameters for Thruster Effectiveness, $\hat{\alpha}_z$, Test

	$\hat{M}_z$	$\hat{\alpha}_z$	η	λ	ϕ	$\hat{b}_z$	z_{com} (ft)
1	27.0	**0.002**	0.10	0.2	0.2	28.8	2.0
2	27.0	**0.004**	0.10	0.2	0.2	28.8	2.0
3	27.0	**0.008**	0.10	0.2	0.2	28.8	2.0

Table 5.4 Controller Gains/Vehicle Parameters for Switching Gain, η, Test

	$\hat{M}_z$	$\hat{\alpha}_z$	η	λ	ϕ	$\hat{b}_z$	z_{com} (ft)
1	27.0	0.004	**0.05**	0.2	0.2	28.8	2.0
2	27.0	0.004	**0.20**	0.2	0.2	28.8	2.0
3	27.0	0.002	**0.20**	0.2	0.2	28.8	2.0

Table 5.5 Controller Gains/Vehicle Parameters for Sliding Surface Position
Error Coefficient, λ, Test

	$\hat{M}_z$	$\hat{\alpha}_z$	η	λ	ϕ	$\hat{b}_z$	z_{com} (ft)
1	27.0	0.004	0.10	**0.1**	0.2	28.8	2.0
2	27.0	0.004	0.10	**0.2**	0.2	28.8	2.0
3	27.0	0.004	0.10	**0.4**	0.2	8.8	2.0

The of portion of the Strategic Level Prolog used for the submergence control
experiments is given below.

```
                    .
                    .
execute_phase(2)    :- exec_next_setpt_data(X), exec_submerge(X),
                    exec_start_depth_error_filter(X), exec_start_timer(X),
                    repeat, phase_completed(2).


phase_completed(2) :- exec_sleep(1,X), ask_depth_reached(X), X==1,
                    asserta(complete(2)).


phase_completed(2) :- ask_time_out(X), X==1, exec_surface(X),
                    printsc('PHASE 2 ABORTED DUE TO TIME OUT!'),
                    repeat, ask_surface_reached(X), X==1,
                    asserta(abort(2)).
                    .
                    .
```

Note that vehicle initialization is performed during phase (1), and the submergence test
occurs during phase (2). Running this section of code instructs the Tactical Level to send
the depth set point, control mode, and phase timeout information to the Execution Level
that is specified in the mission file (exec_next_setpt_data). Once this has been done, the
vehicle is commanded to submerge (exec_submerge) and have the depth error filter started
in the Execution Level (exec_start_depth_error_filter). At this point, the timeout counter is
started in the Tactical Level and the query predicate, "ask_depth_reached" is repeatedly

155

executed until the commanded depth has been reached. If the set point is not attained before the time out , an abort is ordered and the vehicle surfaces.

The results for varying the vertical damping coefficient, $\hat{b}_z$, are shown in Figures 5.14 through 5.16. It is evident from the depth response that the damping coefficient used in the controller directly affects the speed of response. Using relatively small values results in lower command voltages on the thrusters since the controller is assuming a very lightly damped system exists. Increasing the damping to relatively large values produces the opposite effect and the control voltages are very high. In fact, the results from the fourth test, $\hat{b}_z = 90.0 \ lb - sec^2 \ / \ ft^2$, caused the thrusters to saturate, and the vehicle struck the bottom of the tank and bounced upwards before being controlled to the set point.

The next series involved varying another vehicle parameter, the thruster effectiveness coefficient, $\hat{\alpha}_z$, and these results are shown in Figures 5.17 and 5.18. Overdriving of the thruster can be clearly seen in the upper trace of Figure 5.18. Using a larger value for $\hat{\alpha}_z$ causes a much softer control action since a very "strong" thruster is assumed, as shown in the lower trace of the figure, and was so small that the vehicle had difficulty reaching the set point.

The switching gain, η, was the next value to be investigated and these results appear in Figures 5.19 and 5.20, showing the effects of varying the control gain as opposed to vehicle parameters. A limit cycle is seen for the largest value of η since the slightest error is strongly amplified. The third test shows an even larger amplitude limit cycle since the thruster effectiveness, $\hat{\alpha}_z$, was lowered from the nominal resulting in an even higher gain. Using a small gain provided a very soft control preventing the set point to be reached.

The final series analyzed the effectiveness of the sliding surface position error coefficient, λ, shown in Figures 5.21 and 5.22. These results followed those of the switching gain series since both the values of λ and η affect the overall gain of the control system. However, the contribution of λ, which is part of the sliding surface definition, to the gain is attenuated by the saturation, ϕ. This is reflected by smaller amplitudes of control action oscillation compared to those generated by η.

The oscillatory control voltage behavior seen in the previous results is not solely caused by high gain. One reason is the fact that in the steady state, there is virtually no damping force present due to the low velocity. Another cause is the sensitivity of the square root function in Eqn. (5.15). For values of $x(t)$ near the origin, which reflect small errors, the control voltage is more strongly amplified than in other regions. The coarse

156

discretization resolution from the depth sensor channel is causing the velocity estimate from the filter to be very noisy and the controller is excited by this. The resolution of the depth measurement was only 0.0182 feet which was mainly due to the large operating range of the depth cell (0 - 37.0 ft). The A/D converter had a digital resolution of 0-2048 for an input voltage range of 0 - 10 Vdc which is proportional to the depth range. This relatively coarse resolution can be improved by either using a higher resolution A/D unit or selecting a depth cell with a smaller operating range and comparable output voltage.

5. Submergence Integral Control

So far the analysis has assumed that the value of the net buoyancy force F_B in Eqn. (5.12) is zero. In practice, this is usually not the case since achieving complete neutral buoyancy of an actual vehicle is very difficult given the variability of water temperatures, salinity, and the hull compression effects of increased depth. Also it is usually advisable to ballast an underwater vehicle slightly positive (light) since if the power or control systems fail, recovery will be possible. The consequence of a positively or negatively ballasted vehicle is that a steady-state offset from the commanded depth will result. Figure 5.23 shows the simulated depth response for a light, heavy and a neutrally buoyant vehicle for a commanded depth of 2.0 feet. The disturbance can be satisfactorily handled using integral control.

The sliding surface equation, (5.14) can be modified to include an integral term given by

$$\sigma(t) = \dot{\tilde{z}}(t) + \lambda_1 \tilde{z}(t) + \lambda_2 \int \tilde{z}(t) dt, \qquad (5.21)$$

where the integral term should not exceed a pre-defined maximum value, I_{max}, such that

$$| \int \tilde{z}(t) dt | \leq I_{max}.$$

Limiting the growth is referred to as anti-reset windup, and will be discussed later in this section relating to position control performance.

The new control for thruster volts becomes

$$v(t) = \sqrt{|\gamma(t)|}\, sgn(\gamma(t)),$$ (5.22)

where

$$\gamma(t) = \frac{\hat{M}_z}{2\hat{\alpha}_z}\left(\ddot{z}_{com}(t) + \lambda_1 \dot{\hat{z}}(t) + \lambda_2 \hat{z}(t) + \eta \tanh(\hat{\sigma}(t) / \phi)\right) + \frac{1}{2\hat{\alpha}_z}\left(\hat{b}_z \dot{\hat{z}}(t)\left|\dot{\hat{z}}(t)\right| + \hat{F}_B\right)$$

The addition of the term $\lambda_2 \hat{z}(t)$ will remove any steady state depth errors from the response, but for best results, integral control should be applied only during certain phases of the maneuver.

Figure 5.24 shows the simulated depth response using integral control from time $t = 0$ until $t = 120$ seconds. Since the position error is large at the start of the maneuver, the integral grows very rapidly and contributes a large control force to submerge the vehicle. The control action persists beyond the set point, resulting in an overshoot that is especially large for the heavy case, and regardless of the anti-reset windup, an overshoot must occur to subtract from the growth of the integral. One solution to this problem is to activate integral control only after a steady state offset is detected, as shown in Figure 5.25 (i.e. error closure has not been achieved within a preset time). Using this approach greatly reduces any overshoot for the heavy case and for a light vehicle, no overshoot will occur. It should be noted that if the anti-reset windup limit, I_{max}, is not large enough to overcome the net buoyancy force, F_B, a steady state error will exist proportional to the difference in these values.

The in water results of the NPS Phoenix are shown in Figures 5.26 and 5.27 for a commanded depth of 2.0 feet. The depth response for a lightly, partially lightly, and heavily ballasted vehicle is shown, along with the commanded control volts to the vertical thrusters. Varying the buoyancy was achieved by adding lead weights on top of the hull to increase the weight for each respective experiment. If the commanded depth was not achieved within 40 seconds, integral control was activated and the errors were significantly reduced as shown. The control voltages reflect the buoyancy condition that exists since the steady state voltage for the light cases is positively biased providing an overall downward force, while the opposite is true for the heavy case, which commands an upward force. If

158

the vehicle is neutrally buoyant, the set point depth will be achieved within the specified time and no integral action will be required and is not activated.

6. Command Generators for Precision Depth Maneuvering

So far all depth changes have been performed using step inputs of position error for the control input, leaving the velocity and acceleration commands zero, leading to a large initial thrust requirement and actuator saturation. Using softer control gains can alleviate this, but usually causes very sluggish response with poor steady-state tracking precision and disturbance rejection. To control the transient response more precisely while maintaining adequate bandwidth, two forms of command generators have been formulated which consist of the desired position, velocity, and acceleration of the vehicle depth as a function of time. Form 1 requires that the maximum velocity and acceleration be specified, along with the desired depth change and the time to complete the maneuver is dependent on these values. Form 2 allows specification of both the final depth and time to complete the maneuver but due to the added constraint, the maximum velocity can not be chosen arbitrarily. For the submerge motion, the two command generators are taken from

$$[z_{com}, \dot{z}_{com}, \ddot{z}_{com}] = G(z_0, z_f, \dot{z}_{max}, \ddot{z}_{max}, T_0) \quad \text{(Form 1)}$$

or

$$[z_{com}, \dot{z}_{com}, \ddot{z}_{com}] = G(z_0, z_f, \ddot{z}_{max}, T_0, T_f) \quad \text{(Form 2)}$$

where a fifth order, zero jerk profile has been chosen so that the maximum acceleration and the bandwidth capacity of the vehicle is not overly exceeded and z_0, T_0 is the initial depth and starting time while z_f, T_f is the final desired depth and time at the end of the maneuver. Using this technique significantly reduces the occurrence of actuator saturation, since the errors are continuously small throughout the depth change phase. A detailed description and the equations used are presented in Appendix B.

Experimental results for both the command generator formulations will now be presented. The conditions were identical to those used for the step response tests except that the initial depth before the maneuver begins was not at the surface but approximately

0.75 feet under. By doing this, any surface effects are minimized that may interfere with the submerging ability of the vehicle.

Figures 5.28 through 5.30 show the results using command generator Form 1. The nominal control parameters in Table 5.1 were used with a commanded depth 2.0 feet and $\dot{z}_{max}$, and $\ddot{z}_{max}$ were 0.02 ft/sec and 0.006 ft/sec^2 respectively. It can be seen from the position response, that the commanded depth is not being tracked very well until the very end of the maneuver. The control voltage trace shows a great deal of oscillation caused not only by the velocity noise, but when using a command generator, the position and velocity errors are small throughout the maneuver, giving rise to the high gain effect of the square root function discussed earlier. Since using a higher gain controller to improve tracking performance caused even more control action, it was decided to relax the command generator specification which is presented next.

Results from Form 2 of the command generator is shown in Figures 5.31 through 5.33. where a final depth of 3.0 feet was specified and the elapsed time to complete the maneuver was 60.0 seconds. It was desired to use the smallest possible $\ddot{z}_{max}$ given the distance and time constraints, which from Appendix B is

$$\ddot{z}_{max} = 8 \frac{|z_f - z_0|}{(T_f - T_0)^2},$$ (5.23)

and computes gentler profiles for the vehicle to follow and results in a much improved response. It should be noted that a slight overshoot occurs near the end of the maneuver due to a heavy vehicle that day but it is quickly removed by the integral control action which was activated once the error was detected. The velocity command was also tracked very accurately.

7. Conclusions From Submergence Control Studies

The results presented have shown that the depth of the Phoenix can be precisely controlled using Sliding Mode control of the vertical thrusters. Although discretization noise from the depth cell produced unfavorable control action, the overall performance is exceptional and this problem can be easily rectified using a better sensor. Applying integral control has shown to be very effective in compensating for any deviation of the vehicle from neutral buoyancy, especially if activated at the proper time. Using command

generators provided extremely precise, time based maneuvering, even in the presence of buoyancy disturbances.

C. HEADING CONTROL

Described in this section is the development of the sliding mode heading control equations for the Phoenix vehicle. The two lateral thrusters are the only actuators used and the nominal forward speed is zero. All experimental data was collected in the NPS AUV test tank using the submergence controller developed previously for depth control. The sensors used for control feedback are the directional and yaw rate gyroscopes. The first section will present the control algorithm development, followed by experimental results of step response for heading command. The final section shows the results for command generator inputs to the controller specifying a time based desired angular position, velocity, and acceleration for a maneuver.

1. Vehicle Model for Heading Control

The vehicle dynamics for rotation about the body-fixed z-axis (yaw) using both the bow and stern lateral thrusters can be described by the following differential equation for the continuous time, continuous state evolution:

$$I_z \ddot{\psi}(t) + b_\psi \dot{\psi}(t)\left|\dot{\psi}(t)\right| = 2\alpha_\psi v(t)|v(t)| + \delta f_\psi(t) \tag{5.24}$$

where

$$I_z = I_{zz} + I_{zza}$$

and I_{zza} is the added inertia about the z-axis, α_ψ is a coefficient relating the square of the lateral thruster motor voltage, $v(t)$, to the moment developed, b_ψ is the coefficient of rotational square law drag, and $\delta f_\psi(t)$ is the force error. It has an upper bound, γ, such that $L_l(\delta f_\psi) \leq \gamma$, where L is the L_l norm of $\delta f_\psi(t)$. Since this development assumes motion restricted to a horizontal plane with depth control assumed by a separate controller, $r(t) = \dot{\psi}(t)$ and $\dot{r}(t) = \ddot{\psi}(t)$.

161

The angular position and rate tracking errors are defined as

$$\tilde{\psi}(t) = \psi_{com}(t) - \psi(t)$$

(5.25)

$$\dot{\tilde{\psi}}(t) = \dot{\psi}_{com}(t) - \dot{\psi}(t),$$

with the sliding surface given by

$$\sigma_\psi(t) = \dot{\tilde{\psi}}(t) + \lambda_\psi \tilde{\psi}(t).$$

(5.26)

The sliding mode control law for rotational control is given by

$$v(t) = \sqrt{|\gamma(t)|} sgn(\gamma(t))$$

(5.27)

where

$$\gamma(t) = \frac{\hat{I}_z}{2\hat{\alpha}_\psi} \left(\ddot{\psi}_{com} + \lambda_\psi \dot{\tilde{\psi}}(t) + \frac{\hat{b}_\psi}{\hat{I}_z} \dot{\psi}(t)|\dot{\psi}(t)| + \eta_\psi tanh(\sigma_\psi(t)/\phi_\psi) - \frac{\delta f_\psi(t)}{\hat{I}_z} \right).$$

Since both the bow and stern lateral thrusters are equidistant from the vehicle center and are of equal size and power, the magnitude of the solution to (7.2) is commanded to each thruster except for a difference in sign such that

$$v_{blt}(t) = + v(t)$$

and

$$v_{slt}(t) = - v(t)$$

where $v_{blt}(t)$ and $v_{slt}(t)$ are the voltage commands to the bow and stern lateral thrusters respectively.

2. Heading Control Experimental Results (Step Response)

Results obtained from in water testing of the heading control will now be presented. Four majors series are shown with the first being the response to varying and observing the effects of the switching gain η_ψ, the second varying the switch saturation level ϕ_ψ, the third involves adding an output filter and dead zone to the control voltage command for further smoothing. The final series presents the results of using time based command generators for precision control to a prescribed heading. All step response tests used $\dot{\psi}_{com} = \ddot{\psi}_{com} = 0$.

The Prolog code used to perform this is given by

.

.

```
execute_phase(3)    :- exec_next_setpt_data(X), exec_submerge(X), exec_rotate(X),
                       exec_start_timer(X), repeat, phase_completed(4).


phase_completed(3) :- exec_sleep(1,X), ask_depth_reached(X), X==1,
                      ask_heading_reached(X), X==1,
                      asserta(complete(3)).


phase_completed(3) :- ask_time_out(X), X==1, exec_surface(X),
                      printsc('PHASE 3 ABORTED DUE TO TIME OUT!'),
                      repeat, ask_surface_reached(X), X==1,
                      asserta(abort(3)).
```

.

.

Each test is started by submerging the vehicle to a depth of 2.5 ft using the submergence control method outlined previously to minimize surface effects during rotation (performed during phase 2). Phase 3 begins with the Tactical Level sending the depth and heading set points, etc. to the Execution Level. If the commanded heading is not reached before the timeout, an abort is declared and the vehicle surfaces.

A nominal set of control gains and vehicle parameters were obtained through computer simulation and parameter identification of the rotational motion which are given in Table 5.6 below (Torsiello, 1994).

Table 5.6 Nominal Parameters and Gains for Heading Control

Parameter/ Gain	$\hat{I}_z$	$\hat{b}_\psi$	$\hat{\alpha}_\psi$	λ_ψ	η_ψ	ϕ_ψ
Unit	$lb{-}ft{-}sec^2$	$lb{-}ft{-}sec^2$	$\dfrac{lb-ft}{V^2}$	$\dfrac{rad}{sec}$	$\dfrac{rad}{sec^2}$	$\dfrac{rad}{sec}$
Value	80.00	55.87	0.006	0.200	0.010	0.010

The first series of tests were run to study the effects of different values of the switching gain η_ψ. The parameters used in the controller are listed in Table 5.7 and the results shown in Figures 5.34 through 5.36 for a commanded heading of $\pi/2$ radians from an initial heading of 0.0.

Table 5.7 Parameters for Switching Gain Series

Case #	$\hat{I}_z$	$\hat{b}_\psi$	$\hat{\alpha}_\psi$	λ_ψ	η_ψ	ϕ_ψ
1	80.00	55.87	0.006	0.200	**0.010**	0.010
2	80.00	55.87	0.006	0.200	**0.050**	0.010
3	80.00	55.87	0.006	0.200	**0.100**	0.010

Figure 5.34 shows the angular position response for the three values of η_ψ listed above. It can be seen that increasing the gain naturally causes the speed of response to increase as well. The time to reach the set point for the smallest gain, $\eta_\psi = 0.01$ rad/sec^2 is approximately 50.0 seconds and half that, only 25.0 seconds using the largest, $\eta_\psi = 0.10$ rad/sec^2. The fast response does come with a cost since the larger gain causes control action chatter once the steady state heading has been achieved, as seen in the lower trace of Figure 5.36, while the control action provided by the softer gain is relatively smooth. The limit cycle behavior is caused by the same reasons as explained for submergence control, namely the low damping and the sensitivity from the square root function in Eqn. (5.27).

164

Since the value of the saturation gain, ϕ_ψ, in the first series of experiments was small giving a relatively sharp switching action, it was decided to increase this value in order to influence the amount of chattering in the steady state while maintaining a respectable response time. The parameters used for this series are in Table 5.8 and the results shown in Figures 5.37 through 5.39.

Table 5.8 Parameters for Saturation Series

Saturation	$\hat{I}_z$	$\hat{b}_\psi$	$\hat{\alpha}_\psi$	λ_ψ	η_ψ	ϕ_ψ
1	80.00	55.87	0.006	1.000	0.090	**0.100**
2	80.00	55.87	0.006	1.000	0.090	**0.200**
3	80.00	55.87	0.006	1.000	0.090	**0.400**
W/F	80.00	55.87	0.006	1.000	0.090	**0.400**

It can be seen that the differences in saturation level have only a slight effect on the response time but does follow a trend of increasing switch sharpness (i.e. decreasing ϕ_ψ) produces a faster system. Observing the control voltage in Figure 5.39, both the frequency and amplitude of the control voltage is decreased by increasing ϕ_ψ. This is attributed to the fact that with a larger saturation gain, larger values of $\sigma_\psi(t)$ which reflect larger errors, will be attenuated by the *sat* function over a much larger range which decreases the control action in the steady state.

Another approach was attempted to reduce the chattering by post filtering the control voltage commands. A first order digital filter of the form

$$v_{f(k+1)} = e^{-T/\tau}v_{f(k)} + (1 - e^{-T/\tau})v_k. \tag{5.28}$$

was used. The value of the sampling time, T, was the usual 0.1 seconds and the time constant, τ, was 0.45 seconds. With filtering, the resulting control voltage is shown in the lower trace labeled "$\phi_\psi = 0.40$ W / F" which still exhibits a limit cycle but is of smaller amplitude, and the frequency is greatly reduced. The results show that increasing the saturation to 0.40 *rad / sec* and using a post filter provides an acceptable response time while significantly reducing the switch frequency and amplitude.

To further reduce the actuator activity in the steady state, a dead zone on the command voltage was applied, that causes any voltage commands below a certain level to

165

be ignored and a value of zero sent to the thrusters. The following results show the comparison of a controller using only the post filter (Case 1) to one using the filter and a control dead zone of ±4.0 volts (Case 2). Table 5.9 gives the parameters used, and the response is shown in Figures 5.40 through 5.42.

Table 5.9 Parameters for Dead Zone Series

	$\hat{I}_z$	$\hat{b}_\psi$	$\hat{\alpha}_\psi$	λ_ψ	η_ψ	ϕ_ψ
W/O D.Z.	80.00	55.87	0.010	1.000	0.090	0.100
With D.Z.	80.00	55.87	0.010	1.000	0.180	0.200

Figure 5.41 shows a very rapid rise time for the controller using the dead zone since the switching gain, η_ψ is relatively high but the control action is very small in the steady state as shown in Figure 5.42. It can also be seen that the control only activates for values above 4.0 volts as designed. Only one side of the dead zone is active since the ethernet tether was placing a small disturbance moment on the vehicle in the $-\psi$ direction.

3. Heading Control Experimental Results (Command Generators)

The command generators described in the previous chapter for submergence control can also be extended to include rotational motion. The two forms for heading control are

$$[\psi_{com}, \dot{\psi}_{com}, \ddot{\psi}_{com}] = G(\psi_0, \psi_f, \dot{\psi}_{max}, \ddot{\psi}_{max}, T_0) \quad \text{(Form 1)}$$

or

$$[\psi_{com}, \dot{\psi}_{com}, \ddot{\psi}_{com}] = G(\psi_0, \psi_f, \ddot{\psi}_{max}, T_0, T_f) \quad \text{(Form 2)}$$

where ψ_0 and T_0 is the initial heading and starting time while ψ_f and T_f is the final heading and time at the end of the maneuver.

Form 1 of the command generator was used to control the vehicle from a heading of 0 to $\pi/2$ radians as shown in Figures 5.43 through 5.45. The values for $\dot{\psi}_{max}$, and $\ddot{\psi}_{max}$ were 0.033 rad/sec and 0.006 rad/sec^2 and the nominal control gains and vehicle parameters were used. The vehicle tracks the commanded angular position perfectly but does not follow the desired rate very well during the constant angular velocity phase. This

166

is due to an uncompensated bias in the rate gyro but does not cause any performance degradation.

The response for Form 2 of the command generator is presented in Figures 5.46 through 7.49. This test controlled the vehicle from 0 to π radians in an elapsed time of 60 seconds using a much gentler profile which has no region of constant angular velocity. Again the heading command is tracked very well even with the rate bias error.

Both tests revealed the high thruster chatter that was seen from the submergence results and are due to the same reason. The errors are continually very small throughout the maneuver and the high gain effect of Eqn. (5.27) is again responsible.

4. Conclusions From Heading Control

The results of Sliding Mode heading control for the Phoenix using the lateral thrusters has been exceptional in both step response and command generator performance. The tendency of the lateral thrusters to chatter has not adversely affected the vehicle motions. Using post filtering and a dead band on the thruster input voltage lead to a significant reduction of this undesirable phenomena. The command generator performance was highly accurate, demonstrated by almost perfect tracking of the input profiles.

D. LONGITUDINAL CONTROL

Described in this section is the development of the sliding mode, longitudinal position control algorithms for the NPS Phoenix vehicle. The control has been demonstrated using wall servoing, which involves maneuvering the vehicle (along the body-fixed x-axis) to a prescribed distance from one of the tank sides, using the ST1000 profiling sonar for position feedback. By using the sonar to ping against one of the tank walls, a smoothed range and range rate may be determined by proper filtering. Maneuvers of this type have important applications in the areas of inspection of underwater structures or close-up examination and classification of mines. The ability to servo a vehicle next to targets of interest greatly enhances mission capabilities.

Results from a simulation study using this control technique, including current disturbances and thruster lags can be found in (Marco, 1992), while the analysis presented here details the experimental evaluation of the controller.

The first section outlines the development of the control algorithm followed by a discussion of a Kalman filter design for range rate estimation from the sonar, followed by experimental results for step response of wall standoff distance commands. Finally, some issues pertaining to sonar operation for this application are discussed.

1. Vehicle Model for Longitudinal Control

The vehicle dynamics for longitudinal (x-axis) motion using the stern screws can be simplified to the following continuous time differential equation.

$$M_x \ddot{x}(t) + b_x \dot{x}(t)|\dot{x}(t)| = 2\alpha_x v_x(t)|v_x(t)| + \delta f_x(t) \tag{5.29}$$

where

$$M_x = m + m_{ax}$$

and

$$v_x(t)|v_x(t)| = (v_{ls}(t)|v_{ls}(t)| + v_{rs}(t)|v_{rs}(t)|)/2$$

m is the vehicle mass, and m_{ax} is the added mass of the body in the longitudinal direction. $u(t)$ is the body-fixed rate for the longitudinal direction. b_x is the square-law damping coefficient, and $v_{ls}(t)$, $v_{rs}(t)$ are the motor input voltages for the left/right stern screws respectively. The voltage to force coefficient is given by α_x and $\delta f_x(t)$ is the force error. It has an upper bound, γ, such that $L_I(\delta f_x) \leq \gamma$ where L is the L_I norm of $\delta f_x(t)$. Since motions are assumed to be restricted to the horizontal plane and along the longitudinal axis only, with no current, $u(t) = \dot{x}(t)$ and $\dot{u}(t) = \ddot{x}(t)$.

Since the vehicle motion dynamics are expected to be second order in position, the sliding surface for servo control is specified as

$$\sigma_x(t) = \dot{\tilde{x}}(t) + \lambda_x \tilde{x}(t), \tag{5.30}$$

168

where the tracking errors are defined as

$$\tilde{x}(t) = x_{com}(t) - x(t)$$

$$\dot{\tilde{x}}(t) = \dot{x}_{com}(t) - \dot{x}(t).$$

(5.31)

The sliding mode control law for servo control is given by

$$v_{ls}(t), v_{rs}(t) = \sqrt{|\gamma(t)|}sgn(\gamma(t))$$

(5.32)

where

$$\gamma(t) = \frac{\hat{M}_x}{2\hat{\alpha}_x}\left(\ddot{x}_{com} + \lambda_x\dot{\tilde{x}}(t) + \frac{\hat{b}_x}{\hat{M}_x}\dot{x}(t)|\dot{x}(t)| + \eta_x tanh(\sigma_x(t)/\phi_x) - \frac{\delta f_x(t)}{\hat{M}_x}\right)$$

The test scenario is shown in Figure 5.49 where the range returned by the ST1000 sonar is denoted $R(t)$, and since the direction towards the wall is positive x, motions in this direction will generate range values from the sonar which follow

$$R(t + \Delta t) < R(t)$$

which implies that the rate is

$$\dot{R}(t + \Delta t) < 0,$$

while for the vehicle

$$\dot{x}(t + \Delta t) > 0.$$

To deal with the sign difference between the range and vehicle rates, a change of variable may be introduced such that

$$x(t) = R_0 - R(t)$$
$$\dot{x}(t) = -\dot{R}(t)$$

169

where R_0 is the initial range to the wall prior to the start of the maneuver and is also the position where x is defined to be 0. The commanded x position can then be described in terms of the wall standoff range by

$$x_{com} = R_0 - R_{com}, \tag{5.33}$$

which is used in the position error calculations forming the sliding surface, Eqn. (5.30).

2. Filter Design for Longitudinal Control

The filter structure used for longitudinal control is identical to the one presented in the first part of this chapter for applied to vehicle submergence. In this case, the input signal is the range from the ST1000 profiling sonar. The filter provides a smoothed range and estimate of the range rate which will be used for vehicle control. The filter model is again based on a three state kinematics model for range excited by acceleration noise,

$$\begin{aligned}
\dot{\hat{R}}_1(t) &= \hat{R}_2(t) \\
\dot{\hat{R}}_2(t) &= \hat{R}_3(t) \\
\dot{\hat{R}}_3(t) &= q(t).
\end{aligned} \tag{5.34}$$

The states $\hat{R}_1(t)$, $\hat{R}_2(t)$, and $\hat{R}_3(t)$ are estimates of the position, velocity, and acceleration of the range signal $R(t)$, while the measurement equation, with $R(t)$ being the range, is

$$R(t) = \hat{R}_1(t) + v(t), \tag{5.35}$$

where $q(t)$ is taken to be the system noise and $v(t)$ is the measurement noise. The gains used were also the ones obtained for Filter 2 defined earlier, namely

$$L = \begin{Bmatrix} 0.2544 \\ 0.3727 \\ 0.2731 \end{Bmatrix}$$

which provided good results for the environment of the test tank and nature of the signal.

170

Figure 5.50 shows the raw sonar data together with it's filtered estimate and the vehicle velocity estimate. Range drop outs and false readings from the water column are evident in the raw signal as shown. Anomalies of this magnitude cause serious problems if used in a control law and must be ignored. One method to detect range anomalies is to monitor the residuals, r, given by

$$r = R_k - H\overline{R}_k,$$
(5.36)

which, for this filter, is the difference between the measured range and the expected range, and is known as the "innovations process" (Friedland, 1986). The covariance of the residuals, σ_r^2, is the model based propagated state error covariance, given by

$$\sigma_{r_k}^2 = \Phi P_{k-1} \Phi^T + \Gamma q_{k-1} \Gamma^T.$$
(5.37)

If a residual exceeds $3\sigma_r$, the corresponding range return is declared an anomaly and is rejected. Since the measurement is invalid, the Kalman gain, L_k, is set to zero and the new state estimate is propagated without correction. The condition for rejection can also be cast as a ratio given by

$$\frac{r_k^2}{\sigma_{r_k}^2} > 9,$$
(5.38)

which is commonly referred to as the "normalized innovation".

Application of this technique provides satisfactory results, except for instances when multiple consecutive anomalies occur. In this case, filter estimates propagate without correction, and after some time, (depending on the estimated state), diverge to the point where filter lock is lost. Checks for filter divergence are incorporated into the vehicle operational software. The decision is keyed on the estimated range rate, where if $\dot{R}_k > \dot{R}_{max}$ divergence is suspected, and use of the filter for navigation is terminated. An $\dot{R}_{max}$ of $3.0 \, ft/sec$ has been used, which is a maximum feasible velocity for the vehicle. Regaining lock is unfortunately not an easy task. Several options for filter re-initialization are available but depend on particular circumstances, and this an area for further work.

171

3. Longitudinal Control Experimental Results (Step Response)

In this section, results obtained from in water testing of wall servoing control are presented. Each test began by submerging the vehicle to 2.5 ft using the depth control method outlined earlier to minimize surface effects during translation. The vehicle was also under heading control to maintain the longitudinal axis perpendicular to an end wall. To perform this maneuver, two phases are needed, one to identify the target, and another to servo the vehicle. The Prolog code used is :

```
                          .
                          .
execute_phase(3)     :- exec_find_sonar_target(X), repeat, phase_completed(3).

phase_completed(3) :- exec_ask_sonar_target_found(X), X==1,
                exec_start_sonar_filter(X), asserta(complete(3)).

phase_completed(3) :- ask_sonar_ping_out(X), X==1, exec_surface(X),
                repeat, ask_surface_reached(X), X==1,
                asserta(abort(3)).


execute_phase(4)     :- exec_next_setpt_data(X), exec_servo_X(X),
                exec_start_X_error_filter(X), exec_submerge(X),
                exec_rotate(X), exec_start_timer(X), repeat,
                phase_completed(4).

phase_completed(4) :- exec_sleep(1,X),ask_X_reached(X), X==1,
                ask_depth_reached(X), X==1, ask_heading_reached(X), X==1,
                asserta(complete(4)).

phase_completed(4) :- ask_time_out(X), X==1, exec_surface(X),
                printsc('PHASE 4 ABORTED DUE TO TIME OUT!'),
                repeat, ask_surface_reached(X), X==1,
                asserta(abort(4)).
                          .
                          .
```

Vehicle submergence is performed in phase 2, followed by activating the sonar to identify the distance to the wall. Note that there is no submerge or rotate command for phase 3, since the depth and heading commands for phase 2 are still in effect. Execution of the predicate "exec_find_sonar_target" starts the search by pinging against the end wall, and if 3 consecutive, consistent ranges are obtained, the predicate

172

"exec_ask_sonar_target_found" evaluates TRUE. At this time, "exec_start_sonar_filter" is executed and the Kalman filter algorithm, part the Execution Level, is initialized to the average of the three consecutive ranges, which completes phase 3. With the range to the wall determined, servoing control is activated in phase 4 where "ask_X_reached" succeeds if the vehicle reaches the set point within the allowed timeout. In phase 3, if the wall is not identified after 5 attempts, based on 3 consecutive pings, the predicate "ask_sonar_ping_out" is evaluated TRUE and the mission aborts.

A nominal set of control gains and vehicle parameters were obtained through computer simulation and parameter identification of the wall servo motion, (Torsiello, 1994), which are given in Table 5.10 below.

Table 5.10 Nominal Parameters and Gains for Servoing Control

Parameter/ Gain	$\hat{M}_x$	$\hat{b}_x$	$\hat{\alpha}_x$	λ_x	η_x	ϕ_x
Unit	$\dfrac{lb - sec^2}{ft}$	$\dfrac{lb - sec^2}{ft^2}$	$\dfrac{lb}{V^2}$	$\dfrac{rad}{sec}$	$\dfrac{ft}{sec^2}$	$\dfrac{ft}{sec}$
Value	14.86	1.33	0.025	0.2	0.1	0.2

Two test runs using the above values are shown in Figures 5.51 through 5.53. A commanded standoff distance, $x_{com}(t)$, of 3.0 feet was chosen, with an initial distance from the wall approximately 9.3 and 11.0 feet for Tests 1 and 2 respectively, with $\dot{x}_{com} = \ddot{x}_{com} = 0$. Both results show an extremely well behaved position response exhibiting no overshoot, regardless of the initial starting point. Figure 5.52 reveals that the estimated velocity is slightly noisy but does not adversely affect the results. Shown in Figure 5.53 is the voltage commands to the stern screws, which exhibit some chattering due to the controller switching term, but is not at such a high frequency to cause significant actuator wear.

4. Robustness Analysis of Longitudinal Motion Maneuvers

A robustness analysis using the control design and experimental data from above will now be presented. Eqn. (2.33), cast in terms of the longitudinal control parameters can be written as

$$\eta_x \geq \frac{\gamma_x F_x}{\hat{M}_x} + \frac{\left|\hat{f}_x\right|_\infty}{\hat{M}_x} \cdot \left|\left(1 - \gamma_x\right)\right| + \left|\left(\mu_x \gamma_x - 1\right)\right| \cdot \left|\left(\ddot{x}_{com} + \lambda \dot{\tilde{x}}\right)\right|_\infty, \qquad (5.39)$$

where

$$\frac{1}{\mu_x} \leq \frac{\hat{M}_x}{M_x} \leq \mu_x; \ \mu_x > 1, \quad \frac{1}{\gamma_x} \leq \frac{\hat{\alpha}_x}{\alpha_x} \leq \gamma_x; \ \gamma_x > 1,$$

and $\left|\hat{f}_x - f_x\right|_\infty = F_x$, where $\hat{f}_x = -\hat{b}_x \dot{x} |\dot{x}|$.

If the experimental data is substituted into (5.41) with appropriate values for the parameter uncertainties, a time history of the minimum value of η_x required for stability can be produced. Figure 5.54 shows the effect different levels of uncertainty listed in Table 5.11 have on the required magnitude of η_x, applied to the data from Test 1.

Table 5.11   Uncertainty Levels for Longitudinal Control
Robustness Analysis

Uncertainty Level	F	γ	μ
1	0.0	1.2	1.2
2	0.0	1.5	1.5
3	0.0	2.0	2.0

Recalling that $\dot{\tilde{x}} = -\dot{x}$ for step inputs, the shape of the curves are proportional to the vehicle velocity, while the magnitude is scaled by the uncertainty level. During the steady-state phase of the maneuver, the requirement on η_x shrinks since the velocity is very small. Since the controller for the experiment used a value of 0.1 for η_x, and provided a stable response, it can be concluded that the parameter estimates used in Table 5.10, are close to the actual values.

5. Sliding Mode Verses PD Control and Sonar Input Power Considerations

The results presented so far have used a Sliding Mode controller which has provided extremely precise positioning response for the non-linear system involved.

However, previous work used a simple Proportional Derivative (PD) linear controller for the same task. While PD control is simple to implement and executes rapidly in real time, as compared to the model based sliding mode control, the performance is inferior as shown in Figure 5.55. A large overshoot results using the PD controller for the set point $x_{com} = 3.0$ feet, while the SMC shows none, clearly demonstrating the superiority of the non-linear controller for this application.

During the course of the wall servoing experimental work, it was discovered that the sonar power level must be chosen carefully when used in the test tank. The nominal power level was determined by placing the sonar head in the middle of the tank and adjusting it until the most favorable results were obtained. A level of 12% of full power was found to be appropriate for the tank environment, and turned out to be suitable for ranges as close as 4.0 feet from the wall, but not closer. By placing the head closer to the wall, over ensonification occurs and the range signals become extremely erratic as shown in Figure 5.56. With the head some distance from the wall, the signal is very clean and stable, but after settling to approximately 3.0 feet away, the signal begins to breakup, causing the filtered range to diverge. By lowering the power setting to 5% corrected the problem for "close to wall" servoing, and was sufficient for target detection as far away as 12.0 feet. One method to automatically adapt the power setting is to use a formulation such as

$$Power = f(R), \tag{5.40}$$

which automatically reduces the sonar power as the head nears an object. Although this approach is encouraging, it has not been incorporated into the sonar control software, but is an area for future work, where learning algorithms can be applied.

6. Conclusions From Longitudinal Control

The results of sliding mode wall servoing control have shown exceptional accuracy in longitudinal positioning of the vehicle. Using a Kalman filter modified to reject range outliers, provided very accurate and stable signal conditioning from the ST1000 sonar data. A comparison between sliding mode and proportional derivative control was presented which demonstrated the superiority of the non-linear approach. Future work in this area could include the use of two sonars to enable both longitudinal and lateral positioning of the

175

vehicle for more general control scenarios. Also, extensions to behaviors used in wave surge and current conditions using this approach could be done.

E. COORDINATED SUBMERGENCE/HEADING CONTROL USING COMMAND GENERATORS

1. Command Tracking Performance

In some robot motions, tracking to commands for multiple behaviors operating simultaneously becomes important. For instance, while maneuvering around targets, position and orientation of the vehicle may need to be coordinated. Coordination of control behaviors requires the use of time synchronized commands through command generators. To prove that these behaviors are possible in an underwater environment, experiments were conducted to simultaneously submerge and rotate the vehicle to a predetermined depth of 3.0 feet and a heading of 180° respectively. It was specified that the final depth and heading both be reached at 60 seconds from the beginning of the maneuver, and was accomplished for both control modes using command generators.

A section of the Prolog code used to control the mission appears below, and executes until the pre-defined depth and heading has been attained, which is determined by the error criteria presented in Chapter III.

```
execute_phase(2) :-
        exec_submerge(X), X==1,
        exec_rotate(X), X==1,
        exec_start_timer(X),
        repeat, phase_completed(2).

phase_completed(2) :- ask_depth_reached(X), X==1,
        ask_heading_reached(X), X==1,
        asserta(complete(2)).

phase_completed(2) :- ask_time_out(X), X==1,
        exec_surface(X), repeat,
        ask_surface_reached(X), X==1,
        asserta(abort(2)).

phase_completed(2) :- ask_sys_problem(X), X==1,
        exec_surface(X), repeat,
```

176

```
ask_surf_reached(X), X==1,
asserta(abort(2)).
```

.

.

It should be noted that the Prolog code is identical to that used for simultaneous depth and heading control for step command inputs. The difference lies in the Tactical Level mission file, which specified the use of command generators for both depth and heading control. Refer to Chapter III for a full description of the Tactical Level mission input file.

The following tables give the values used in the vehicle control laws developed earlier for submergence and heading control as applied to the coordinated maneuver experiments.

Table 5.12. Parameters for Submergence Control

Parameter	Value	Unit
$\hat{m}$	13.5	lb
$\hat{m}_a$	13.5	lb
$\hat{b}_z$	28.8	$lb / ft - sec^2$
$\hat{\alpha}_z$	0.004	lb / V^2
λ_1	0.400	rad / sec
λ_2	0.040	rad / sec^2
η	0.1	ft / sec^2
ϕ	0.2	ft / sec

Table 5.13. Parameters for Heading Control

Parameter	Value	Unit
$\hat{I}_{zz}$	40.0	$lb - ft - sec^2$
$\hat{I}_{zza}$	40.0	$lb - ft - sec^2$
$\hat{b}_\psi$	55.87	$lb - ft - sec^2$
$\hat{\alpha}_\psi$	0.006	$lb - ft / V^2$
λ	0.200	rad / sec
η	0.200	rad / sec^2
ϕ	0.200	rad / sec

Figures 5.57 and 5.58 show the normalized time responses for depth/depth rate and heading/heading rate respectively. Although the depth command is accurately tracked during the transient phase, an overshoot occurs near the final set point, caused buy the vehicle becoming "heavy" from hull compression. At this point, integral action from the controller quickly compensates for the error. Since no disturbance force was present in rotation, the heading response shows a very precise tracking performance with virtually no error.

Referring to the depth rate response, a very noisy signal is evident, caused by discretization noise from the depth cell A/D converter. Although the signal is far from

clean, the overall tracking performance is not adversely affected. The heading rate measured from the onboard gyroscope shows a definite tracking error, and is attributed to a non-zero bias in the unit. Figure 5.59 shows that the depth and heading are simultaneously controlled to the command generator specification except for the small depth overshoot at the end of the maneuver.

2. Conclusions From Coordinated Control

It has been shown that is a relatively easy task to program coordinated vehicle maneuvers since the generality of the Strategic Level rules allow the method of control to be determined by the Tactical Level. Accurate, simultaneous tracking of the command generator trajectories proved that the servo level controllers are currently very well tuned for the vehicle maneuvers shown.

F. CONCLUSIONS

The results presented have shown that the Phoenix can be precisely controlled in the test tank environment using thrusters. Development of sliding mode controllers for submergence, heading, and longitudinal control have proven to be robust and have shown to provide exceptional vehicle performance. The flexibility of the Strategic Level rules allowed many different control scenarios to be quickly tested and evaluated without major code modifications. Using command generators provided extremely precise time based maneuvering and proved very effective when activated concurrently for multiple control modes.

G. CHAPTER V FIGURES
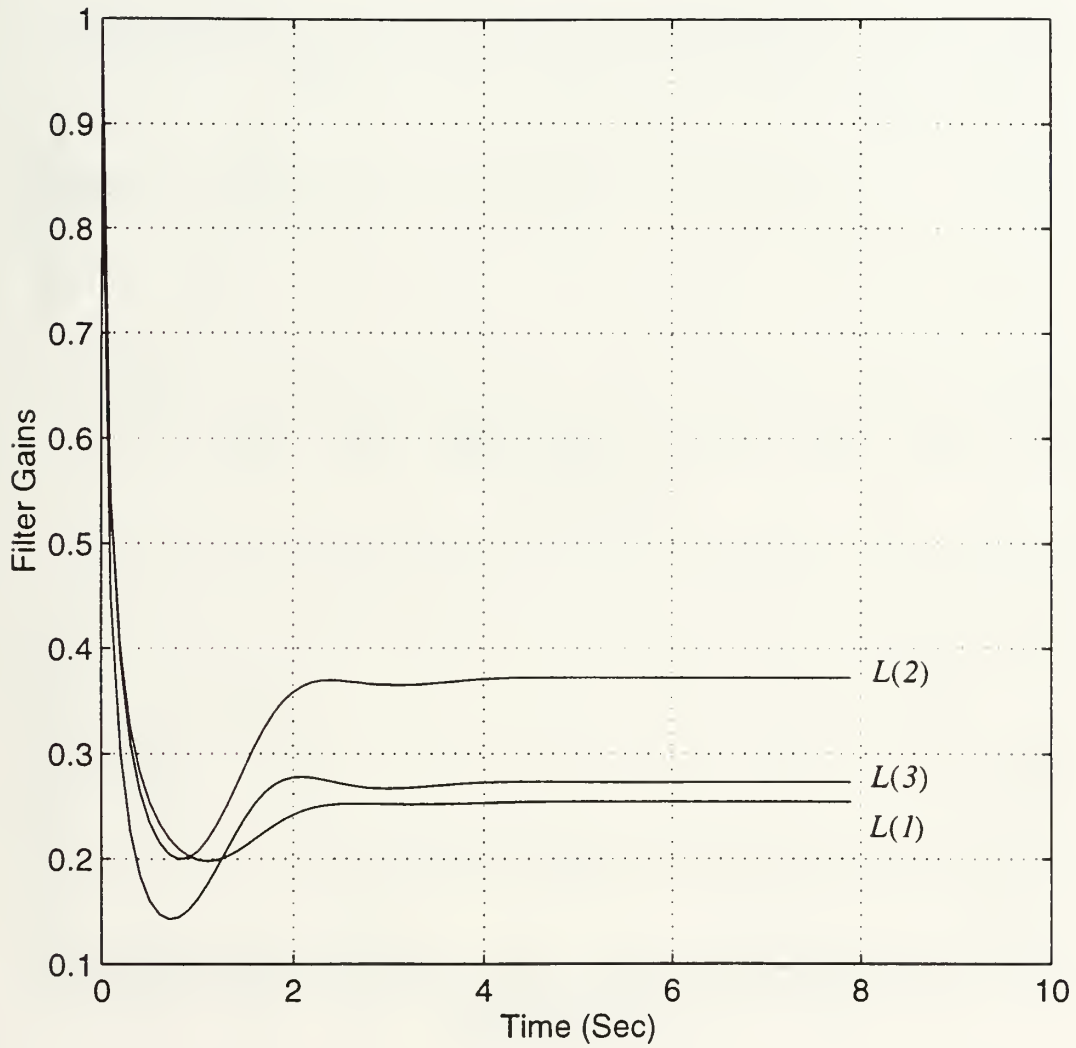
Figure 5.1 Steady State Kalman Filter Gains vs. Time for Filter 2,
where $L_{ss} = [L(1) \ L(2) \ L(3)]^T$.
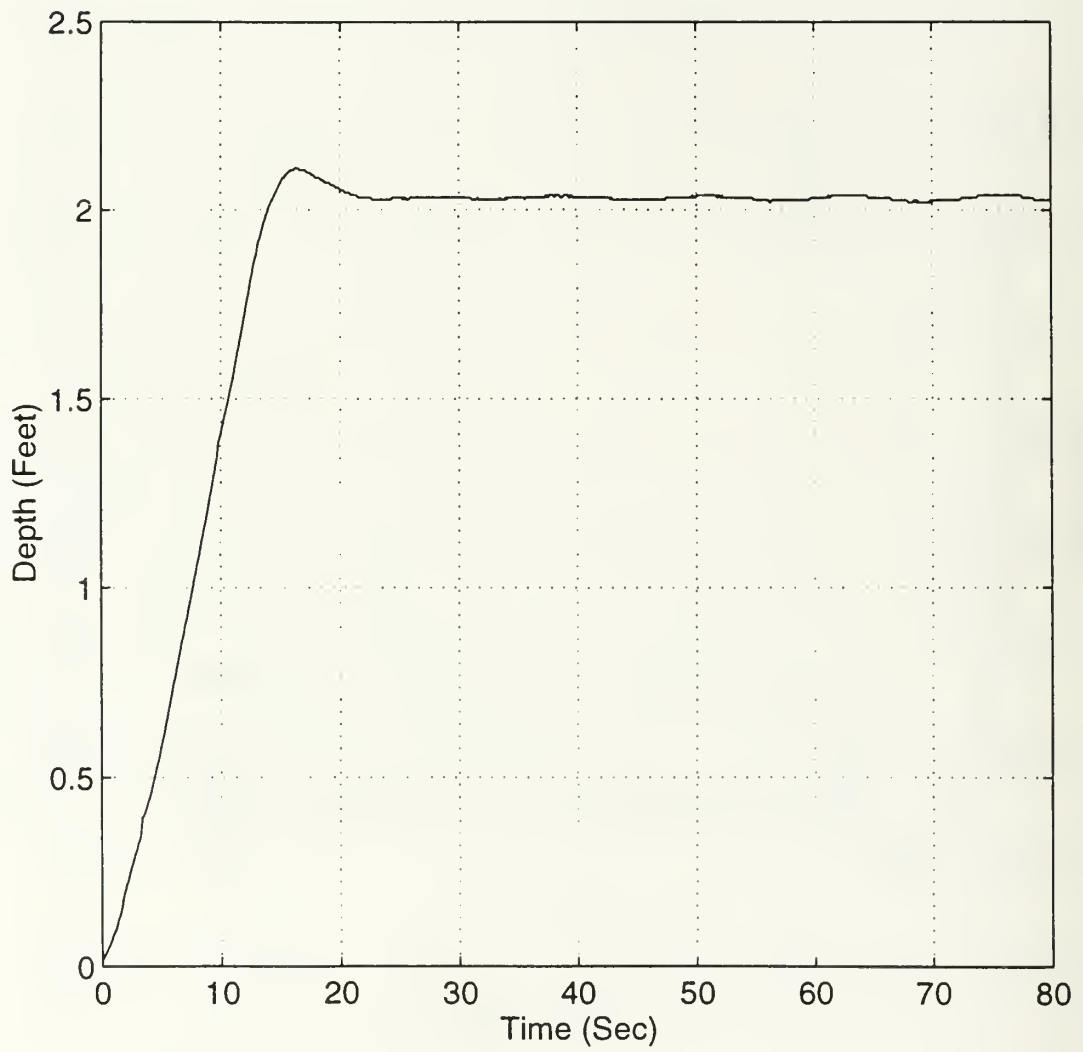
Figure 5.2 Unfiltered Depth Signal vs. Time.
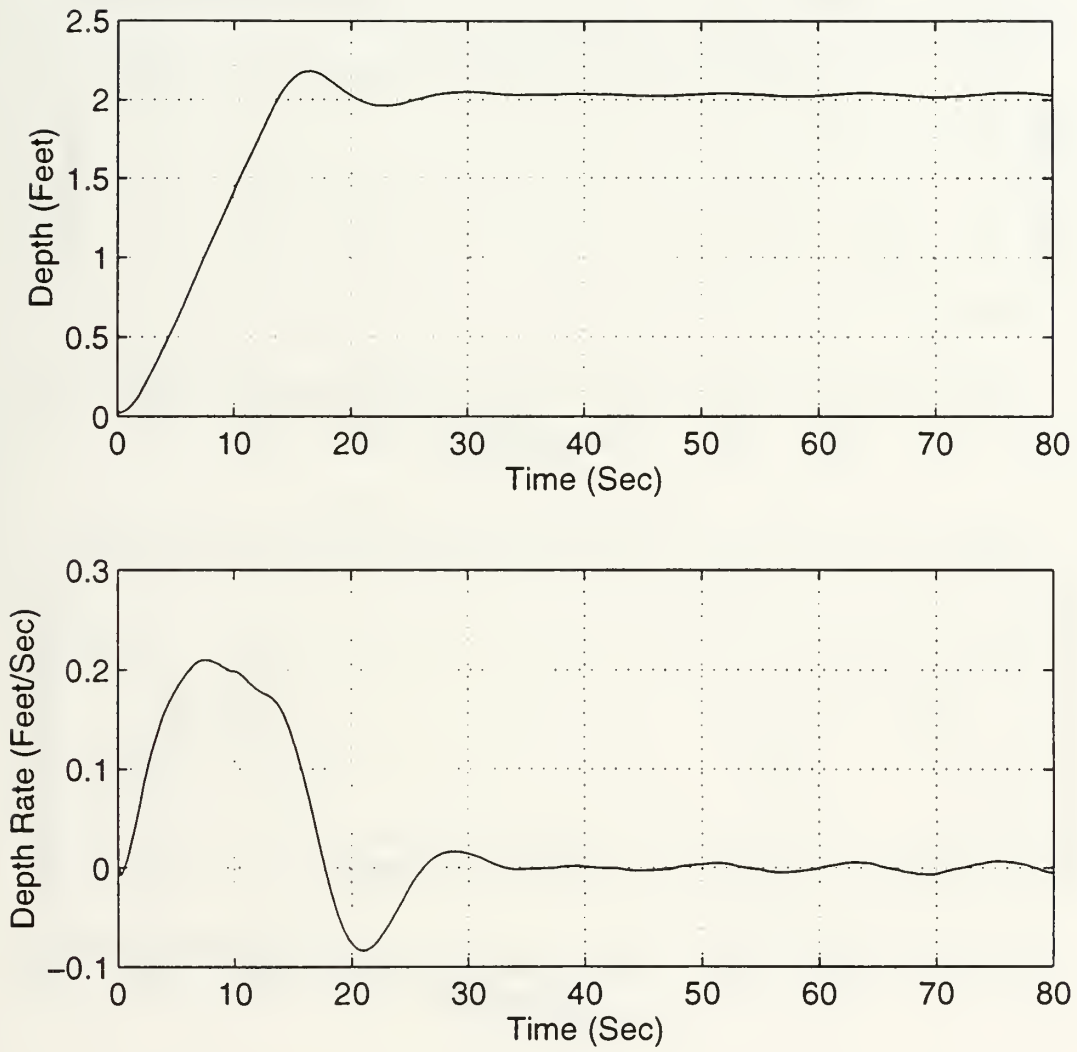
Figure 5.3 Depth and Depth Rate vs. Time Response Using Filter 1.

183
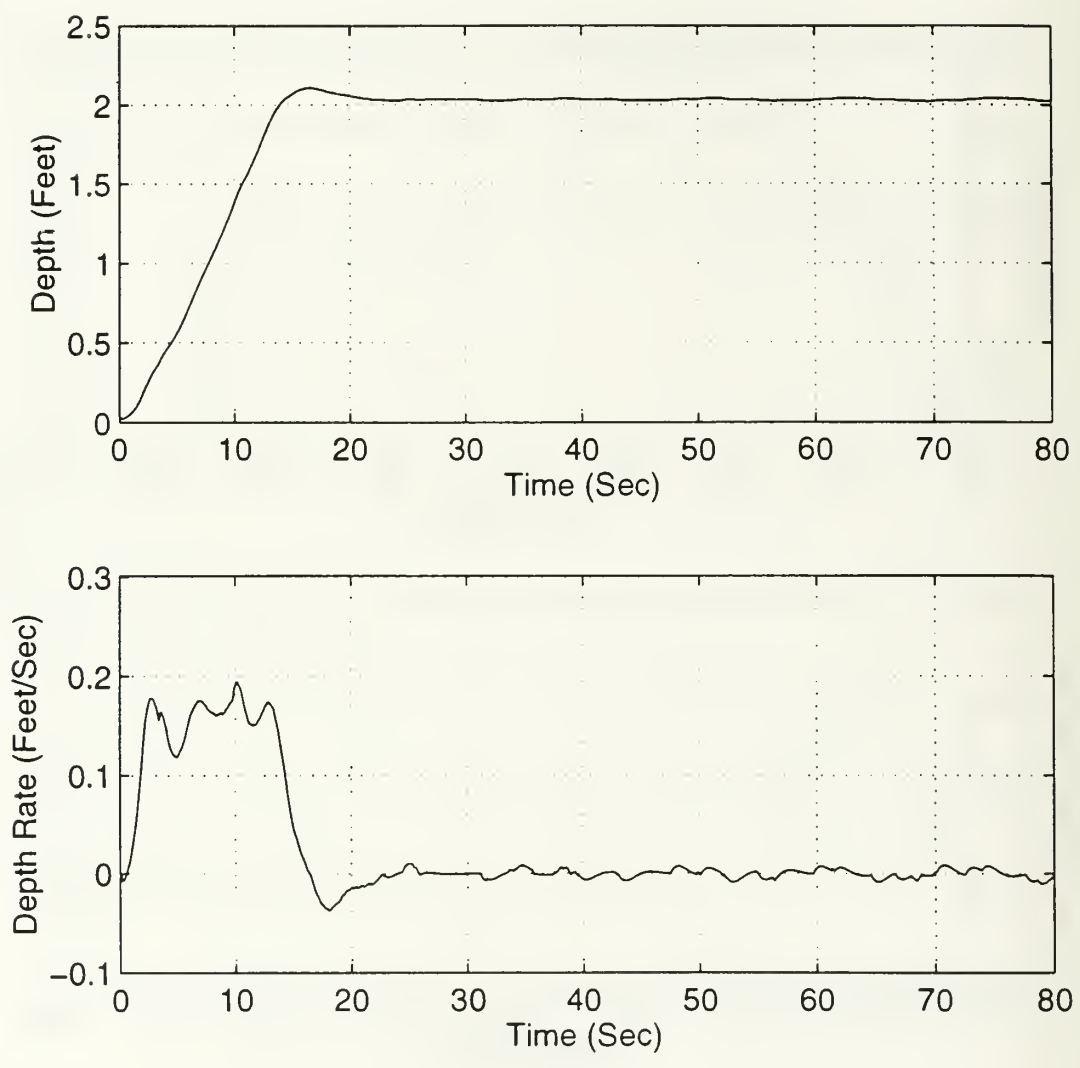
Figure 5.4 Depth and Depth Rate vs. Time Response Using Filter 2.
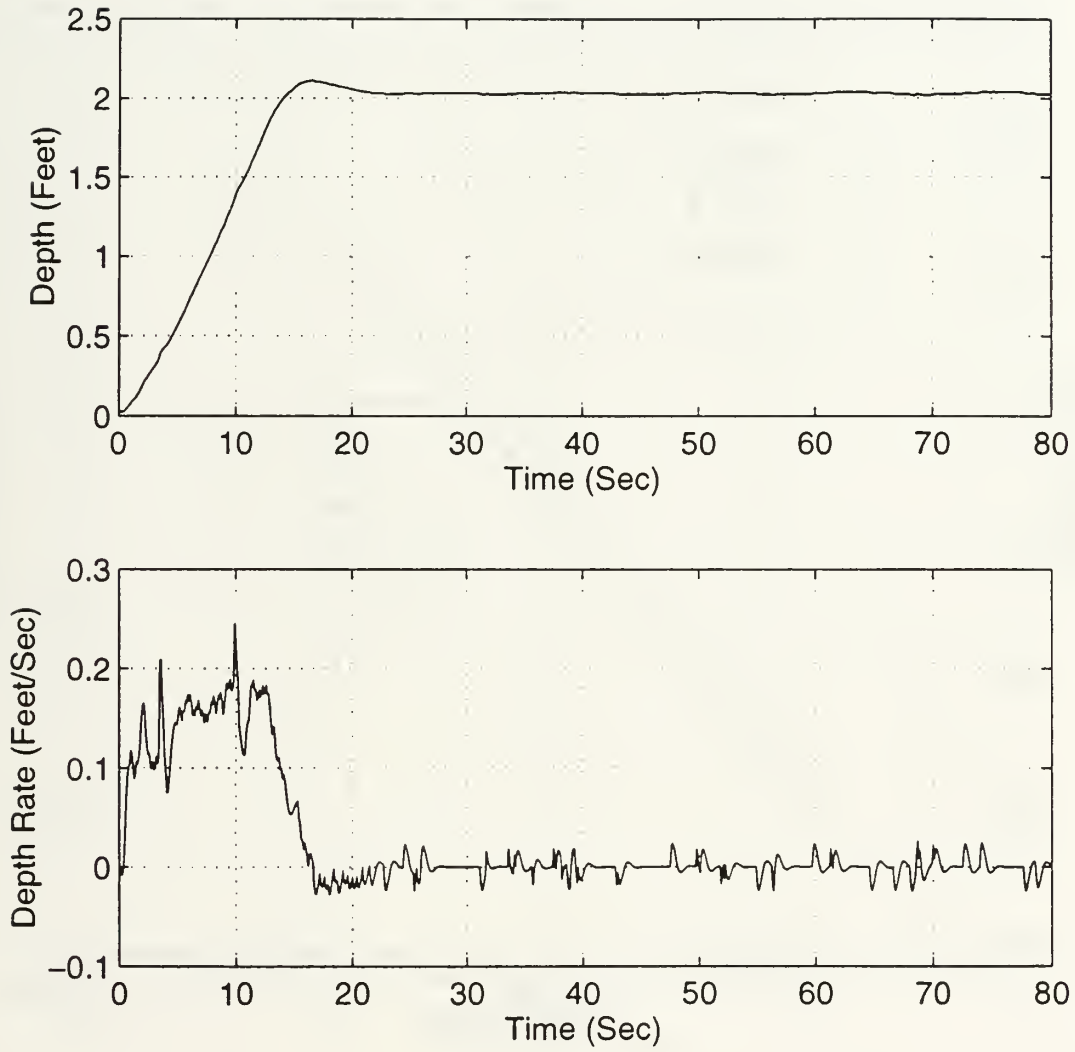
Figure 5.5 Depth and Depth Rate vs. Time Response Using Filter 3.
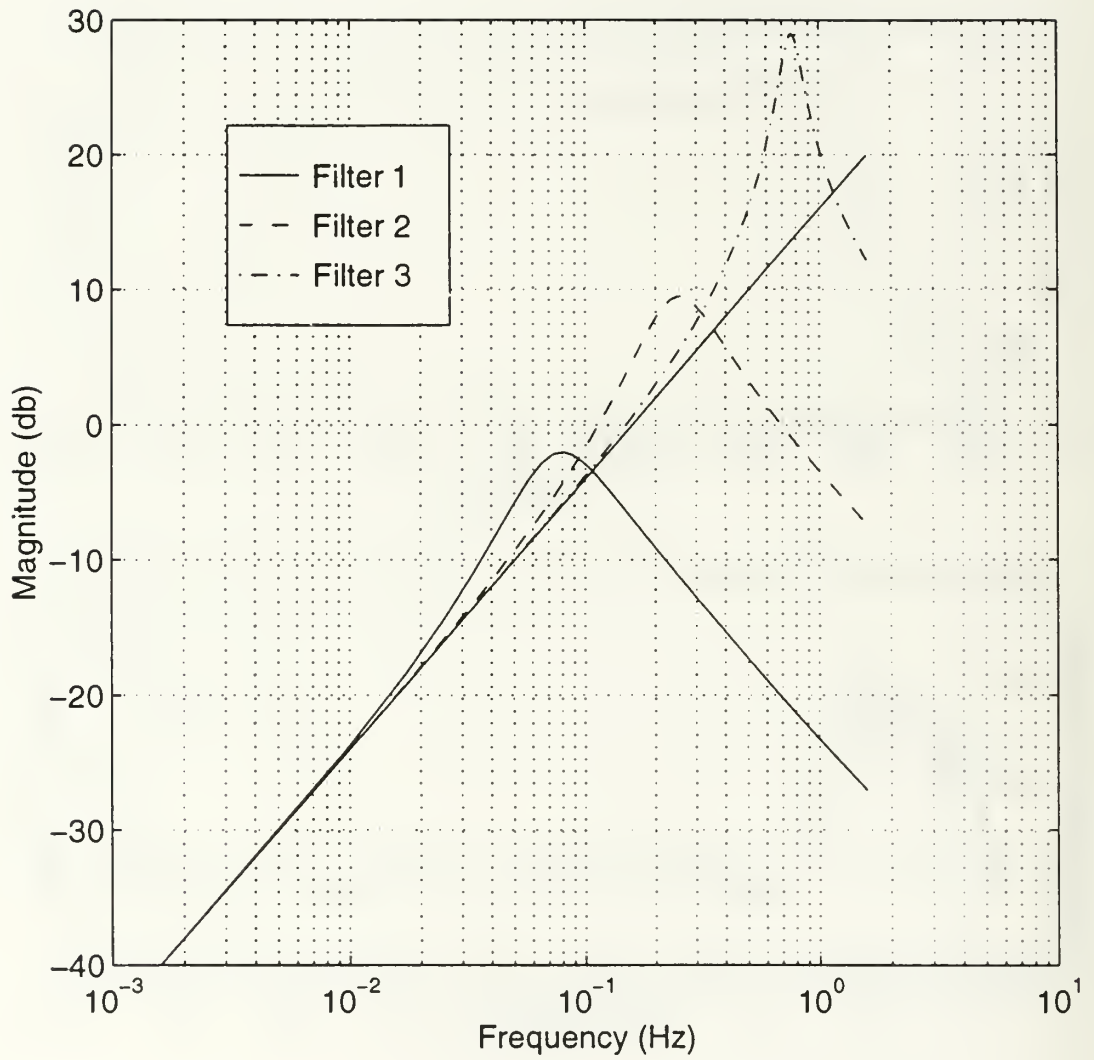
Figure 5.6 Magnitude of Velocity Estimate vs. Frequency for Filters 1, 2, and 3.

186
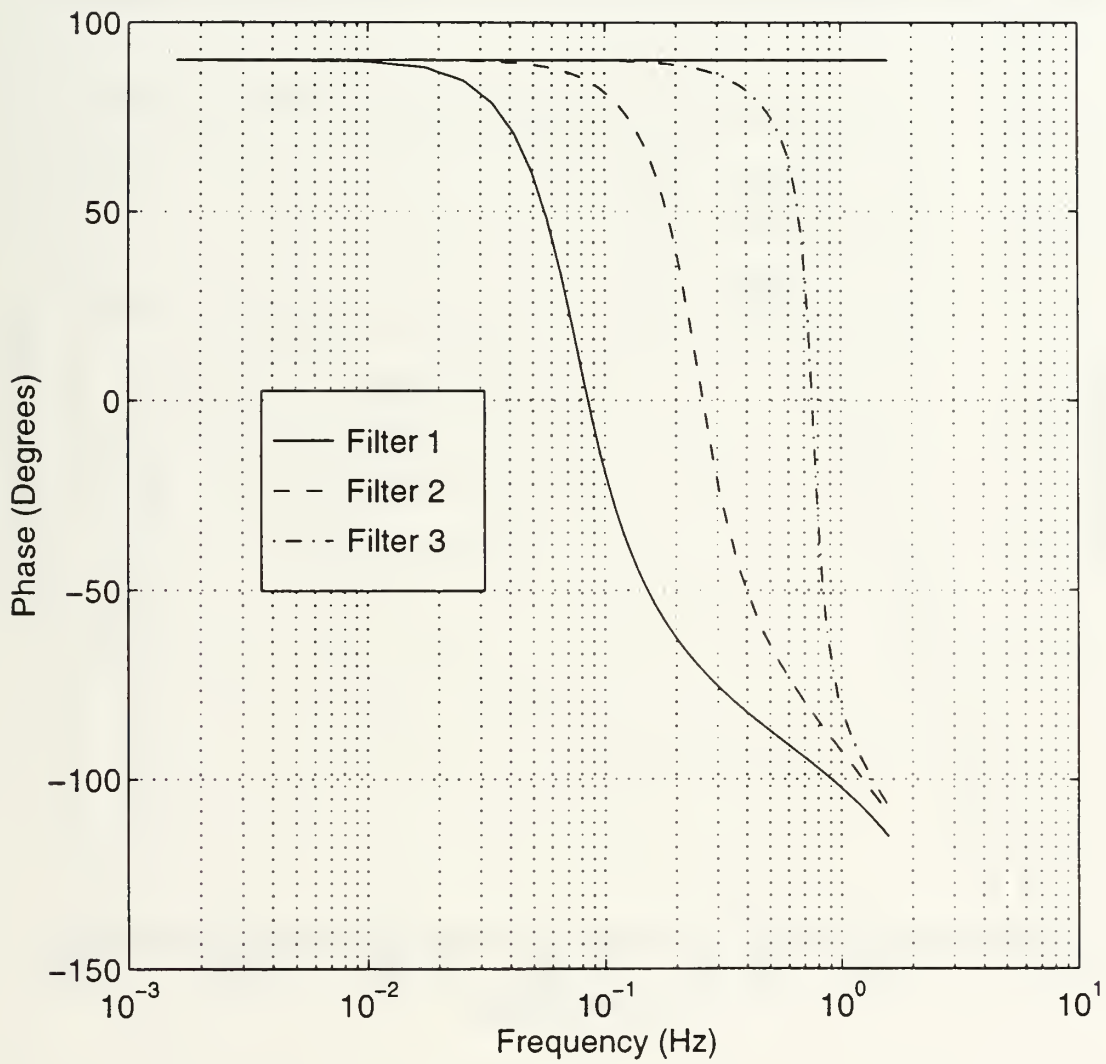
Figure 5.7 Phase Angle of Velocity Estimate vs. Frequency for Filters 1, 2, and 3.
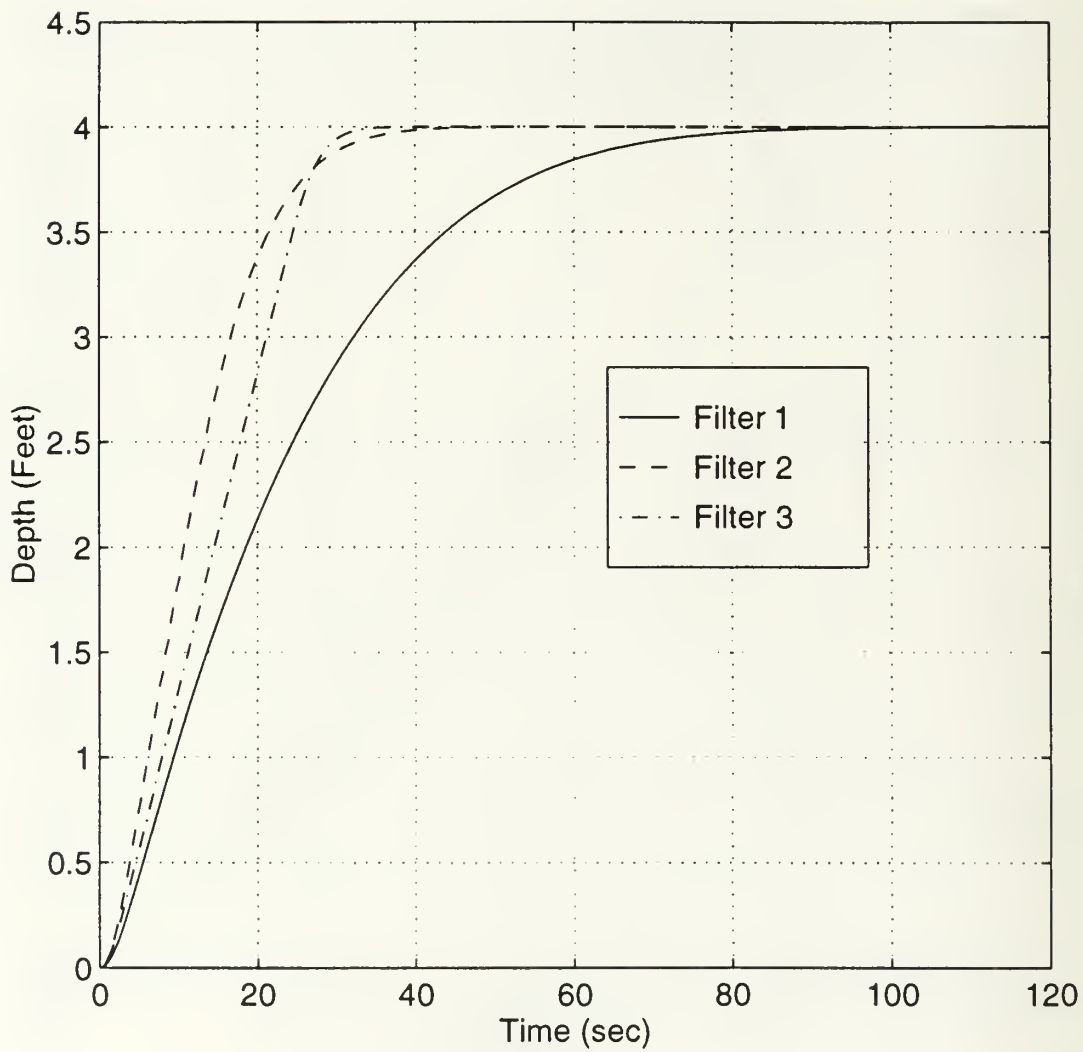
Figure 5.8 Simulated Depth Response vs. Time Using Filters 1, 2, and 3.
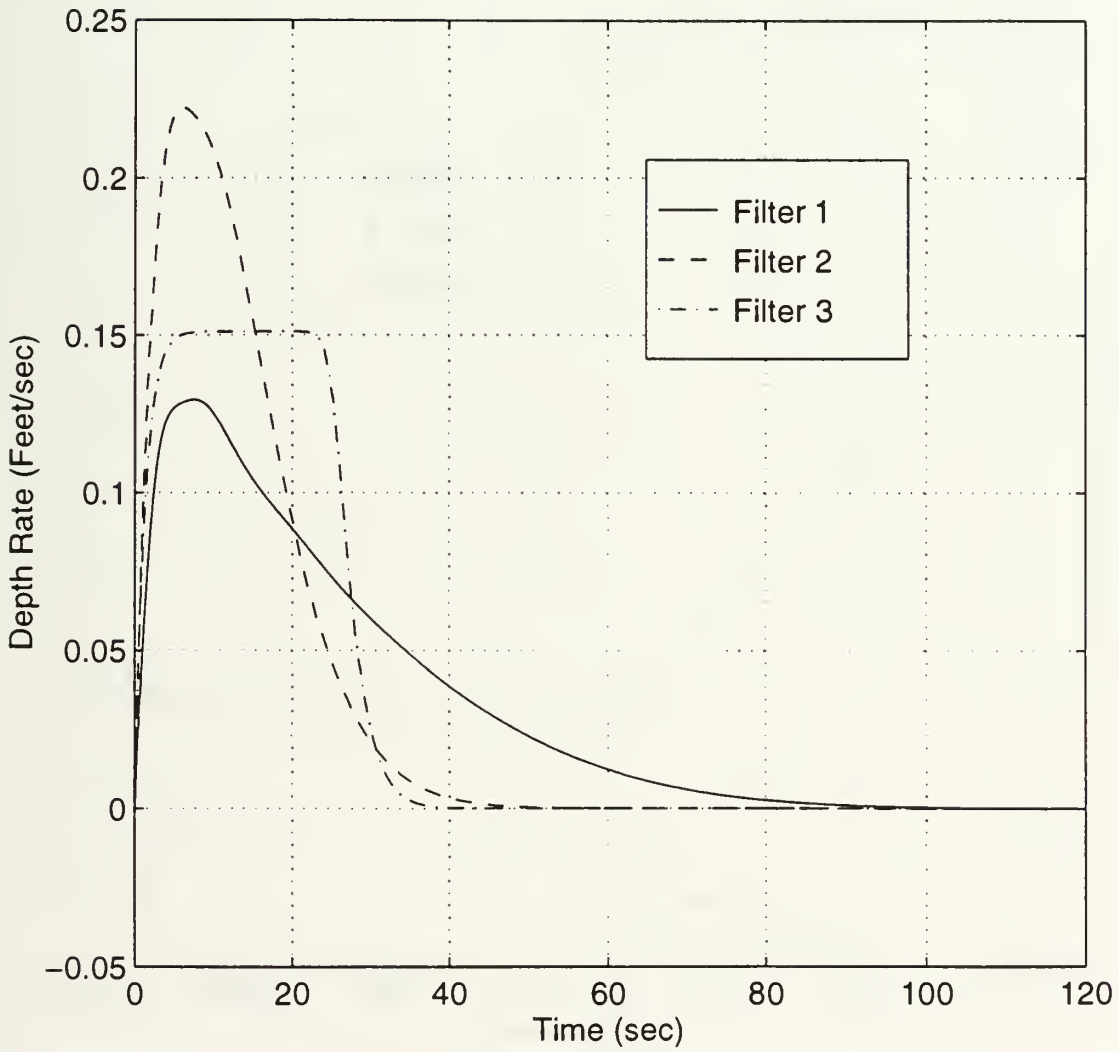
Figure 5.9 Simulated Depth Rate Response vs. Time Using Filters 1, 2, and 3.
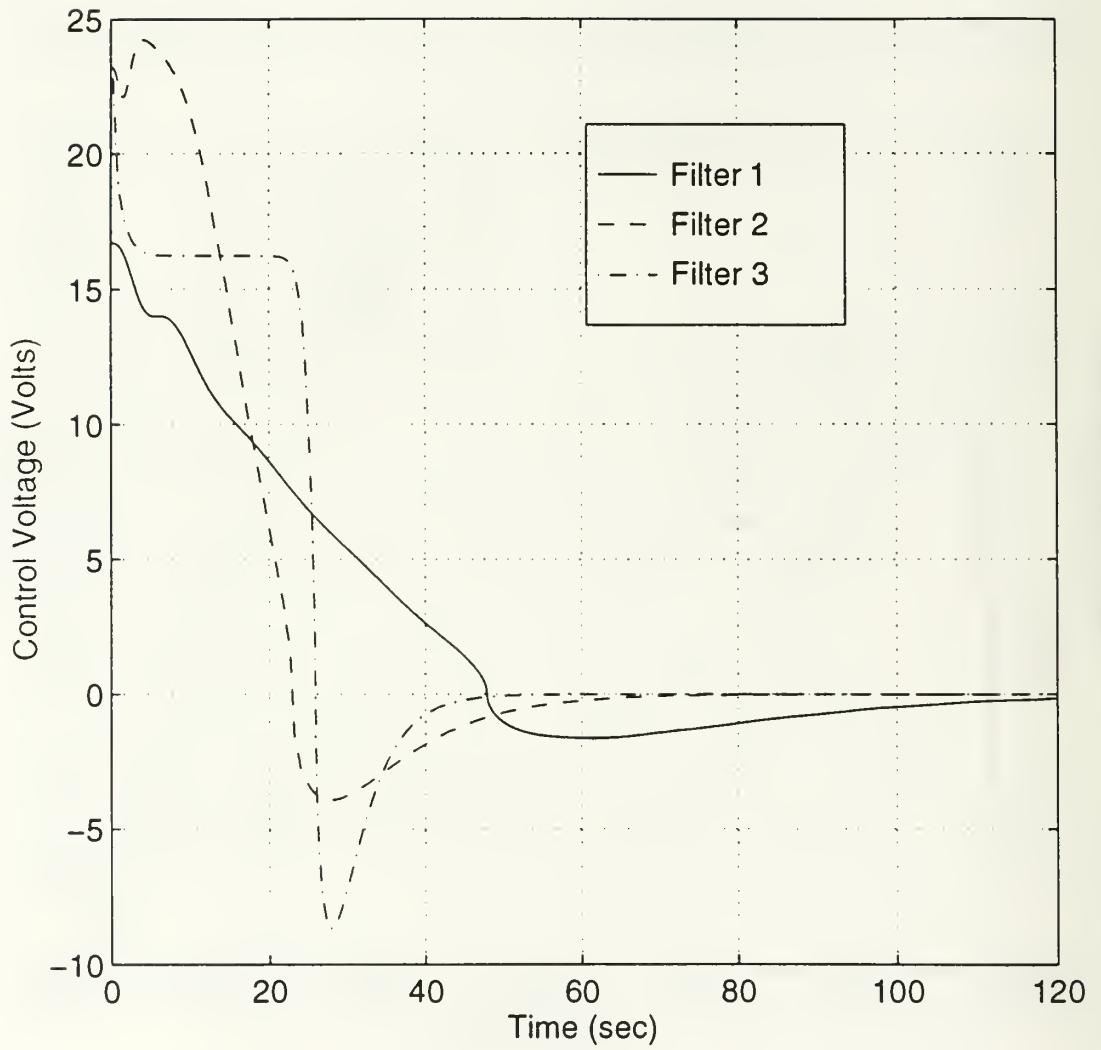
Figure 5.10 Simulated Depth Acceleration Response vs. Time Using Filters 1, 2, and 3.

190
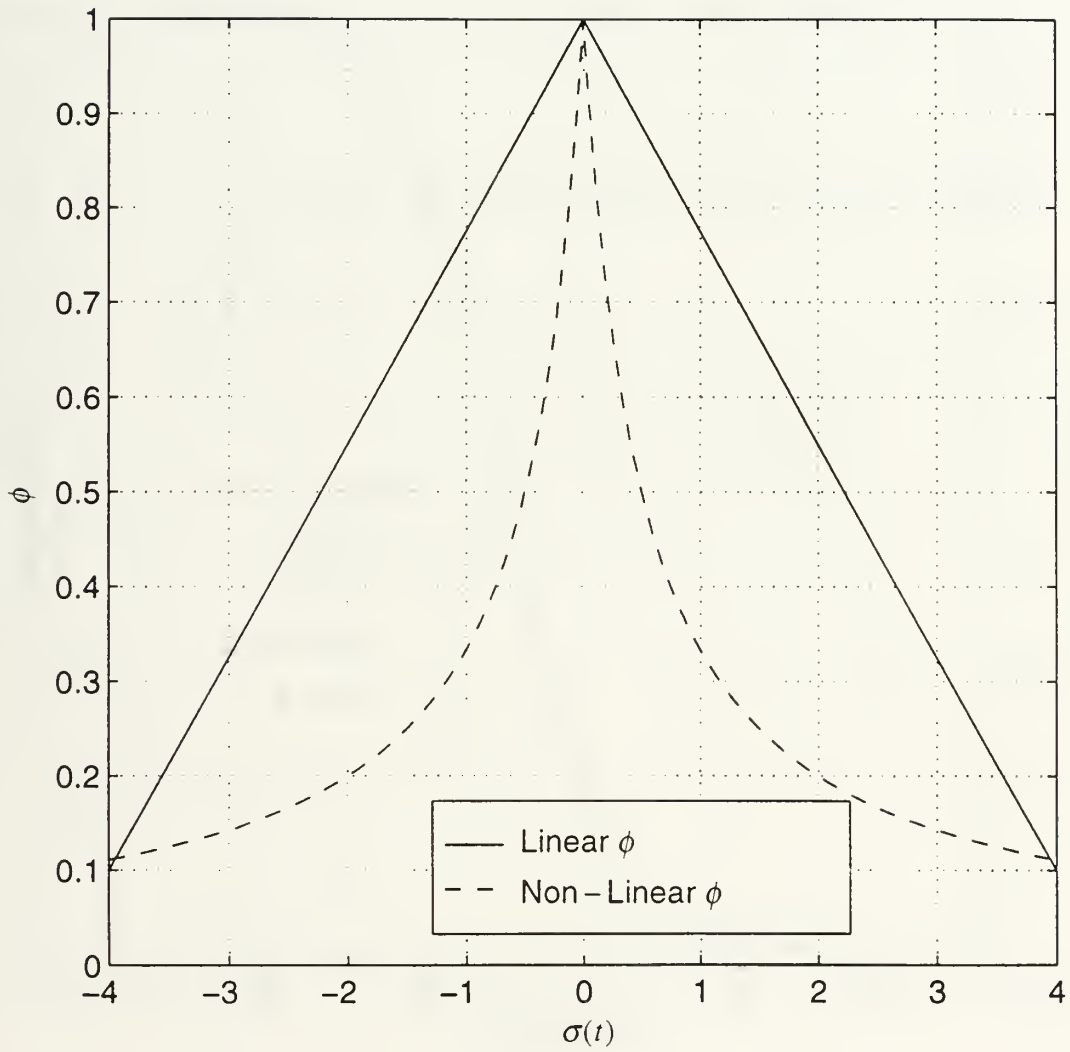
Figure 5.11 Curves for Linear and Non-Linear Formulas of ϕ.

191
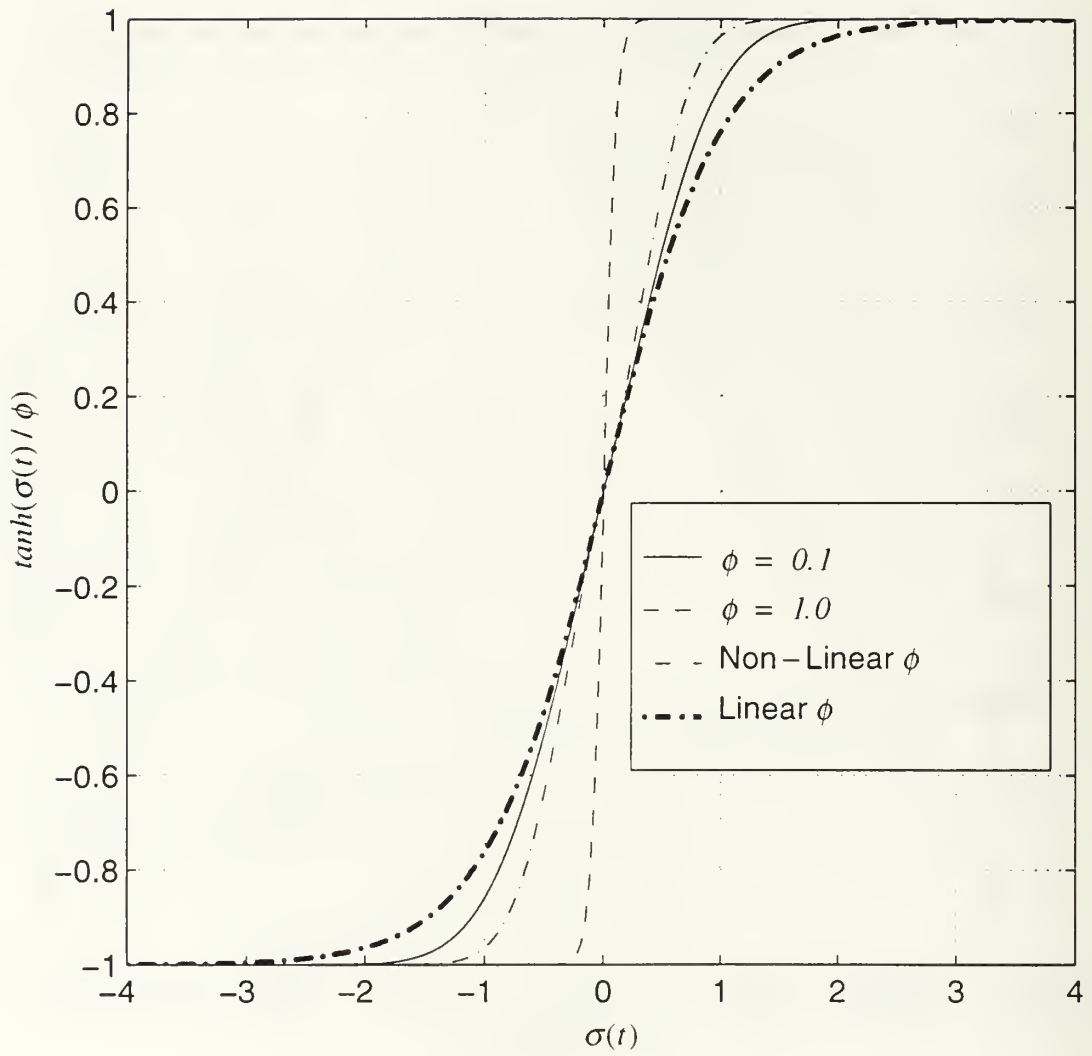
Figure 5.12 Switching Curve for Various Formulas of ϕ.
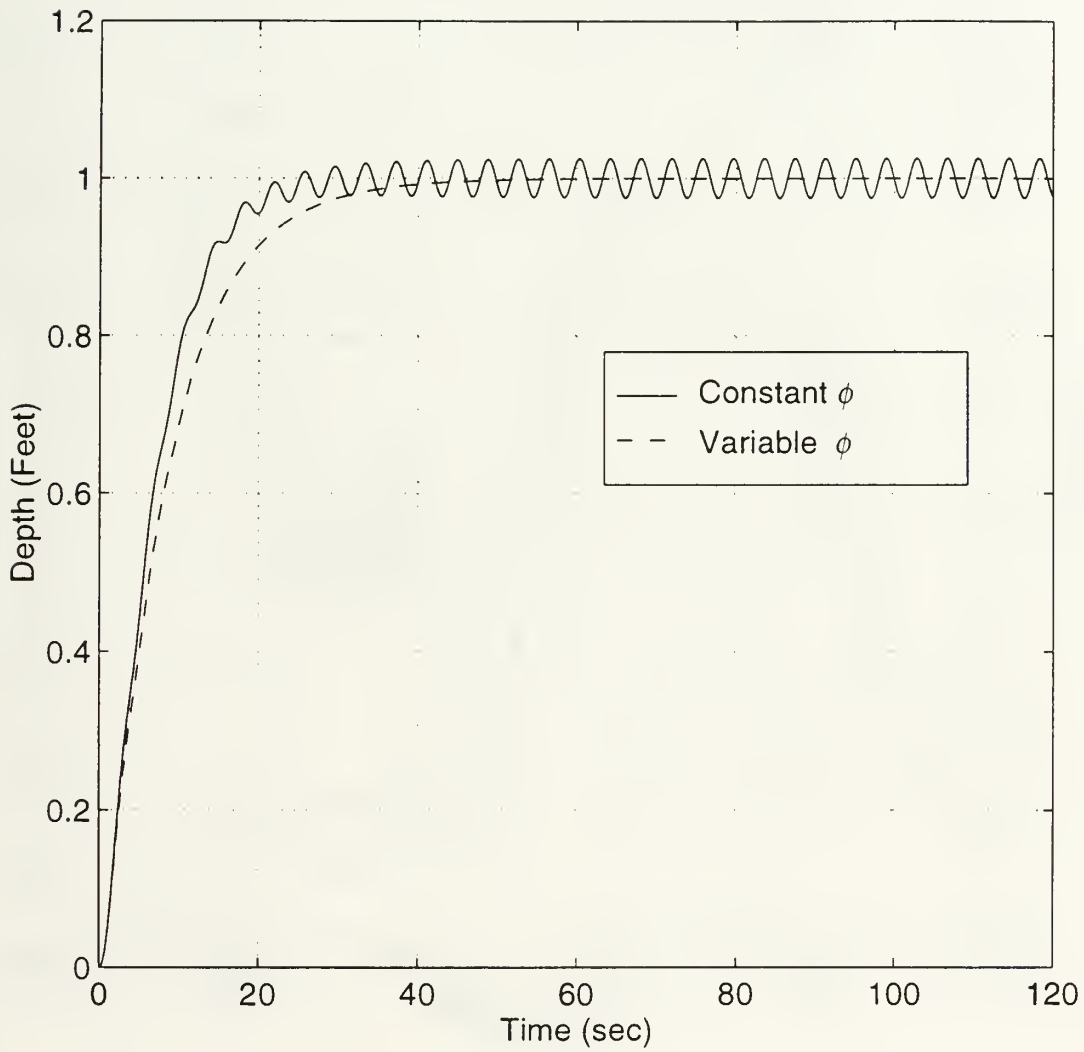
Figure 5.13 Simulated Depth vs. Time Response for a Constant and Variable.
Boundary Layer.
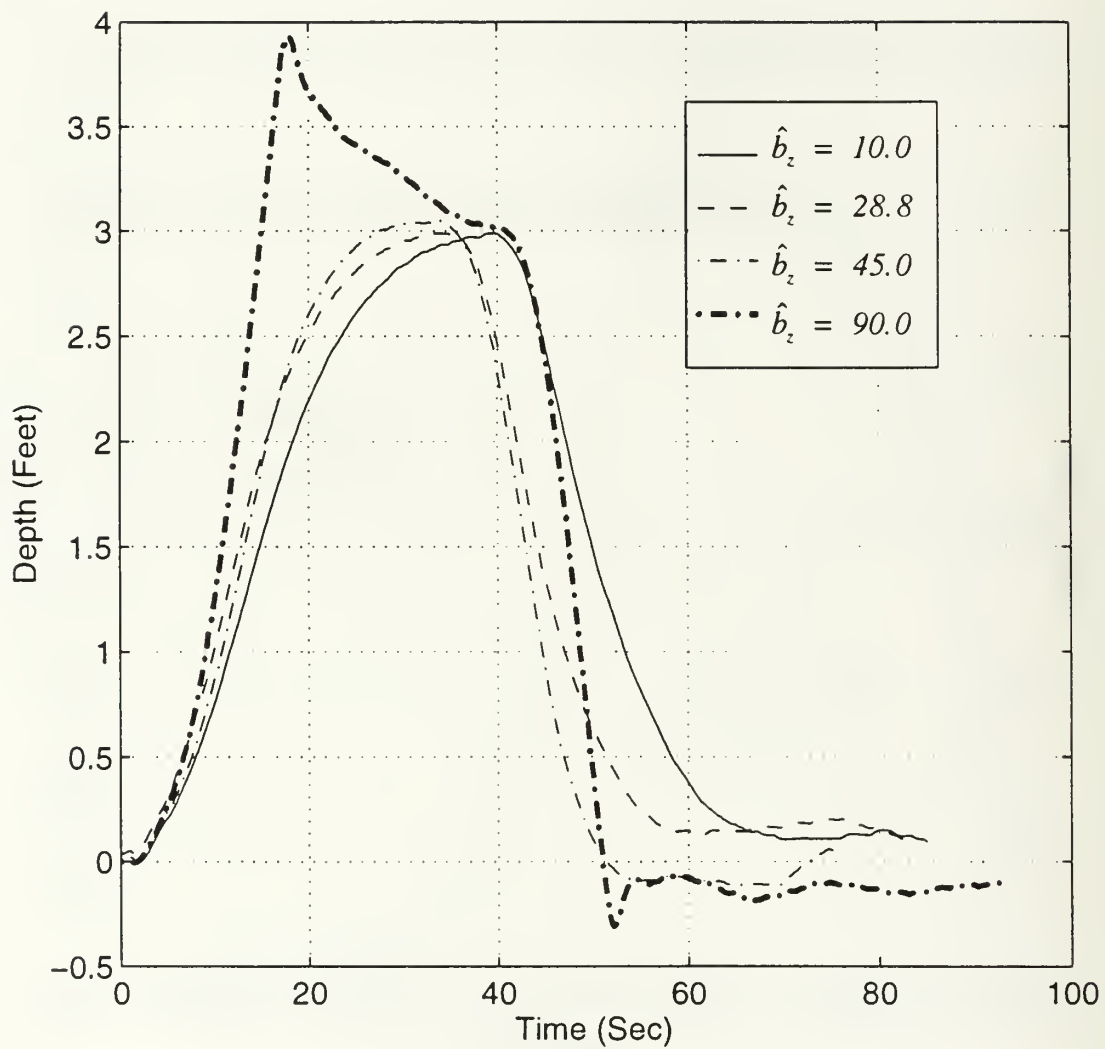
Figure 5.14 Experimental Depth Response vs. Time for Different Values of
Vertical Damping, $\hat{b}_z$.
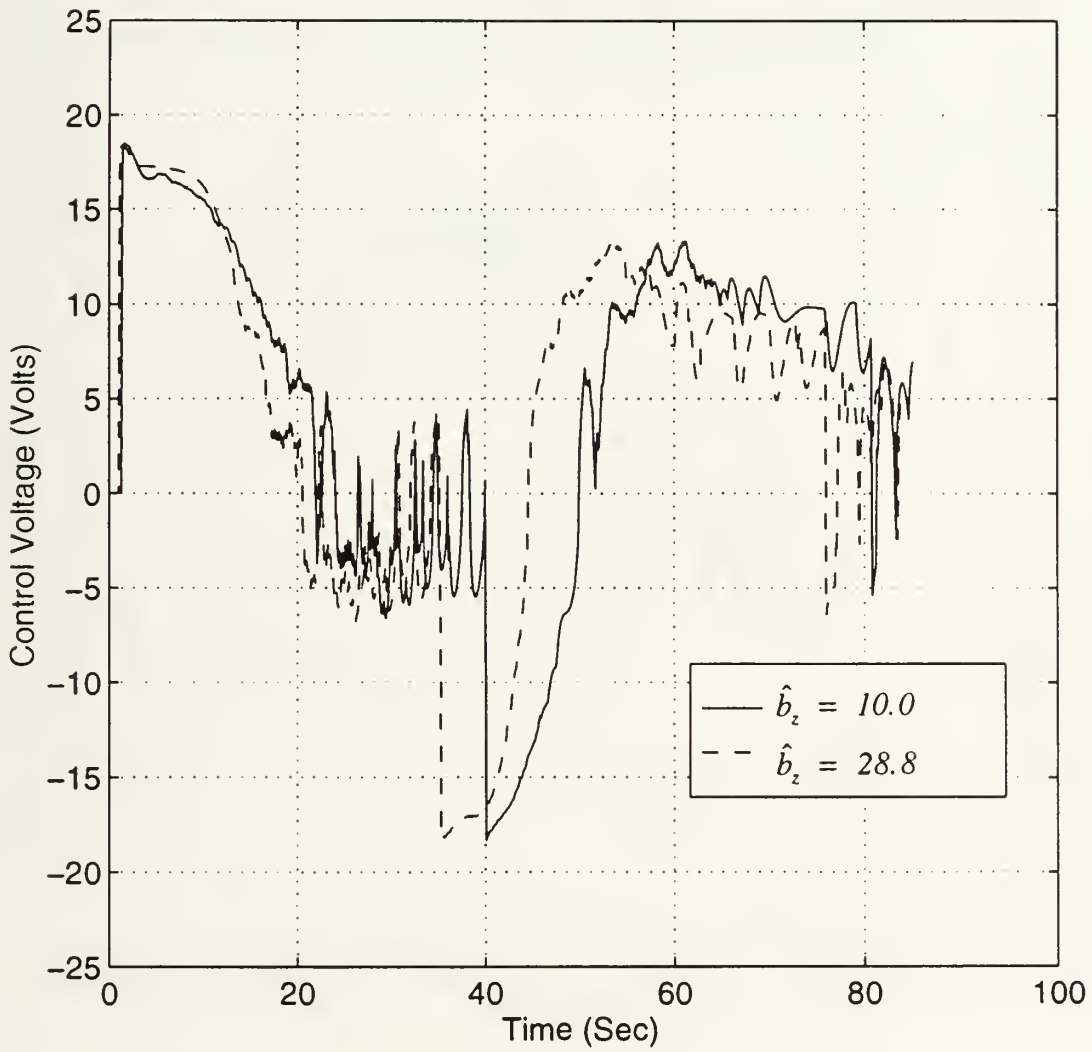
Figure 5.15 Experimental Vertical Thruster Control Voltage vs. Time for Different
Values of Vertical Damping, $\hat{b}_z$.
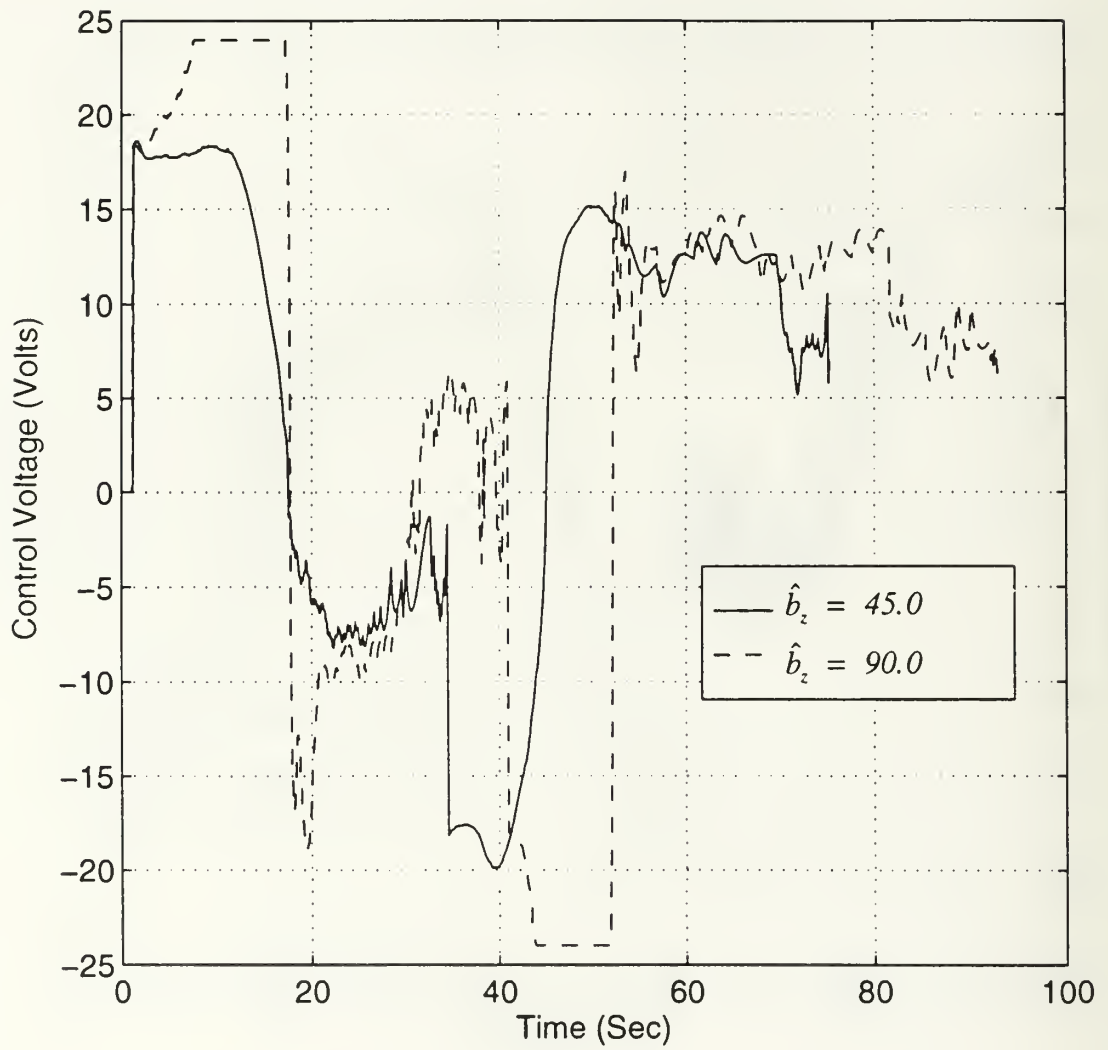
Figure 5.16 Experimental Vertical Thruster Control Voltage vs. Time for Different
Values of Vertical Damping, $\hat{b}_z$.
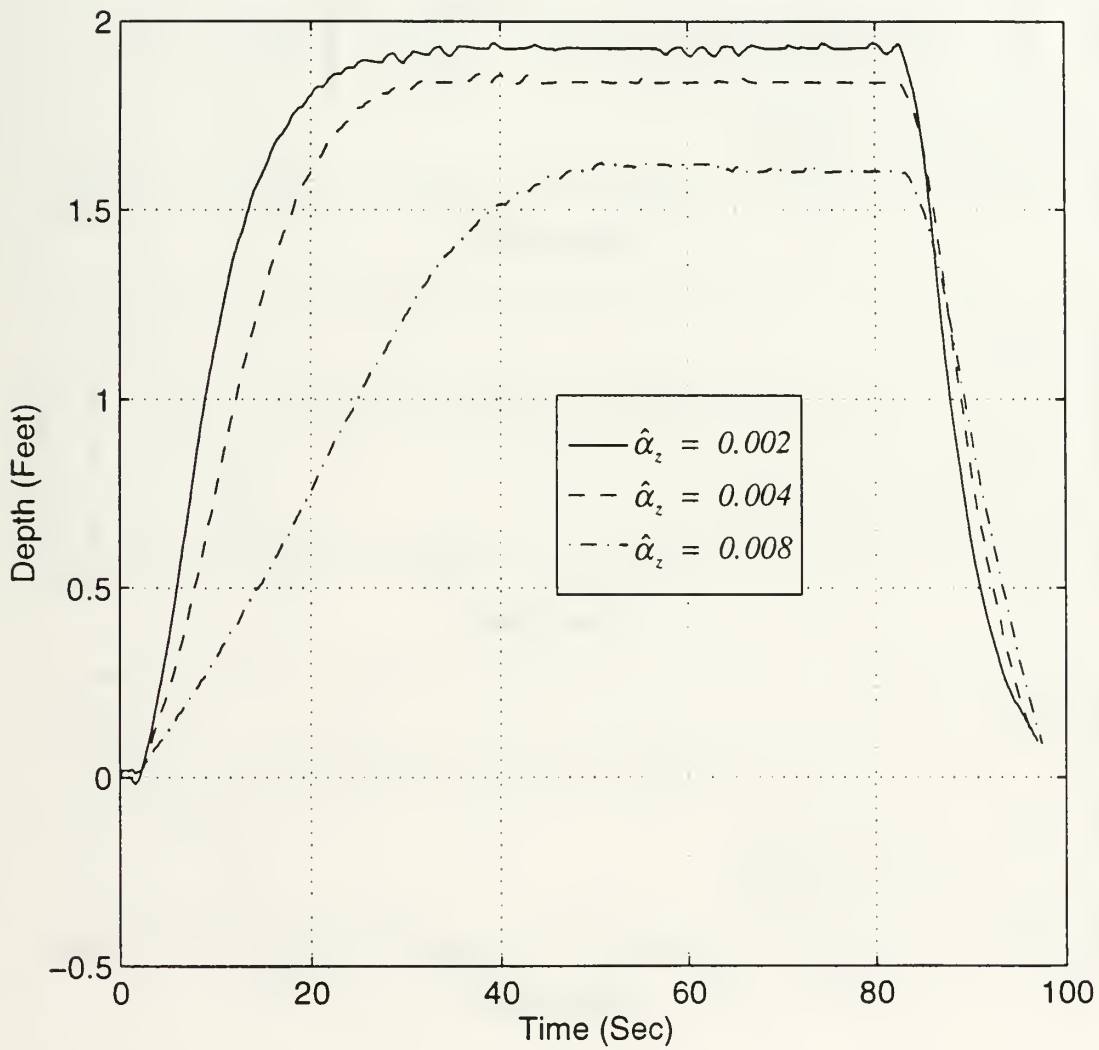
Figure 5.17 Experimental Depth Response vs. Time for Different Values of
Thruster Effectiveness, $\hat{\alpha}_z$.
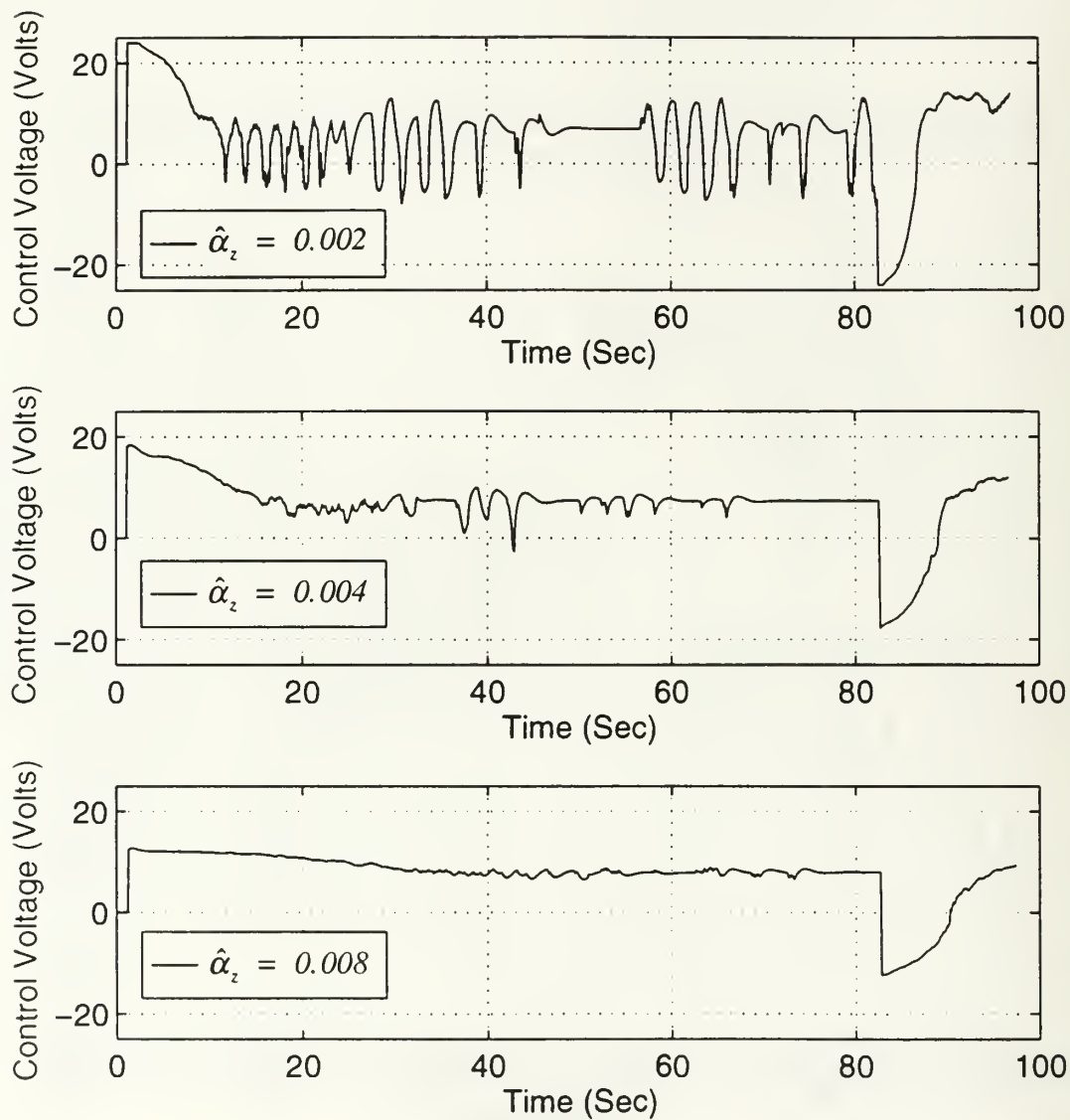
Figure 5.18 Experimental Vertical Thruster Control Voltage vs. Time for Different
Values of Thruster Effectiveness, $\hat{\alpha}_z$.
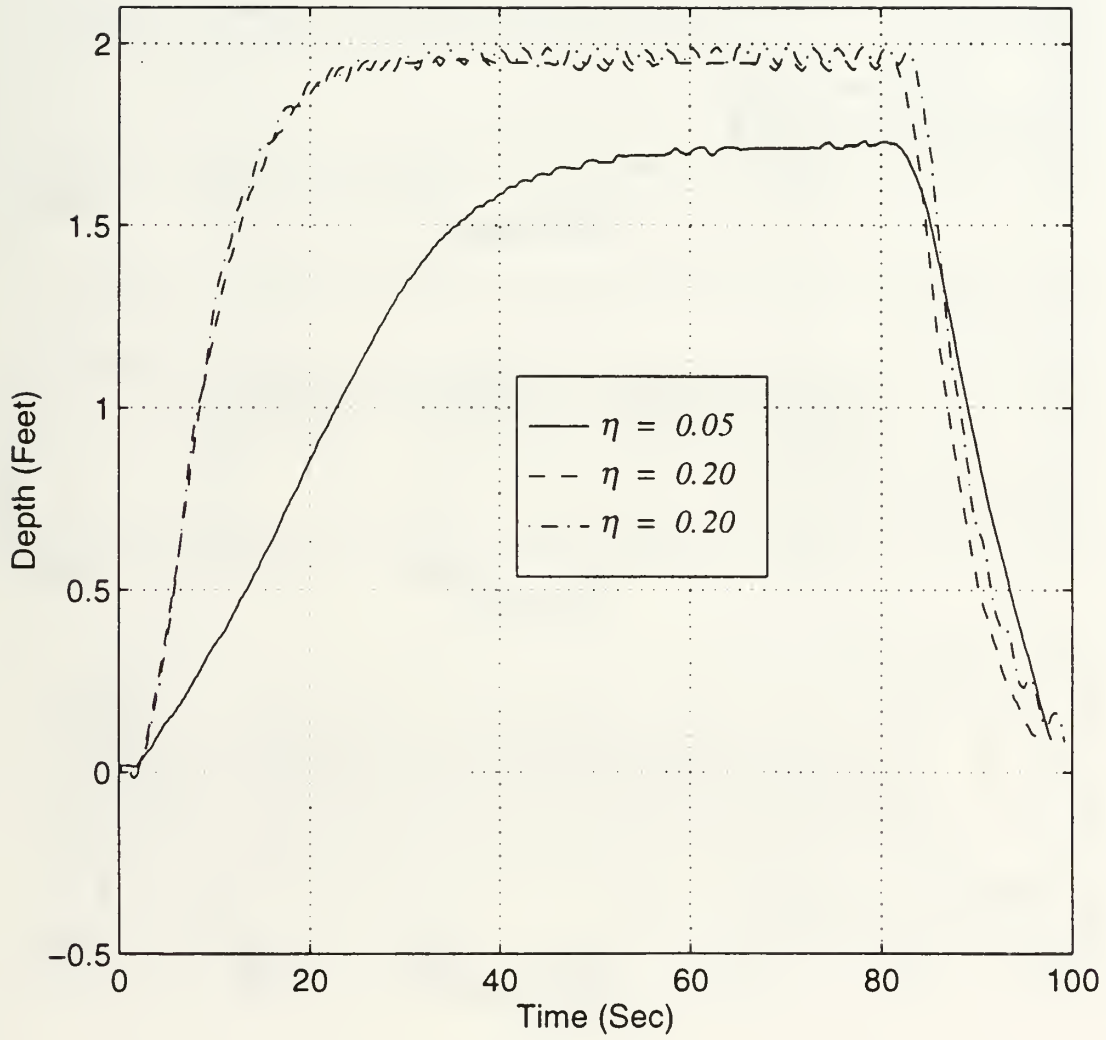
Figure 5.19 Experimental Depth Response vs. Time for Different Values of
Switching Gain, η.
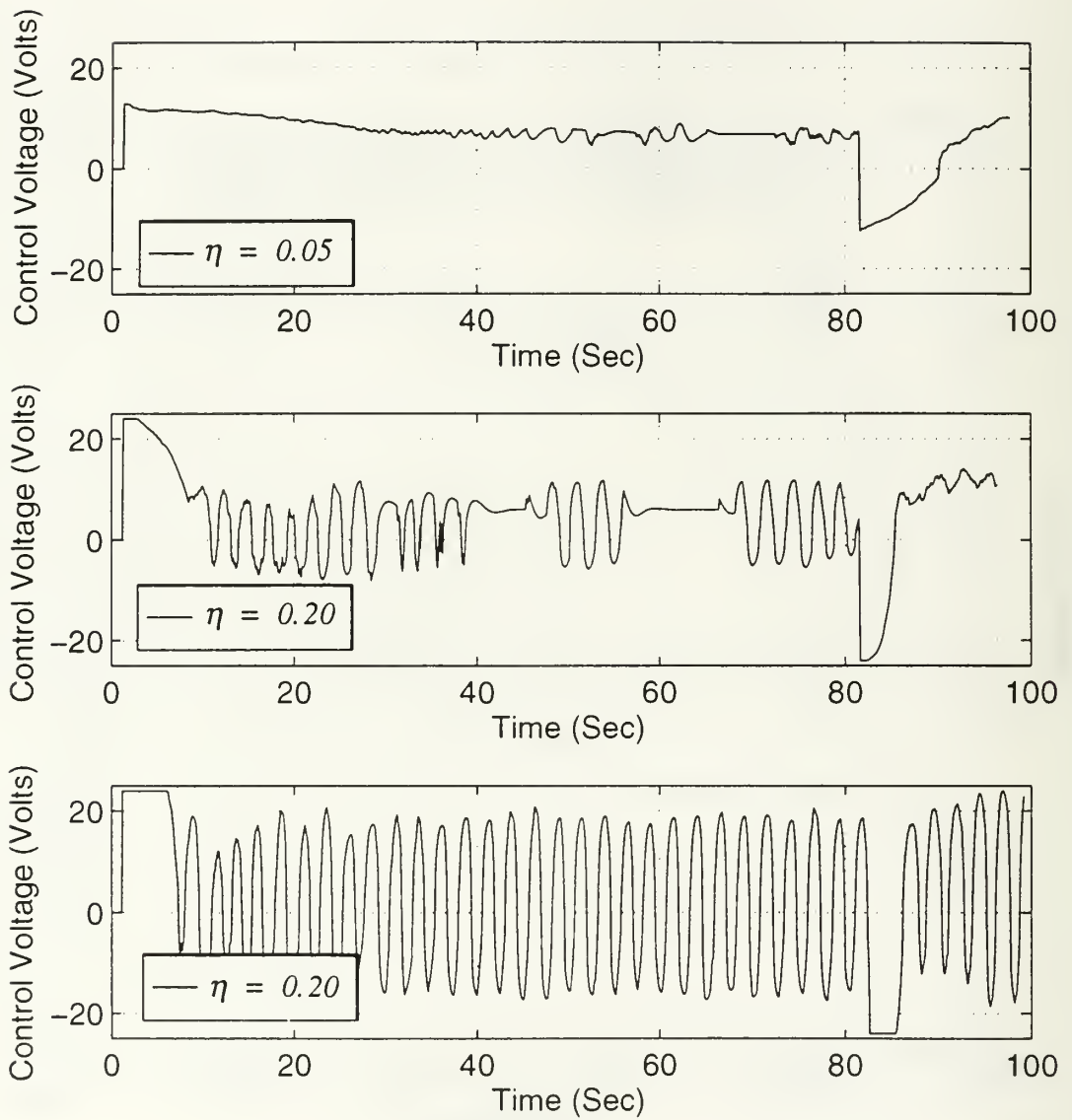
Figure 5.20 Experimental Vertical Thruster Control Voltage vs. Time for Different Values of Switching Gain, η.
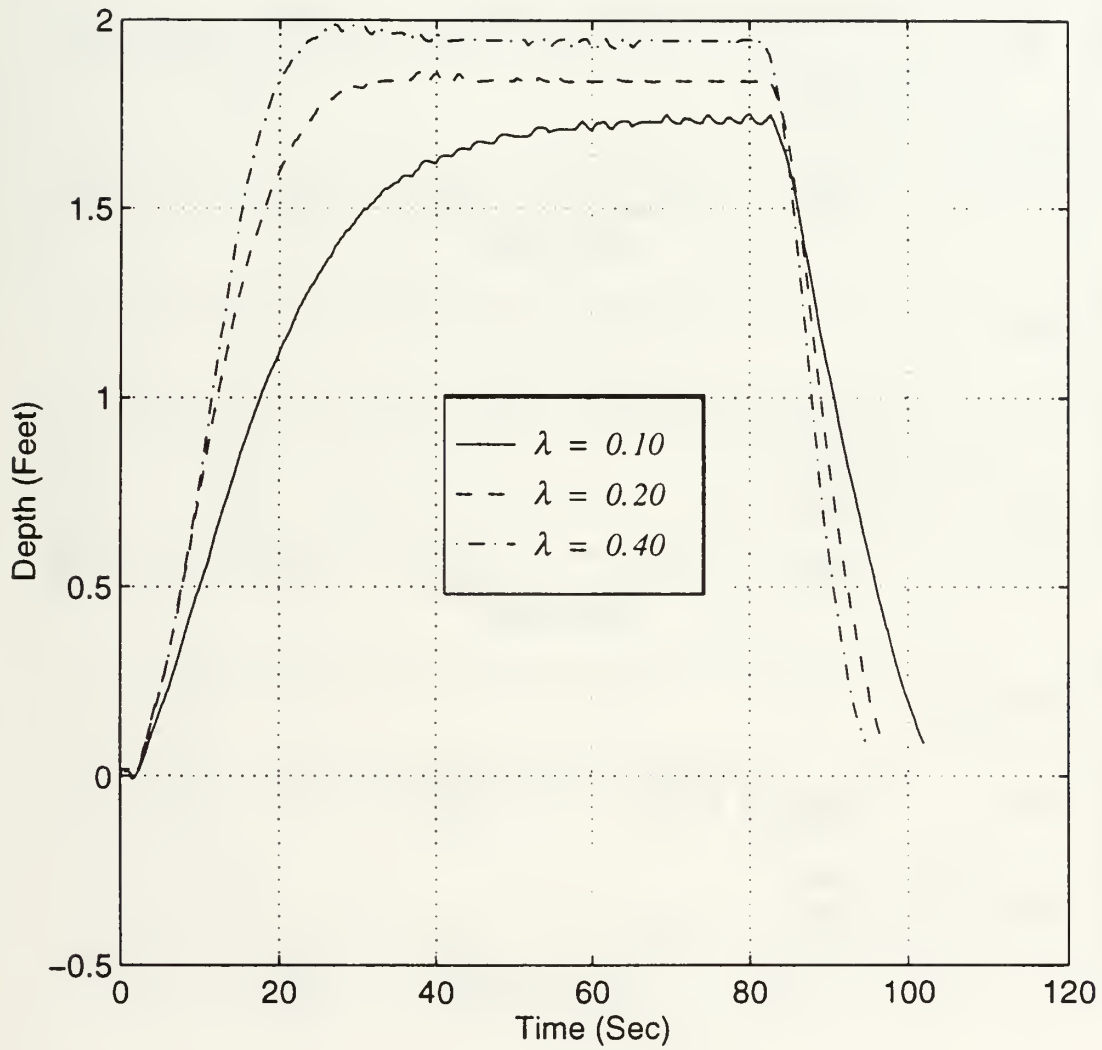
Figure 5.21 Experimental Depth Response vs. Time for Sliding Surface Position
Error Coefficient, λ.
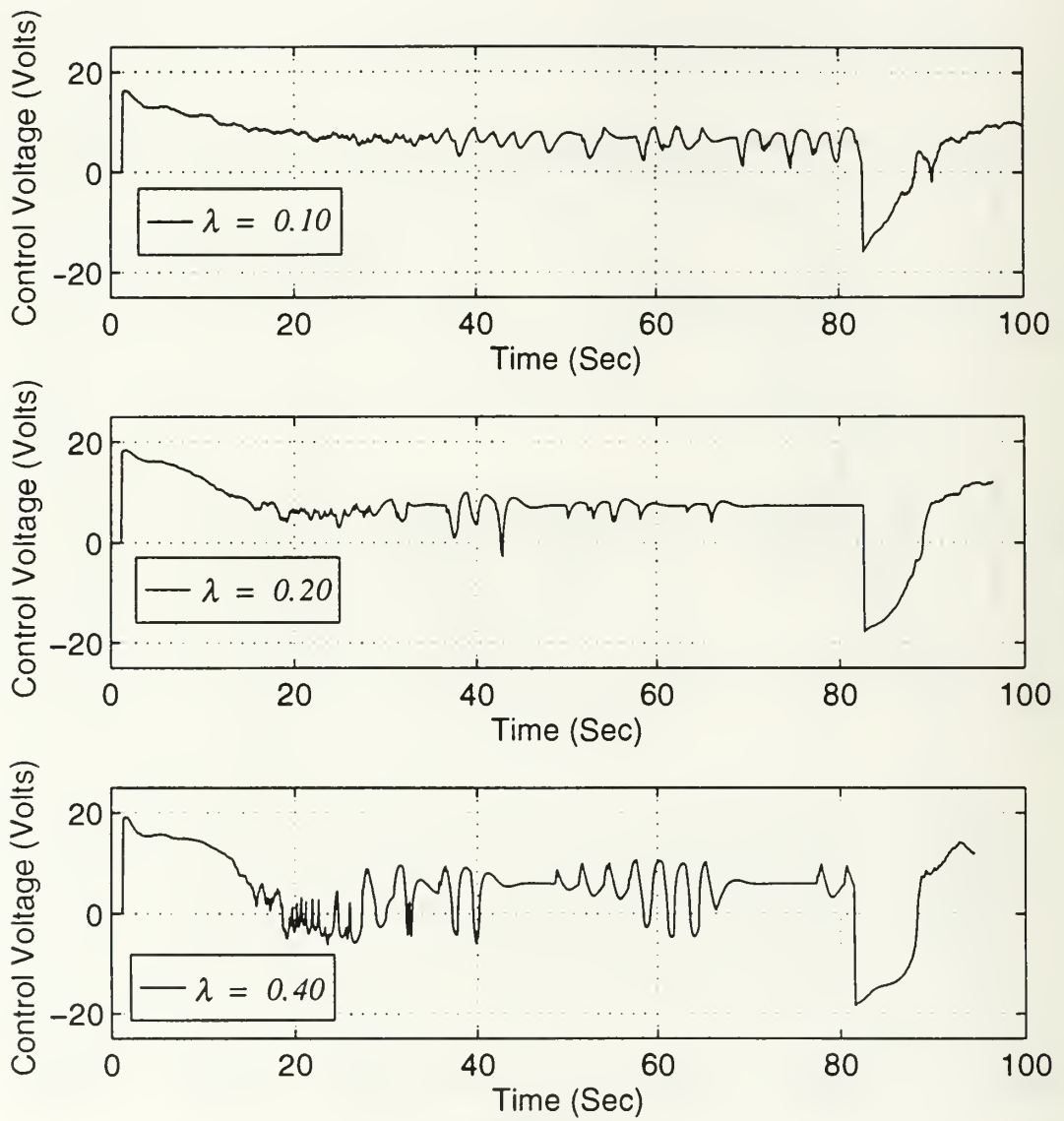
Figure 5.22 Experimental Vertical Thruster Control Voltage vs. Time for Sliding
Surface Position Error Coefficient, λ.
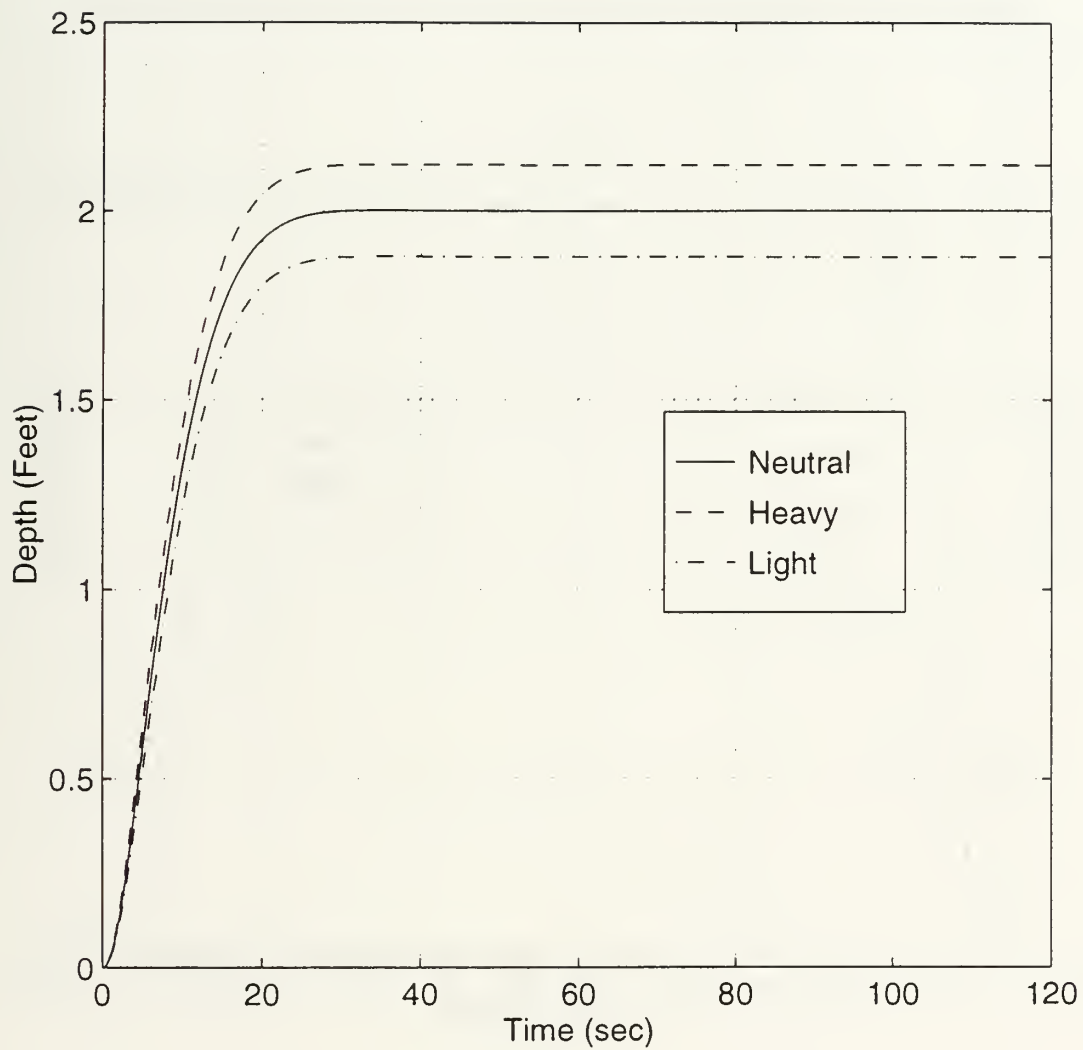
Figure 5.23 Simulated Depth Response vs. Time for Light, Heavy and Neutrally Buoyant Vehicle.
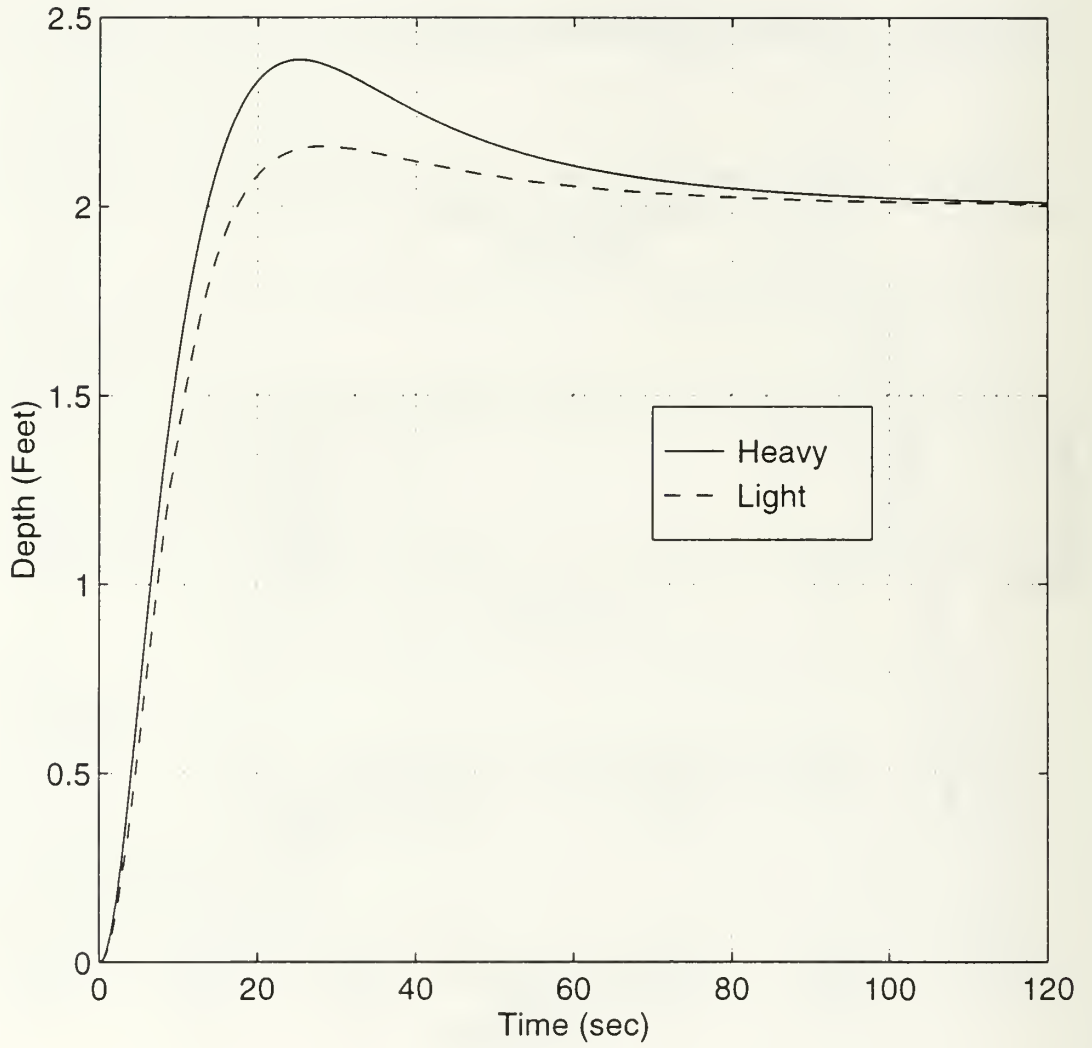
Figure 5.24 Simulated Depth Response vs. Time for Integral Control starting at
$t = 0.0$ Seconds.
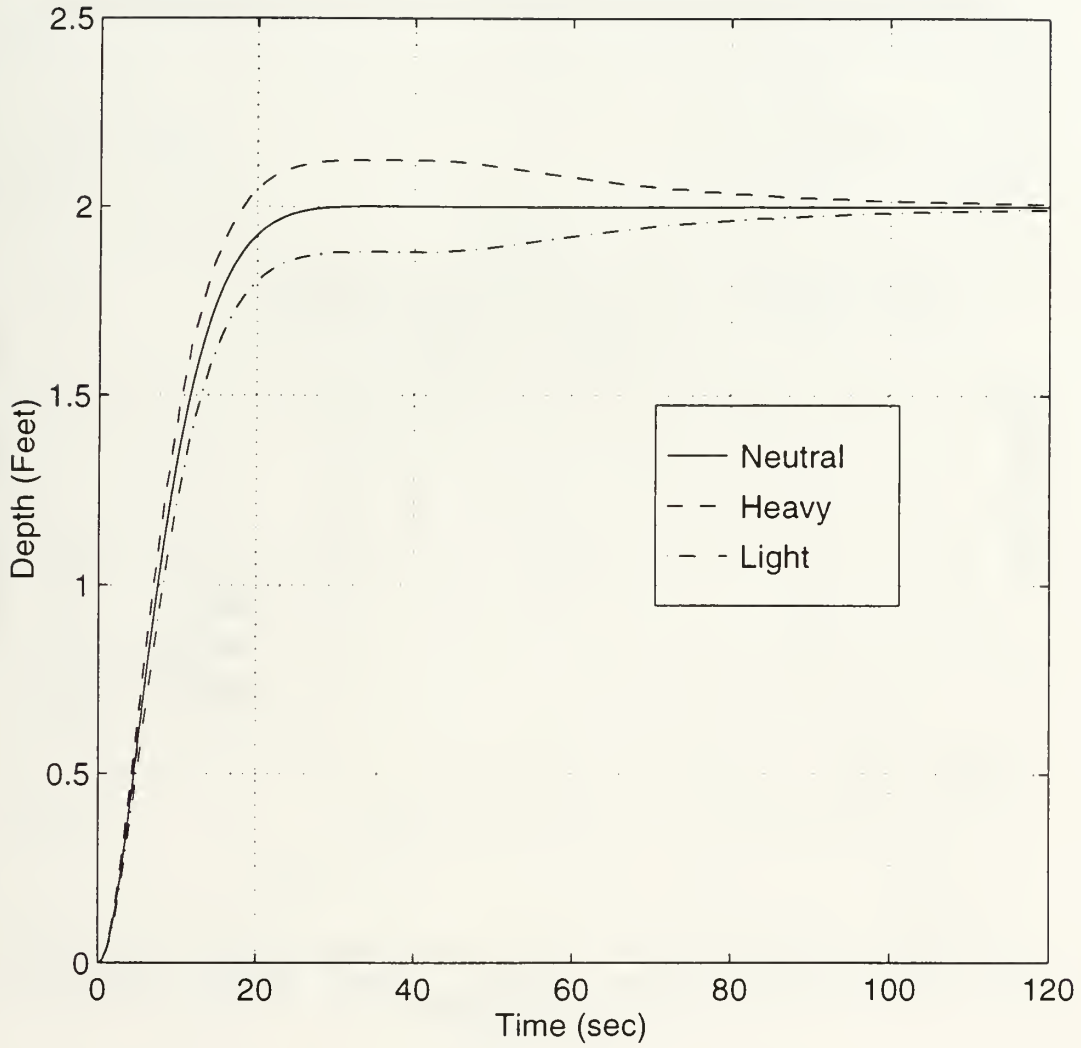
Figure 5.25 Simulated Depth Response vs. Time for Integral Control Starting at a
Preset Time of 40.0 Seconds.
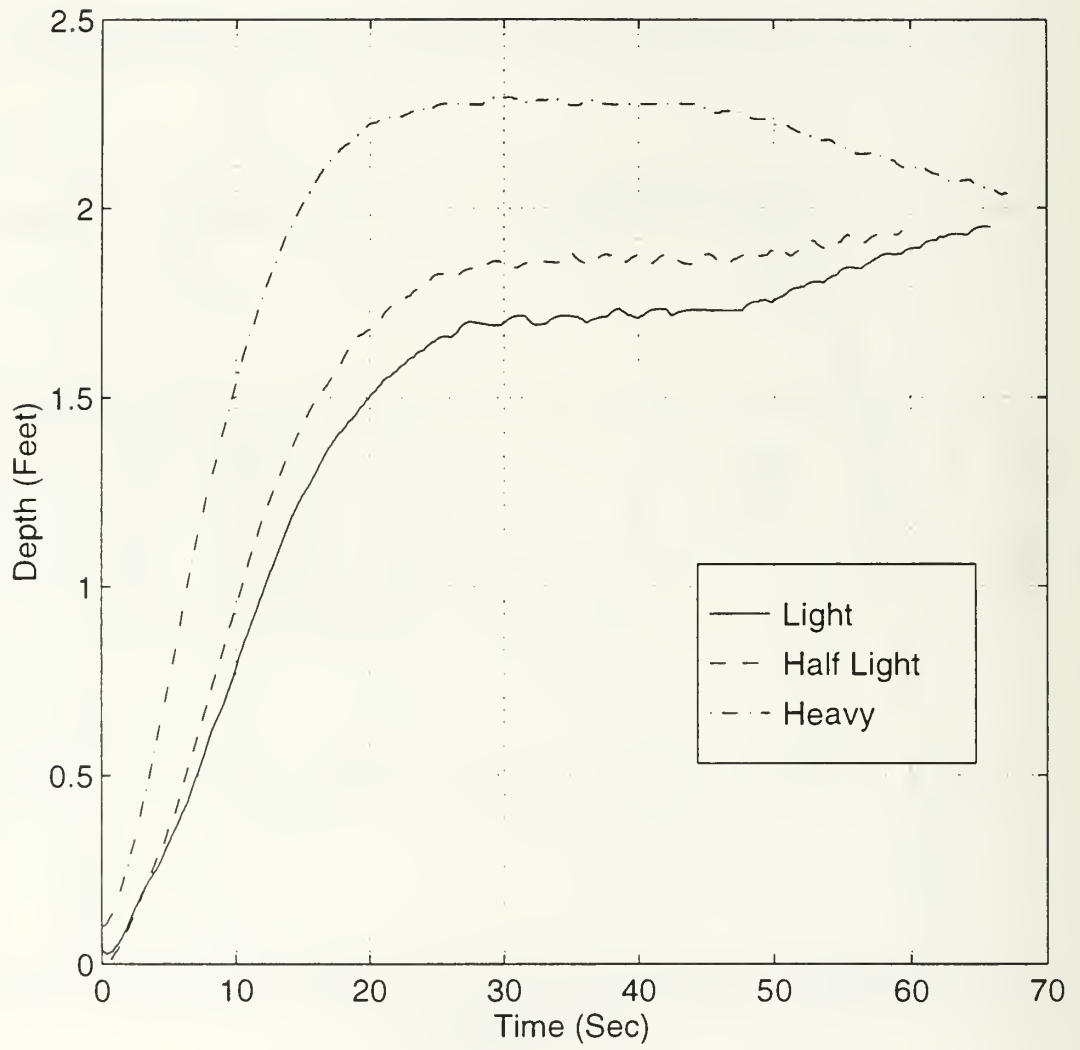
Figure 5.26 Experimental Depth Response vs. Time Using Integral Control.
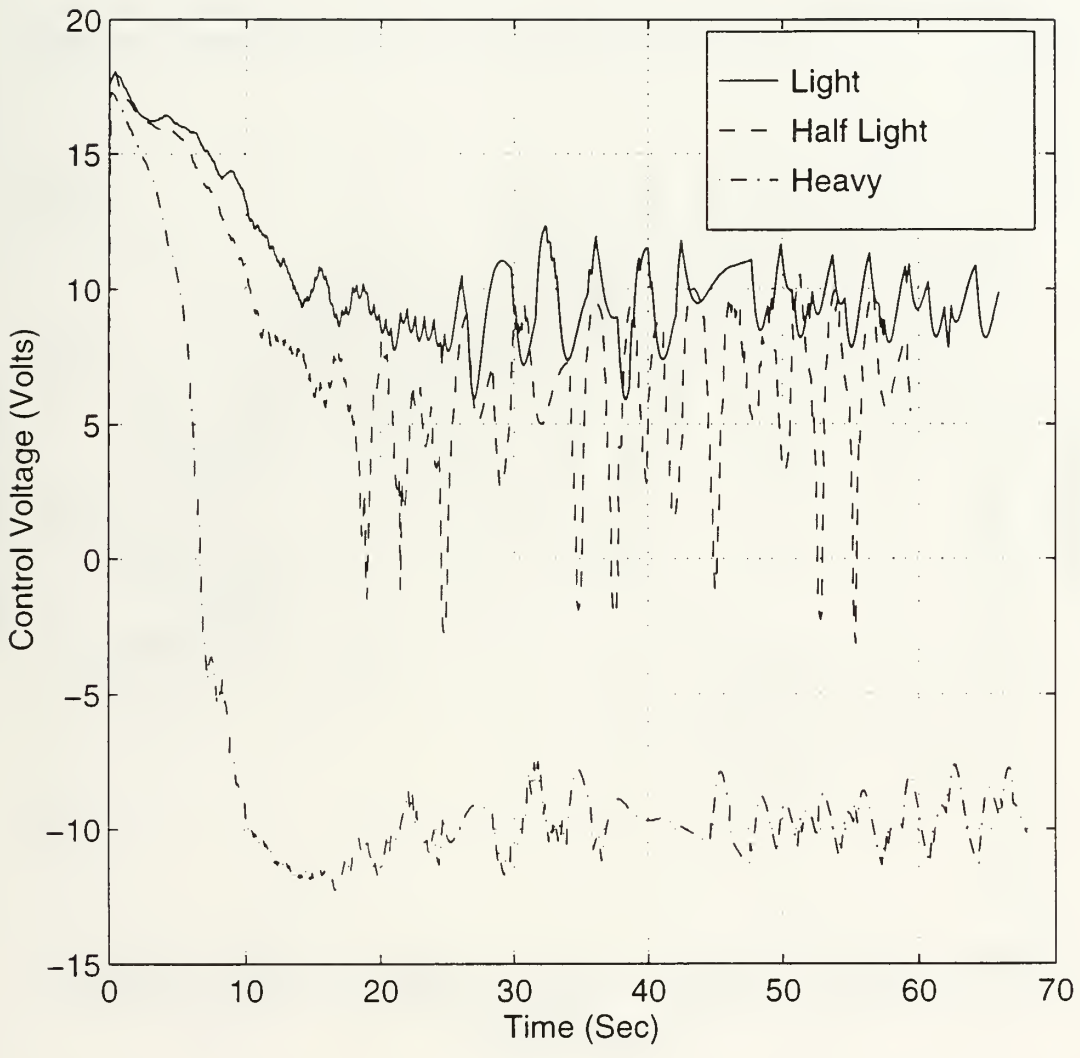
Figure 5.27 Experimental Vertical Thruster Control Voltage vs. Time Using Integral Control.

207
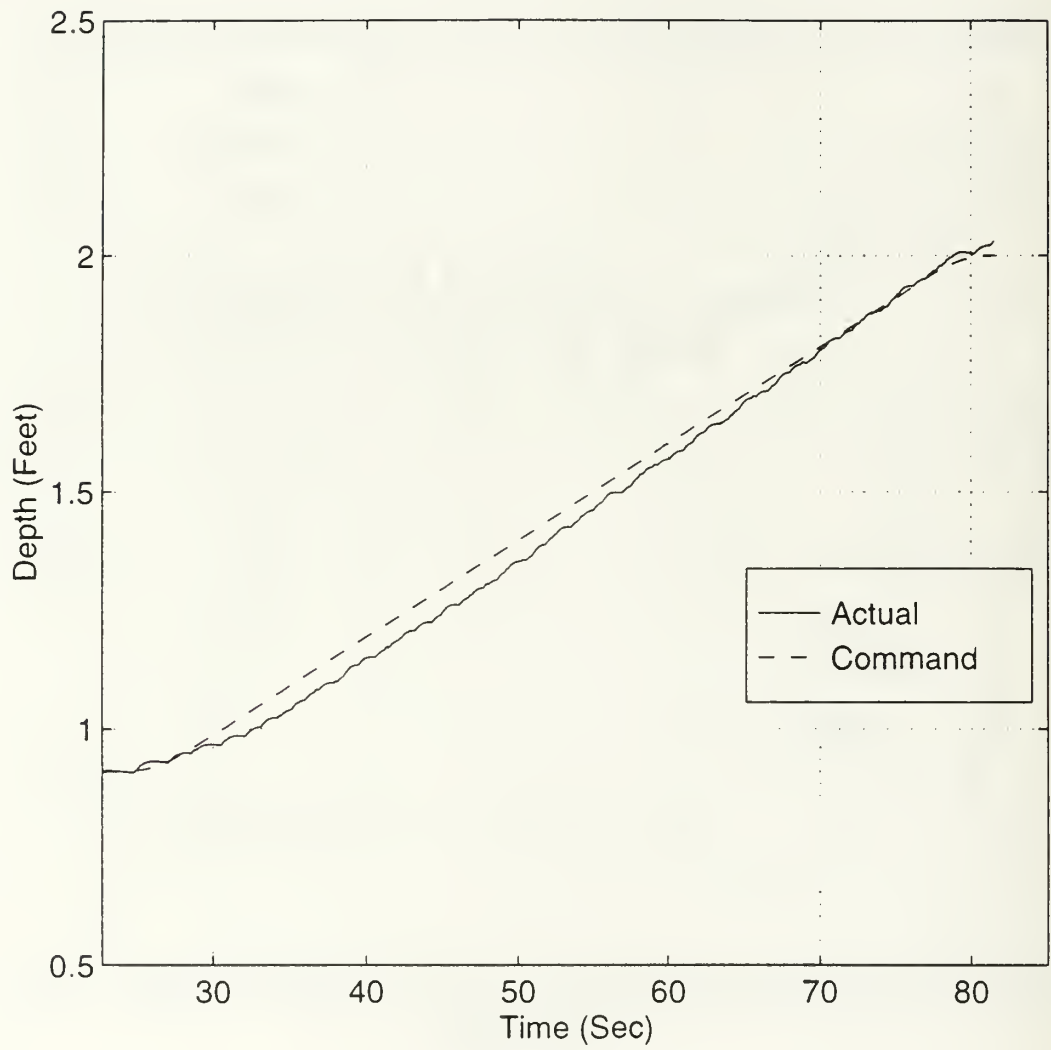
Figure 5.28 Depth Response vs. Time Using Command Generator Design 1.
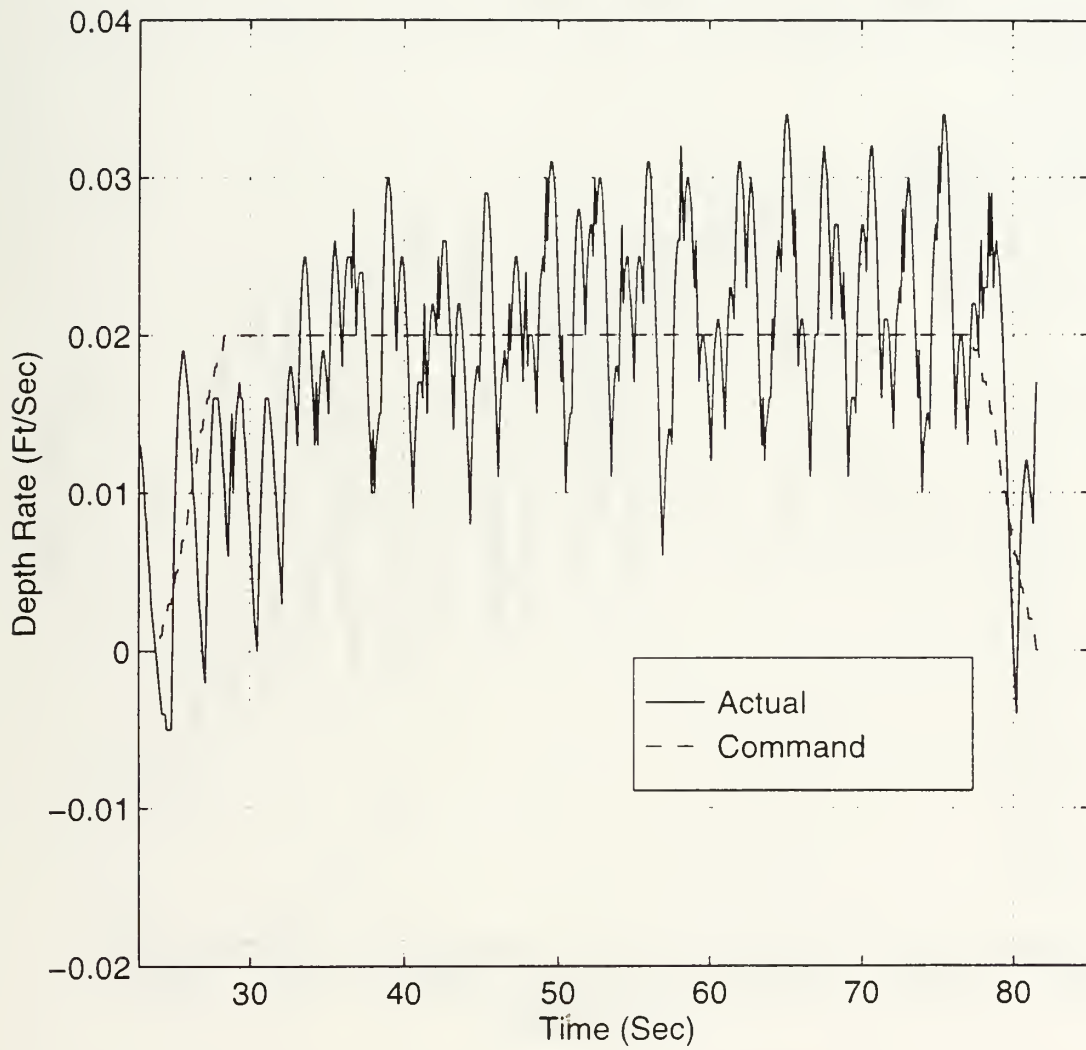
Figure 5.29 Depth Rate Response vs. Time Using Command Generator Design 1.
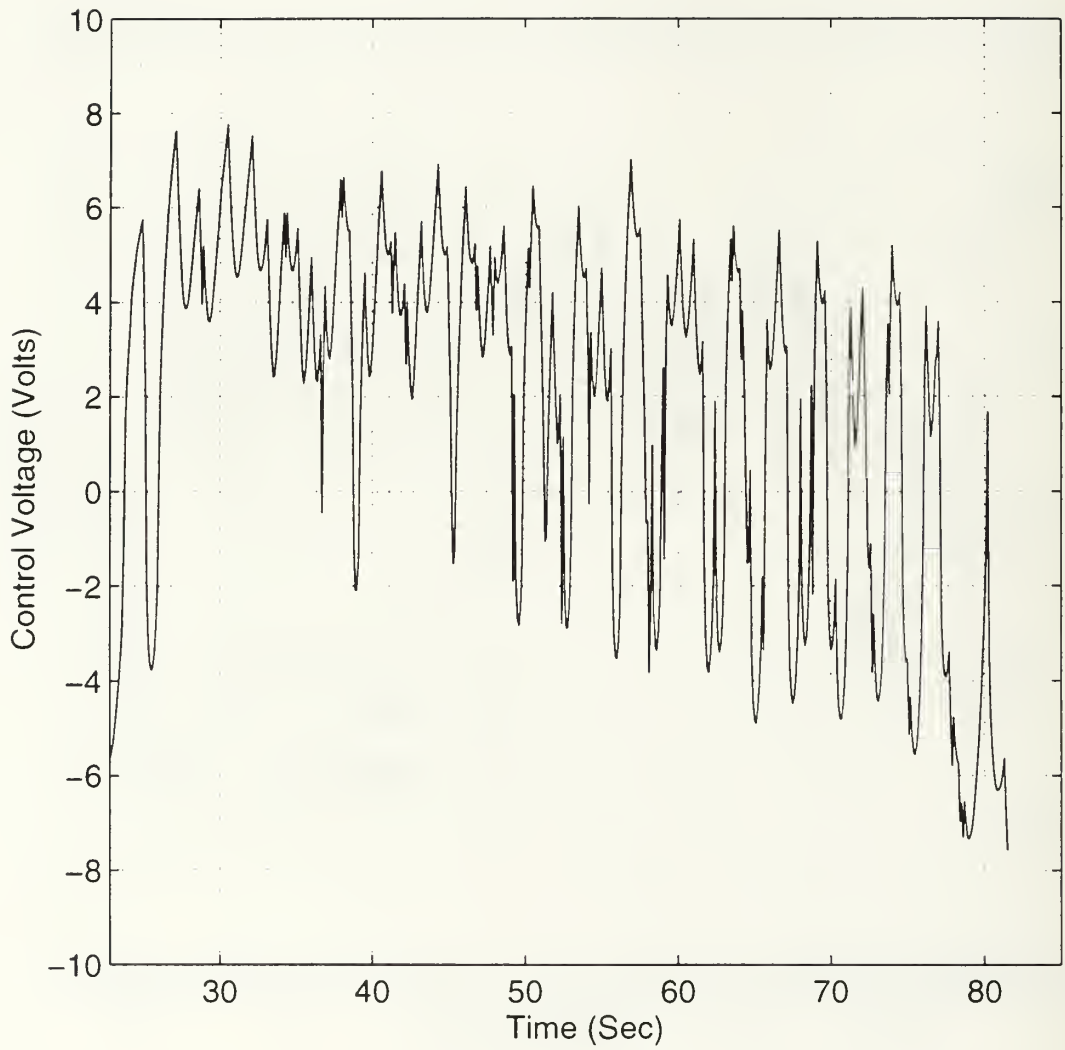
Figure 5.30 Vertical Thruster Control Voltage vs. Time Using Command
Generator Design 1.
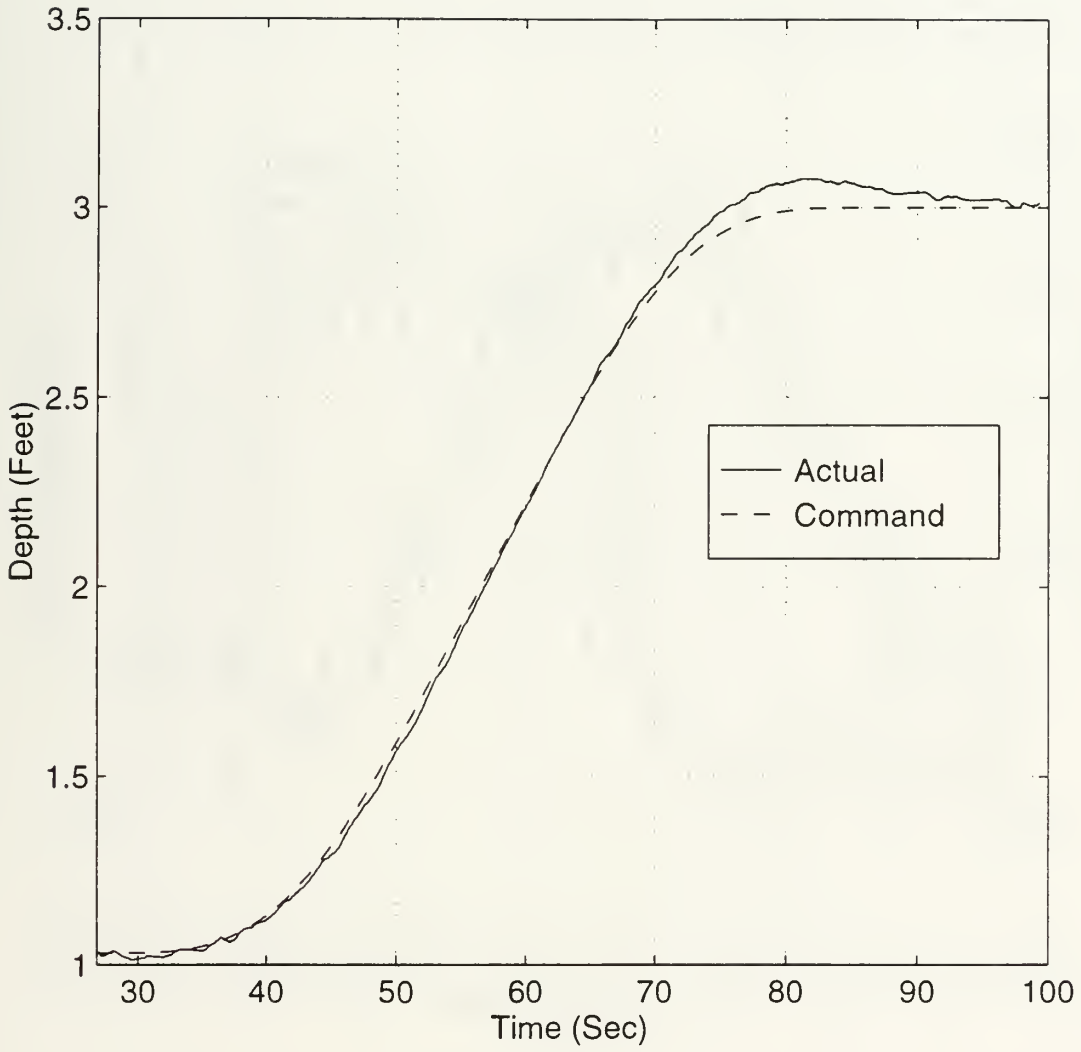
Figure 5.31 Depth Response vs. Time Using Command Generator Design 2.
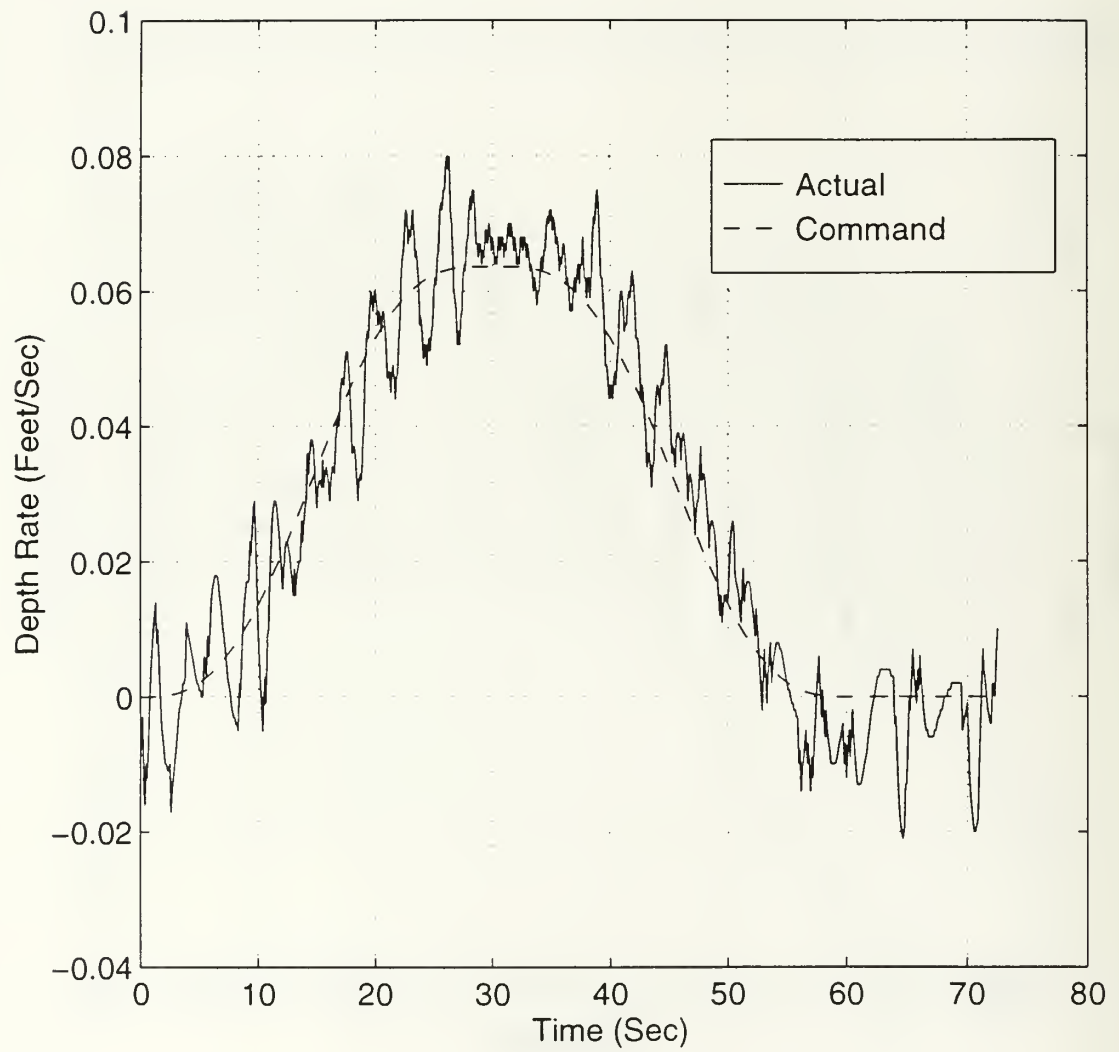
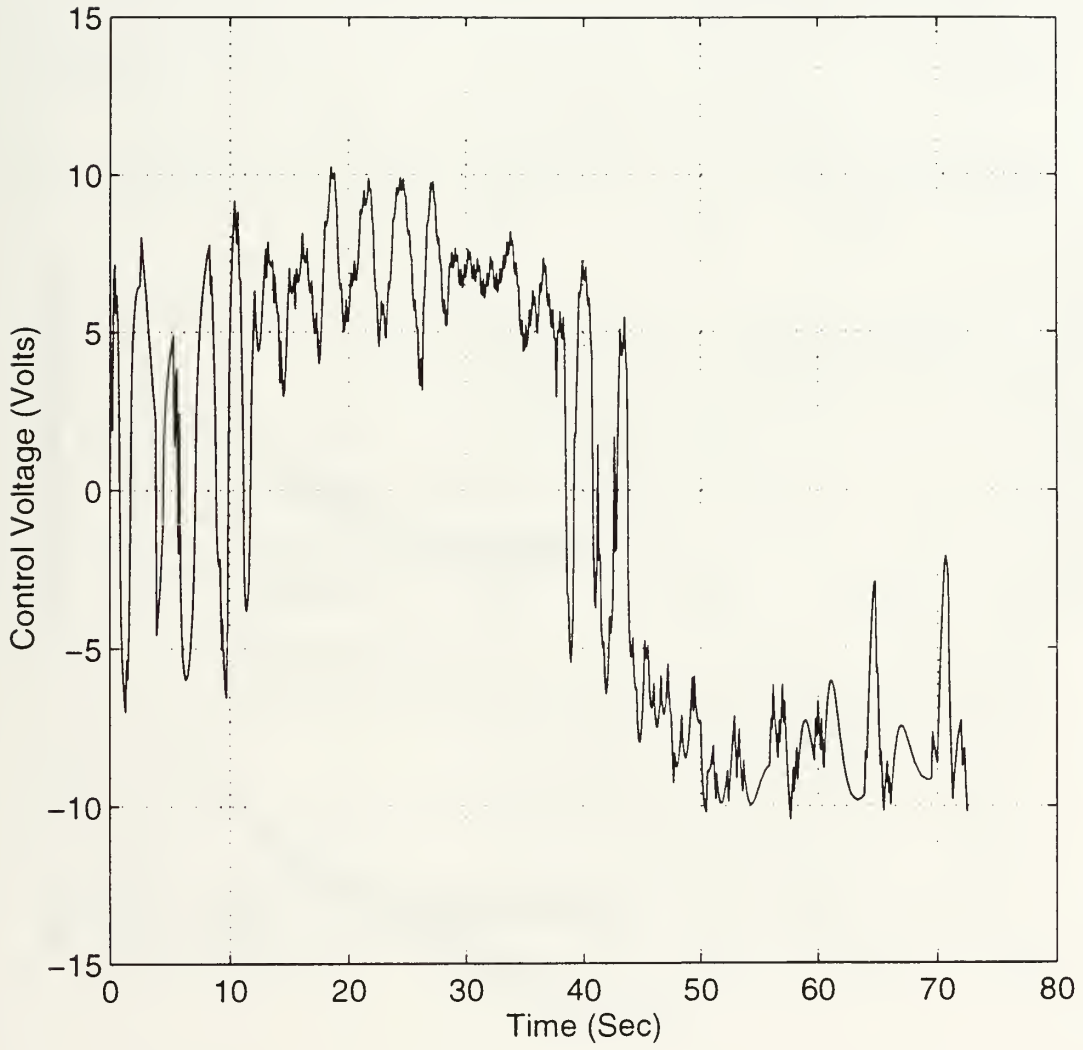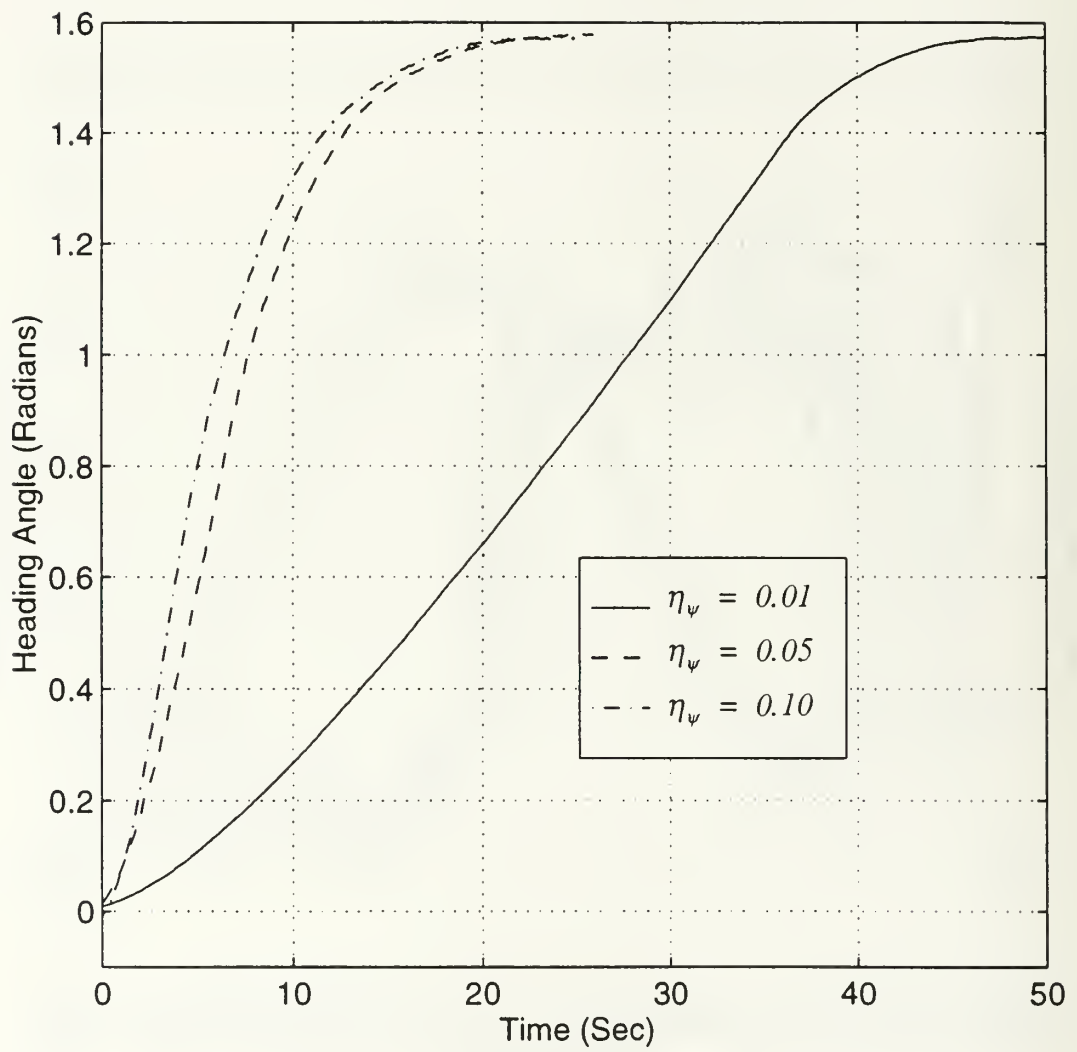Figure 5.32 Depth Rate Response vs. Time Using Command Generator Design 2.

Figure 5.33 Vertical Thruster Control Voltage vs. Time Using Command
Generator Design 2.

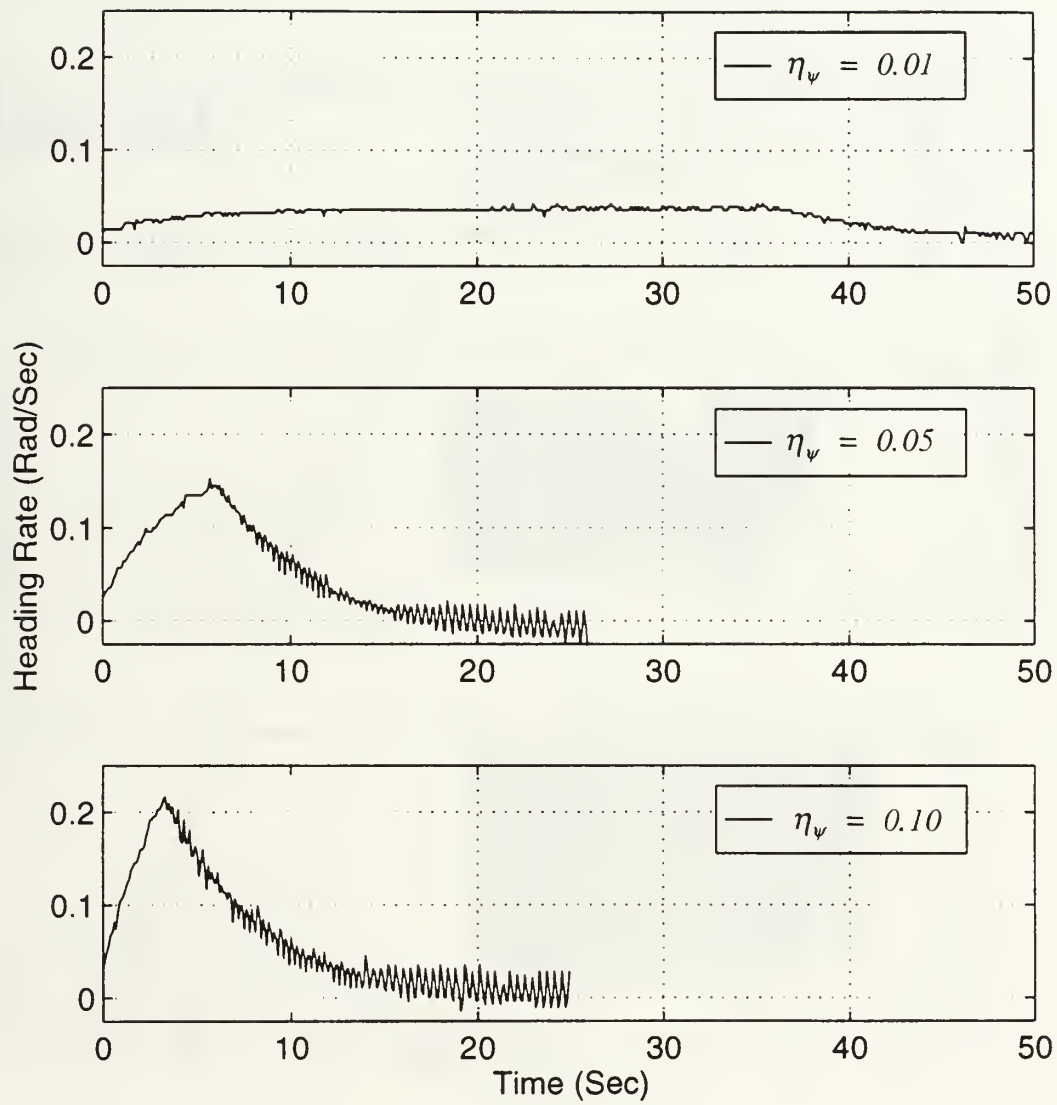Figure 5.34 Heading Angle vs. Time Step Response for Switching Gain, η_ψ.

214

Figure 5.35 Heading Rate Response vs. Time for Switching Gain, η_ψ.
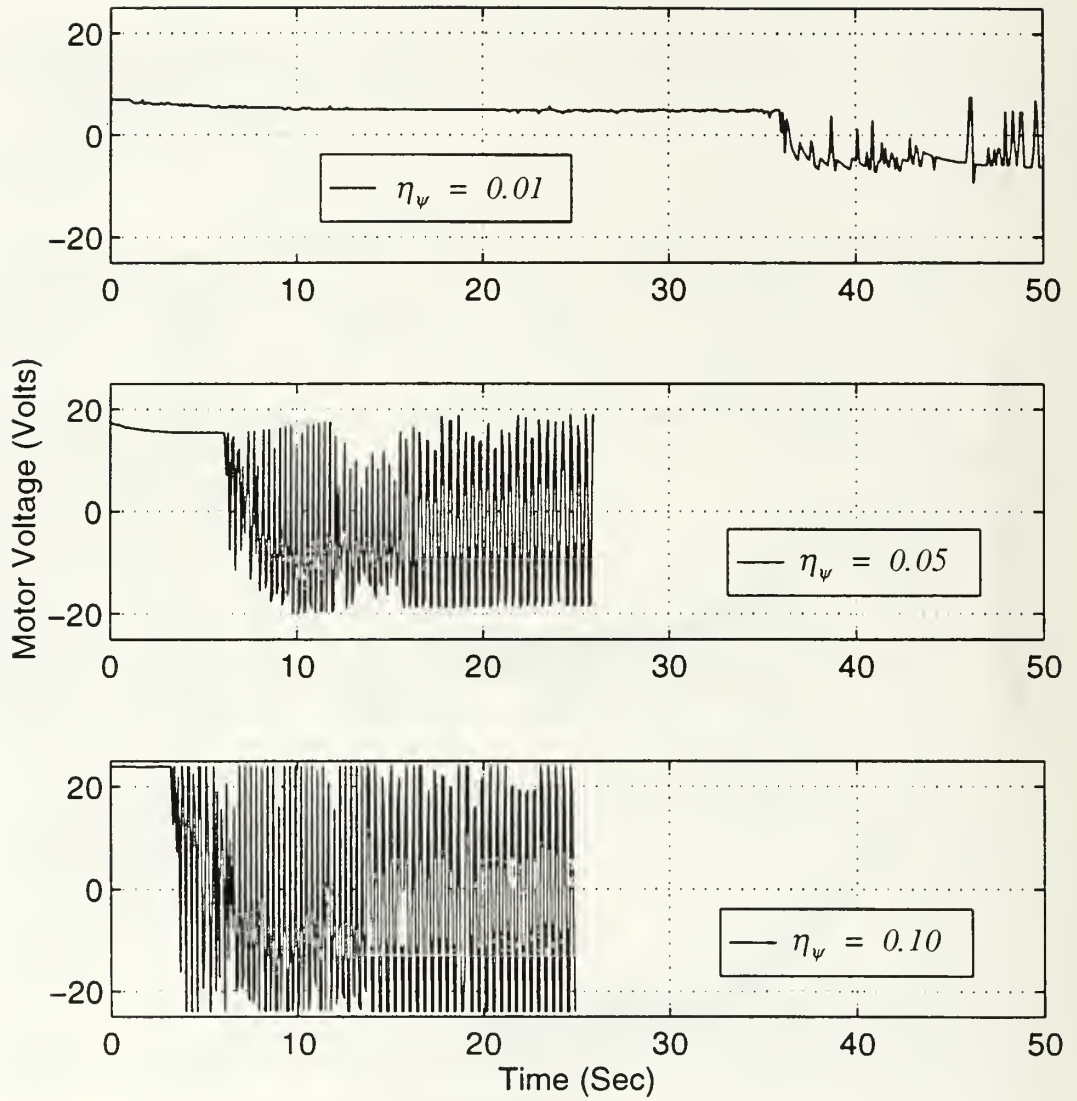
215

Figure 5.36 Lateral Thruster Control Voltage vs. Time for Switching Gain, η_ψ.
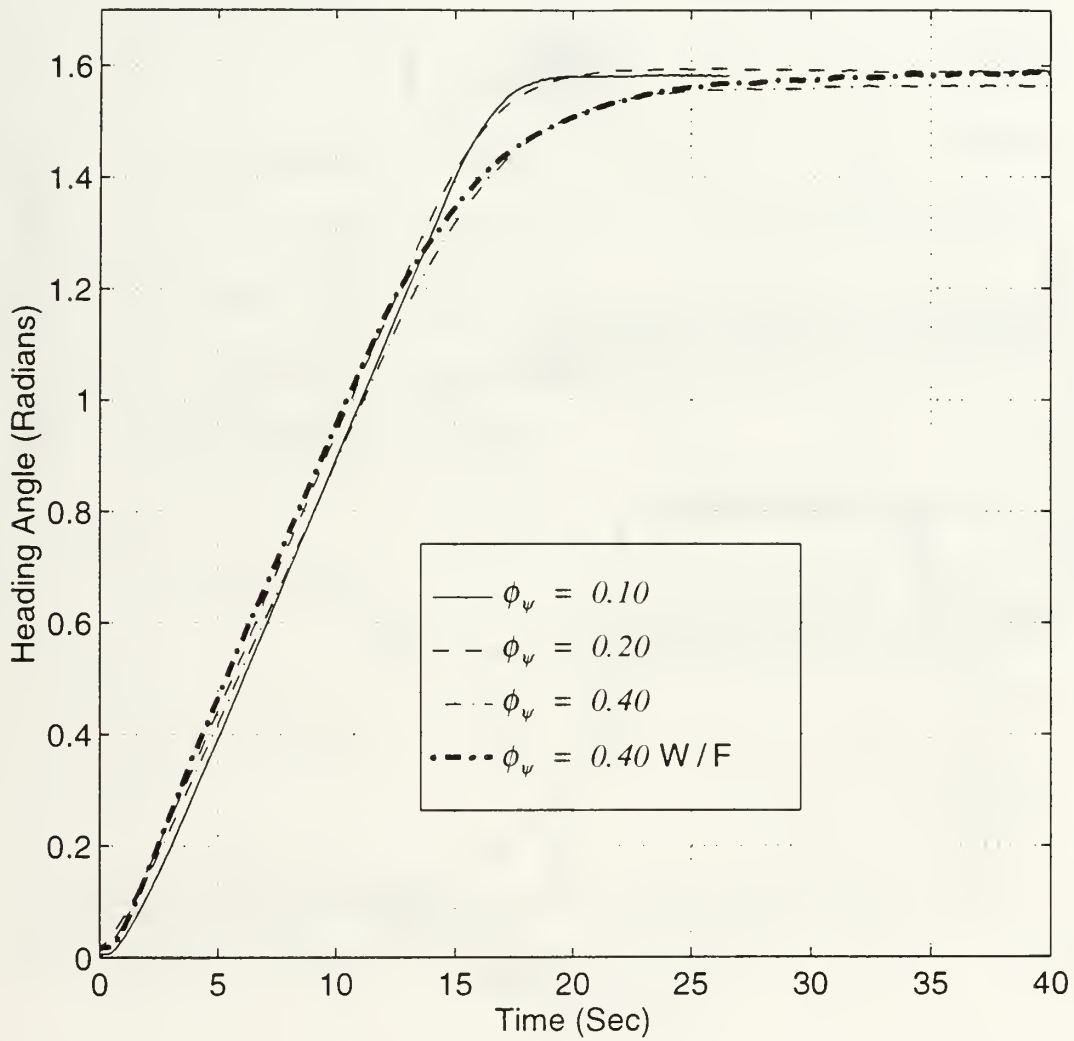
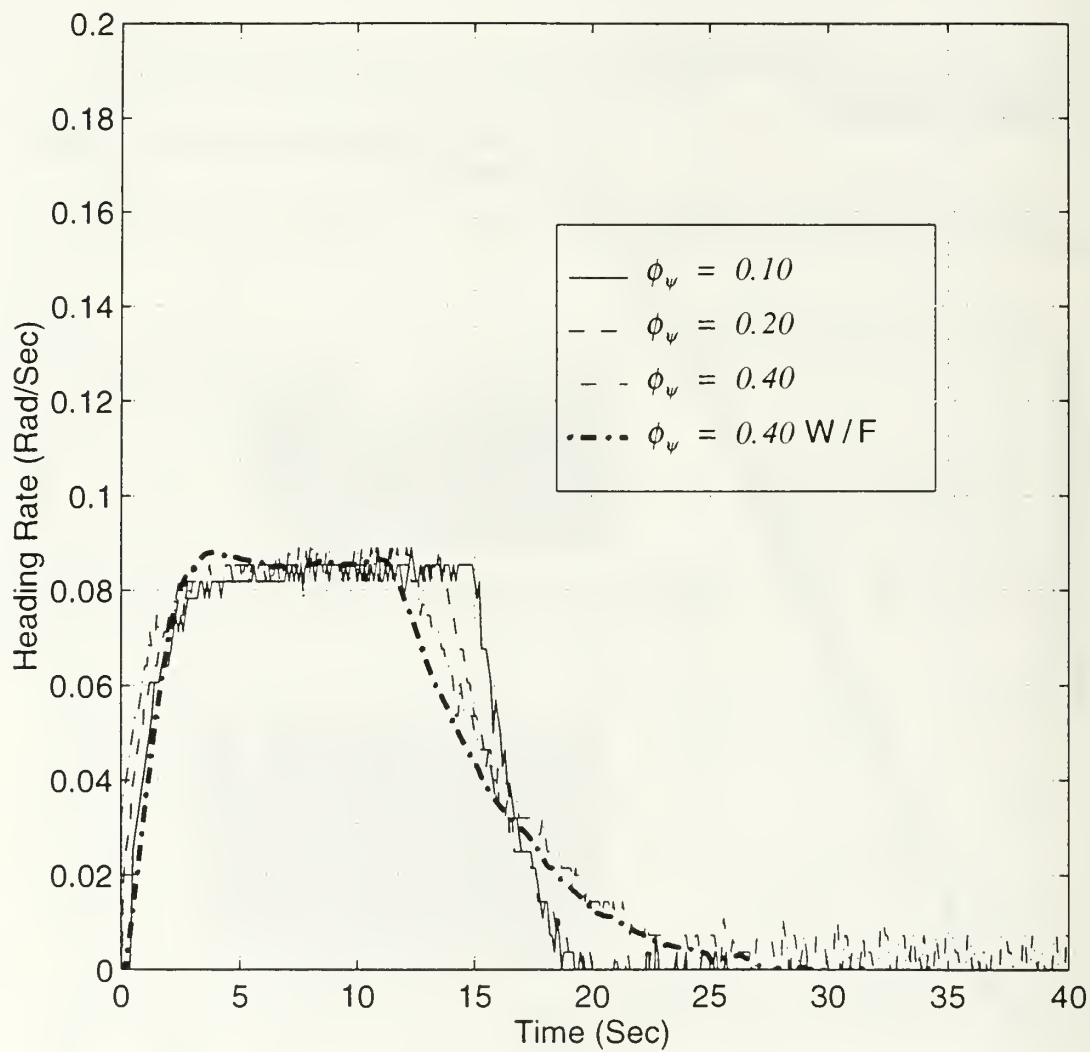Figure 5.37 Heading Angle Step Response vs. Time for Switch Saturation Level, ϕ_ψ.

Figure 5.38 Heading Rate Response vs. Time for Switch Saturation Level, ϕ_ψ.
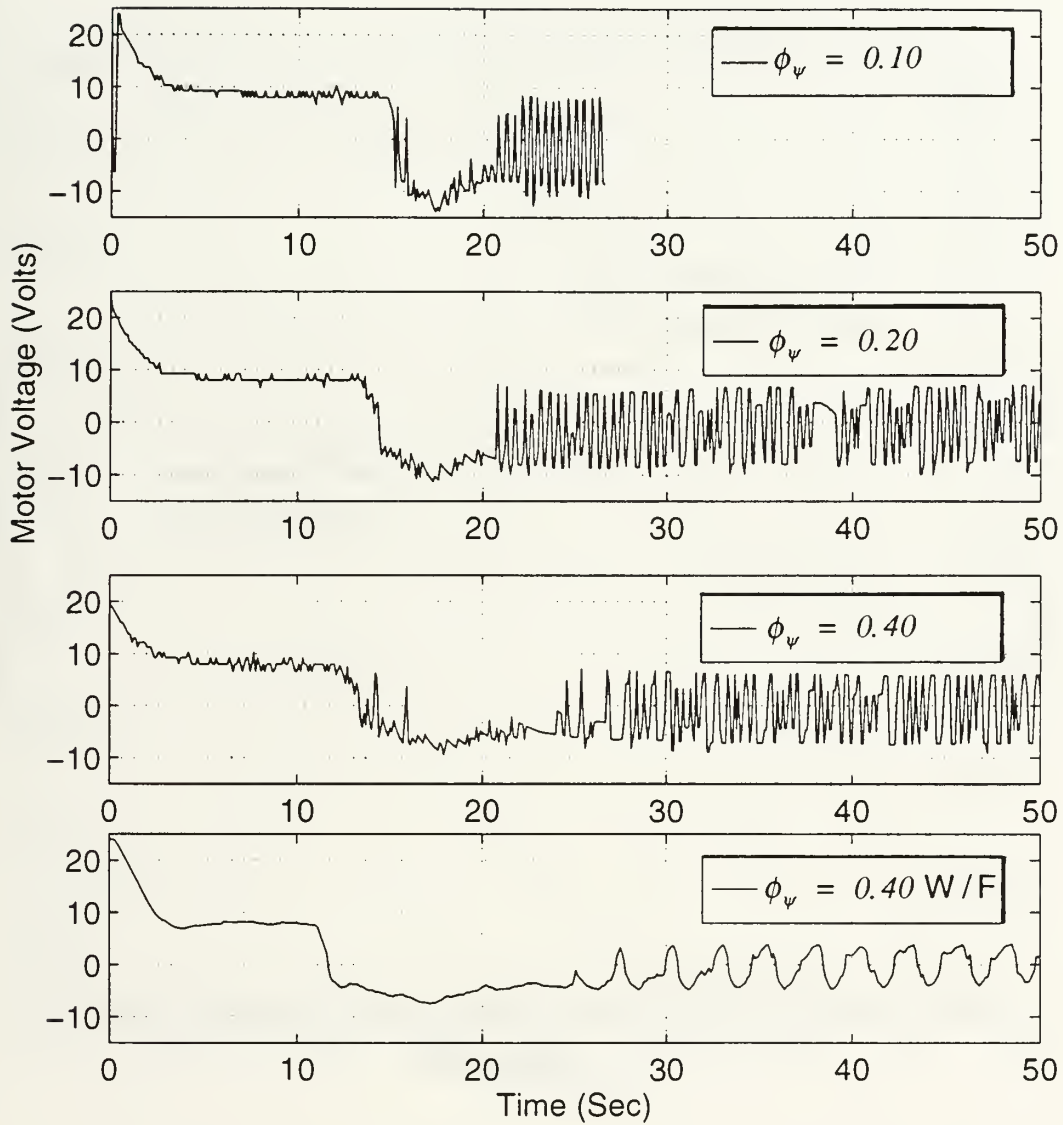
Figure 5.39 Lateral Thruster Control Voltage vs. Time for Switch Saturation Level, ϕ_ψ.
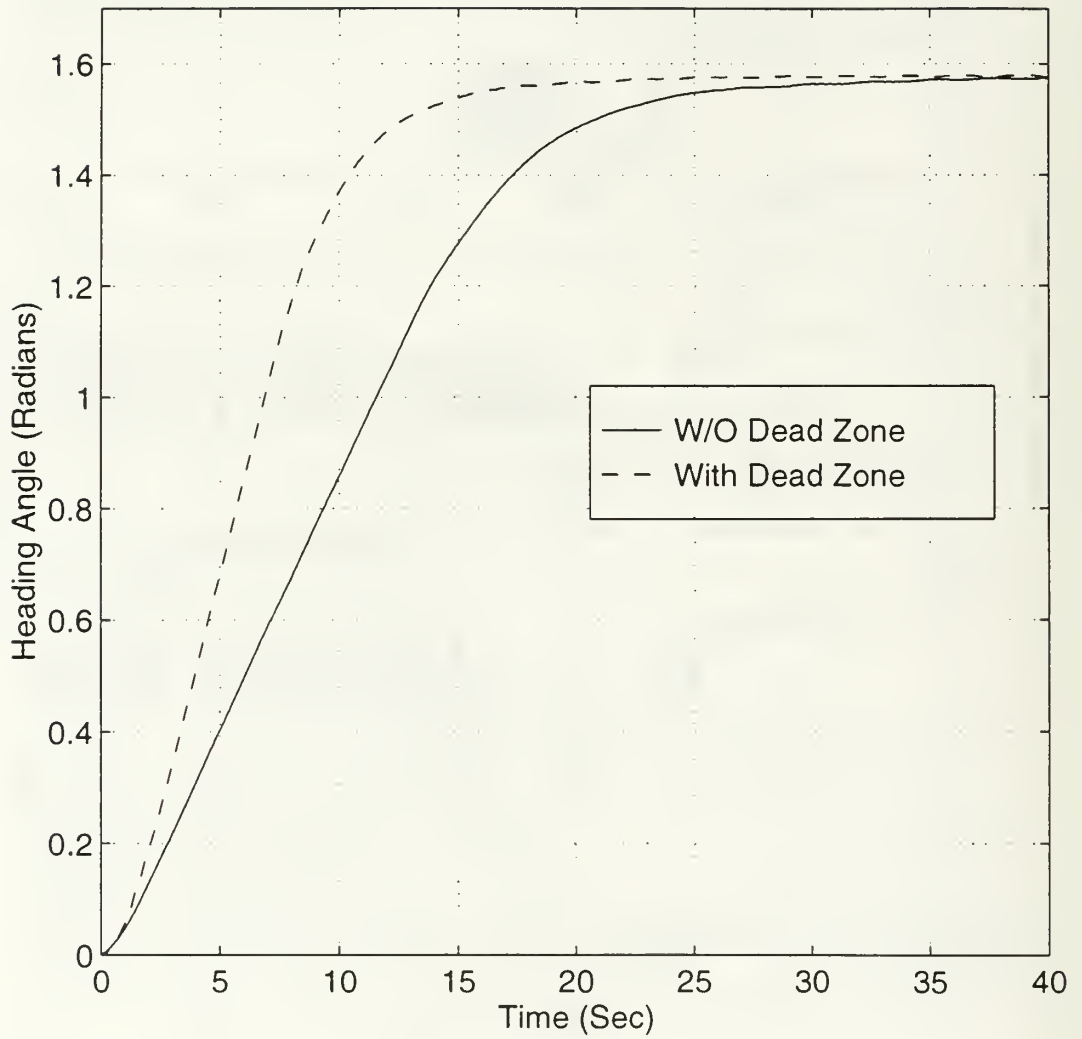
Figure 5.40 Heading Angle Step Response vs. Time With and Without Control
Voltage Dead Zone.

220
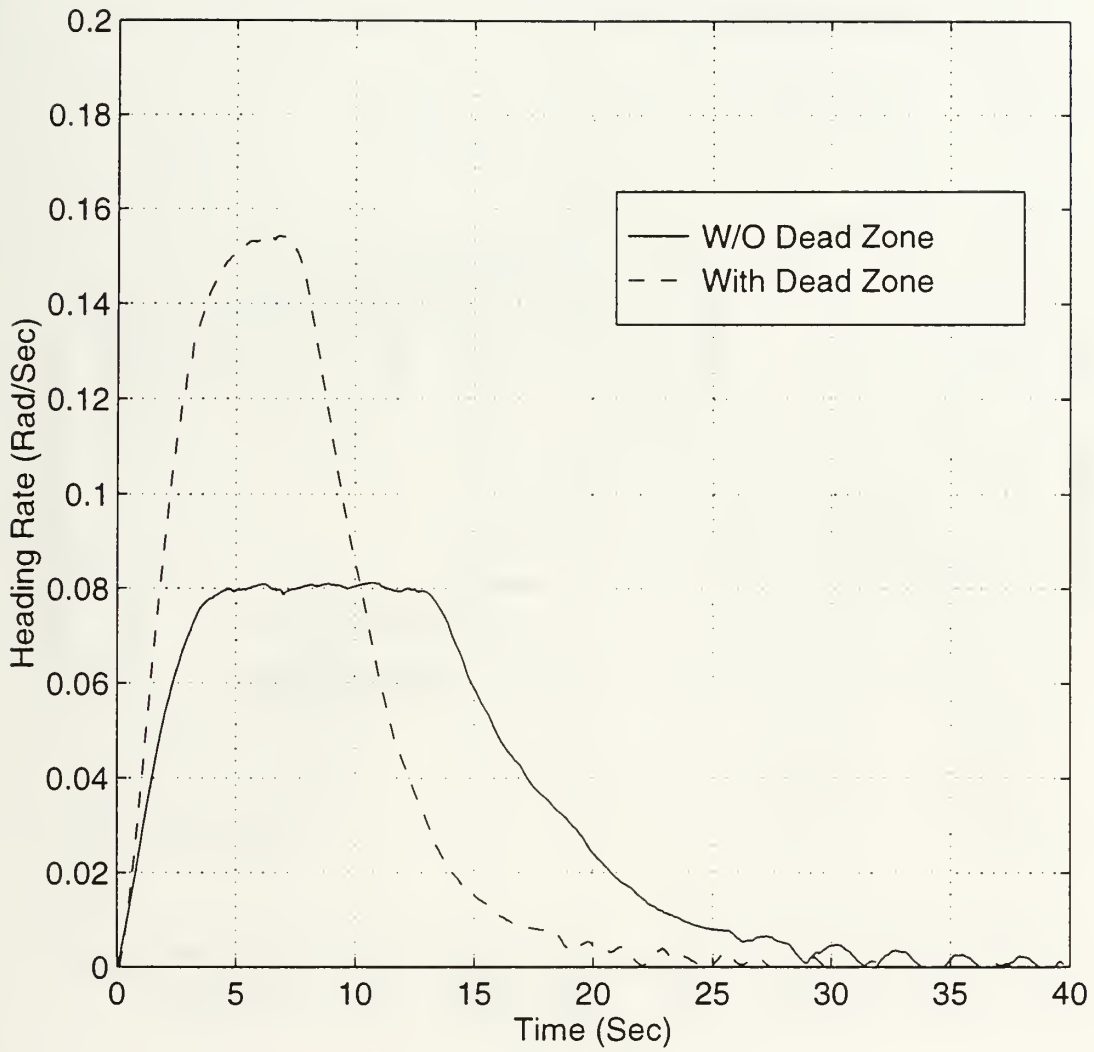
Figure 5.41 Heading Rate Response vs. Time With and Without Control
Voltage Dead Zone.

221
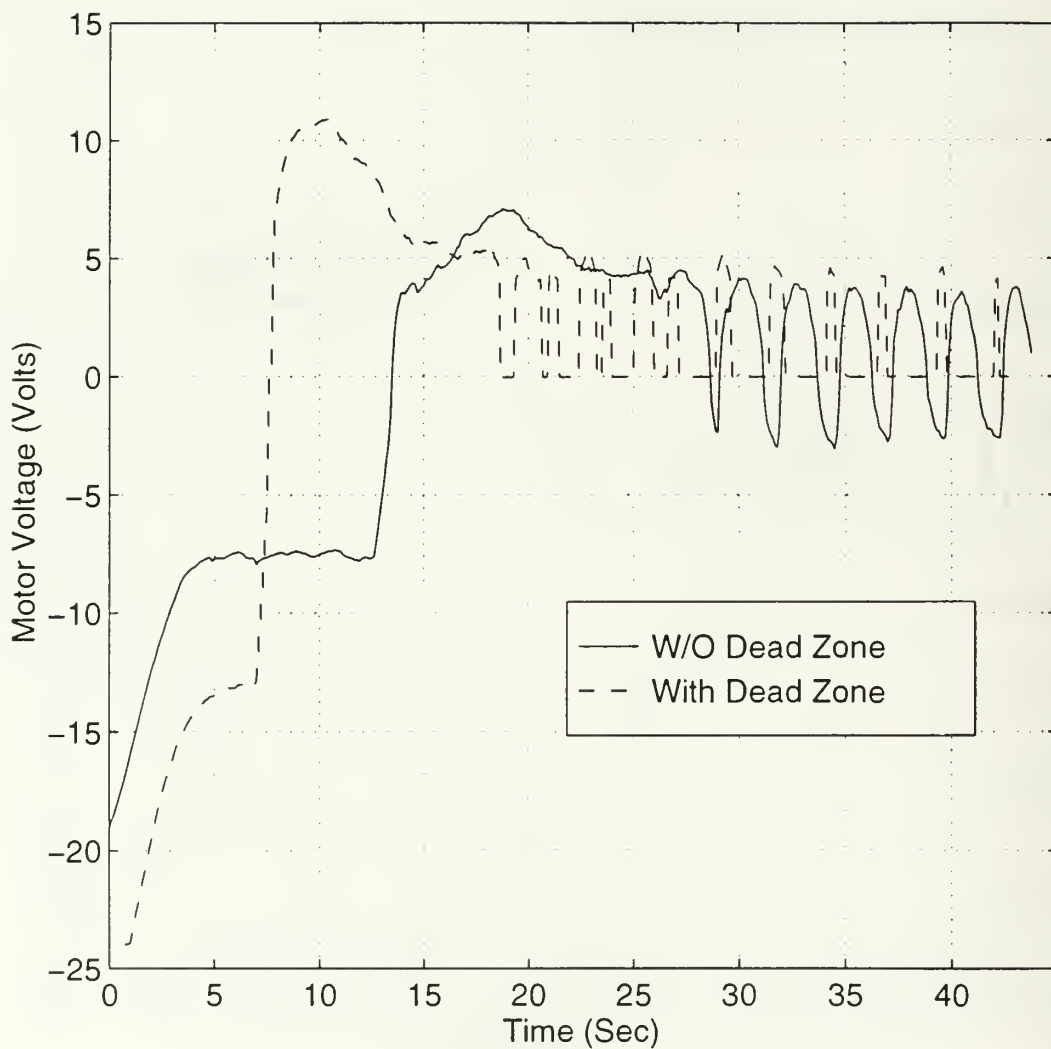
Figure 5.42 Lateral Thruster Control Voltage vs. Time With and Without Control Voltage Dead Zone.
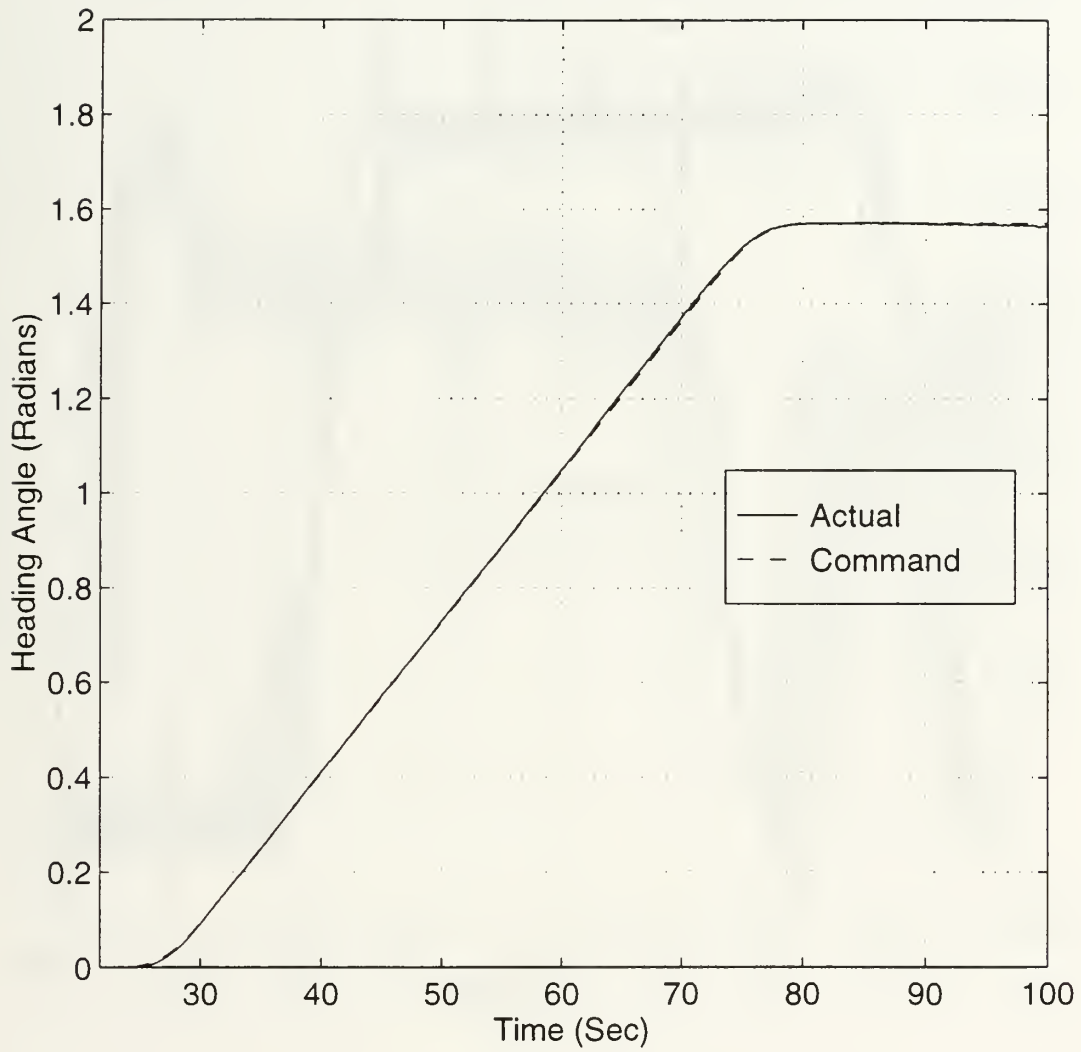
Figure 5.43 Heading Angle Response Using Command Generator Design 1.

223
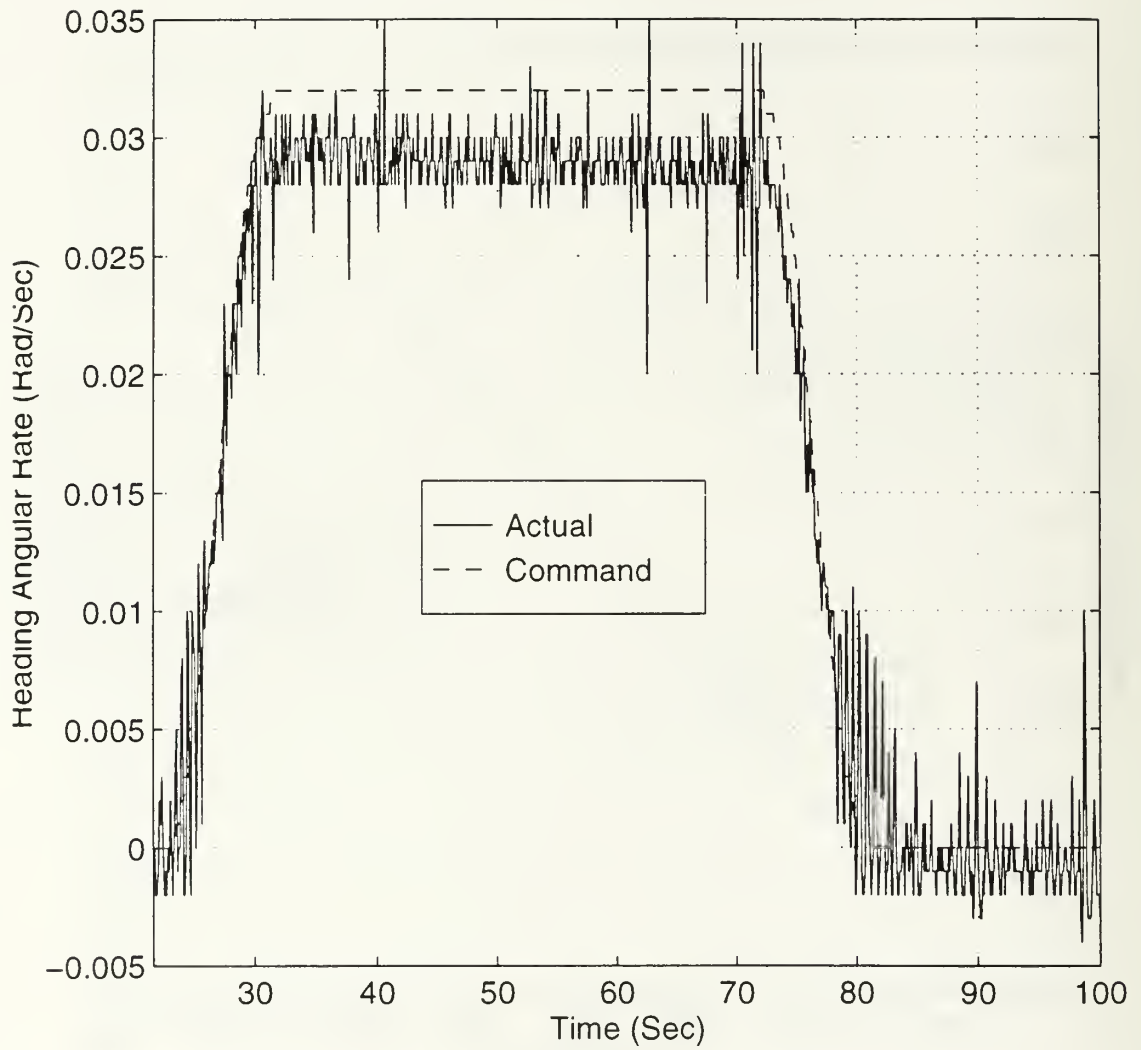
Figure 5.44 Heading Rate Response Using Command Generator Design 1.
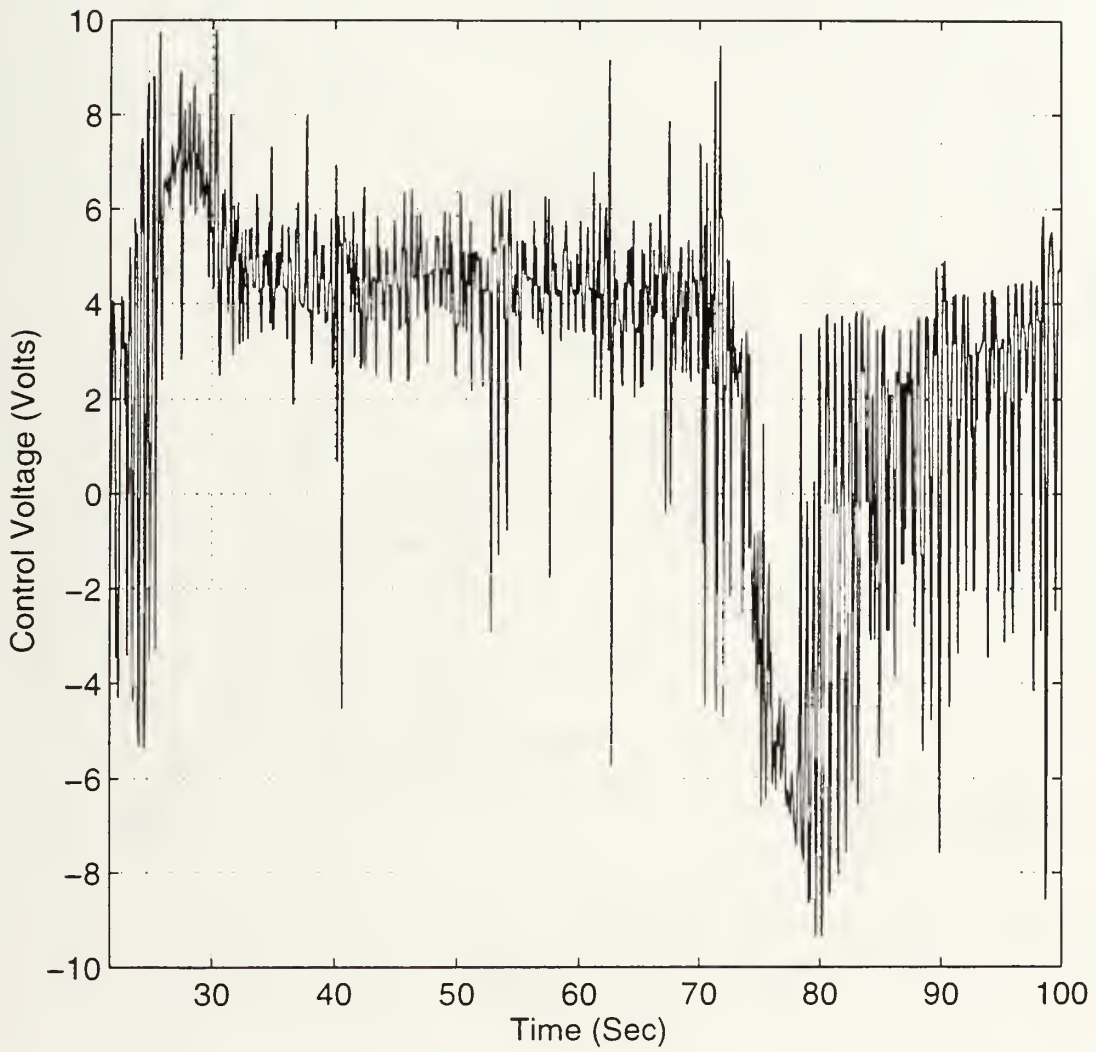
Figure 5.45 Lateral Thruster Control Voltage Using Command Generator Design 1.
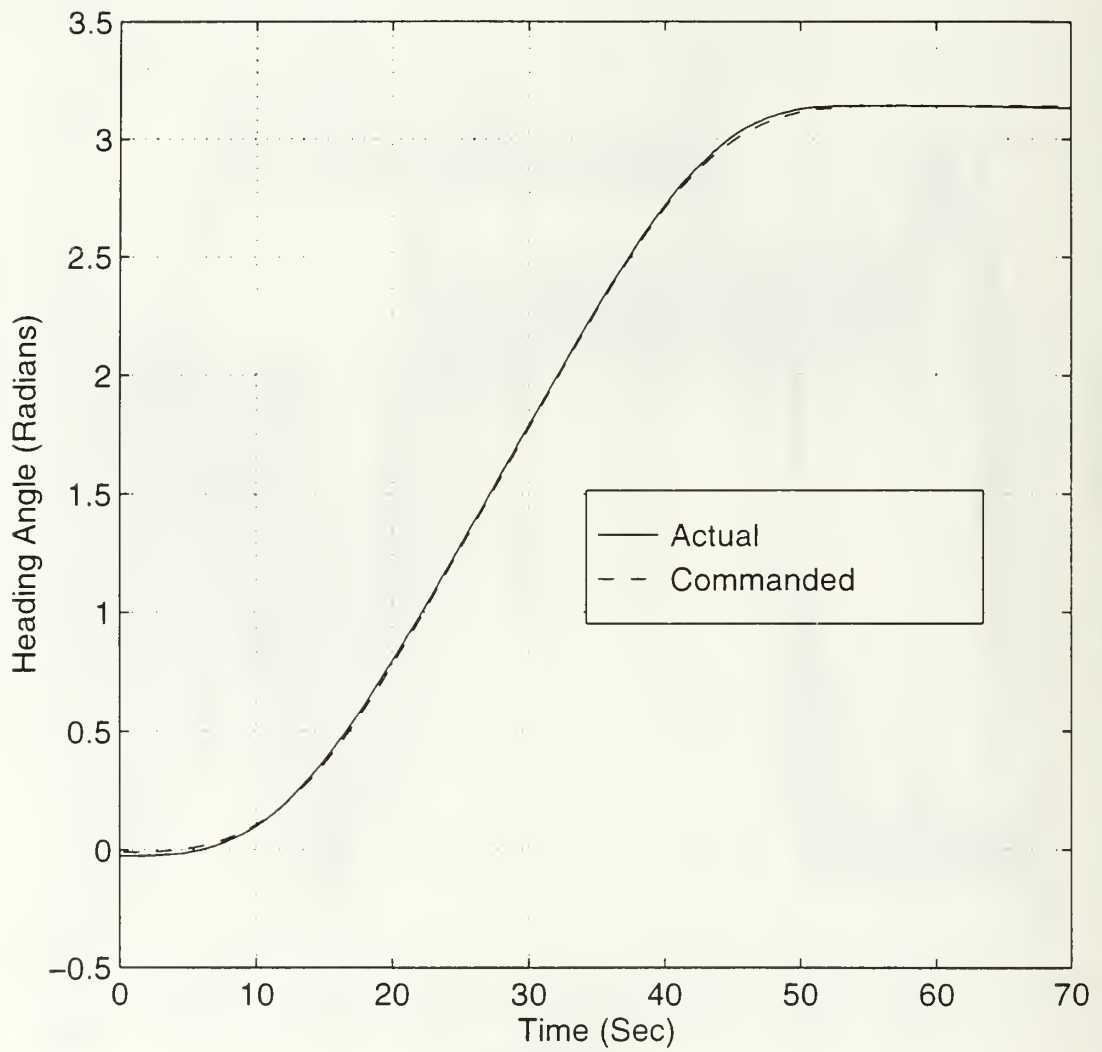
Figure 5.46 Heading Angle Response vs. Time Using Command Generator Design 2.
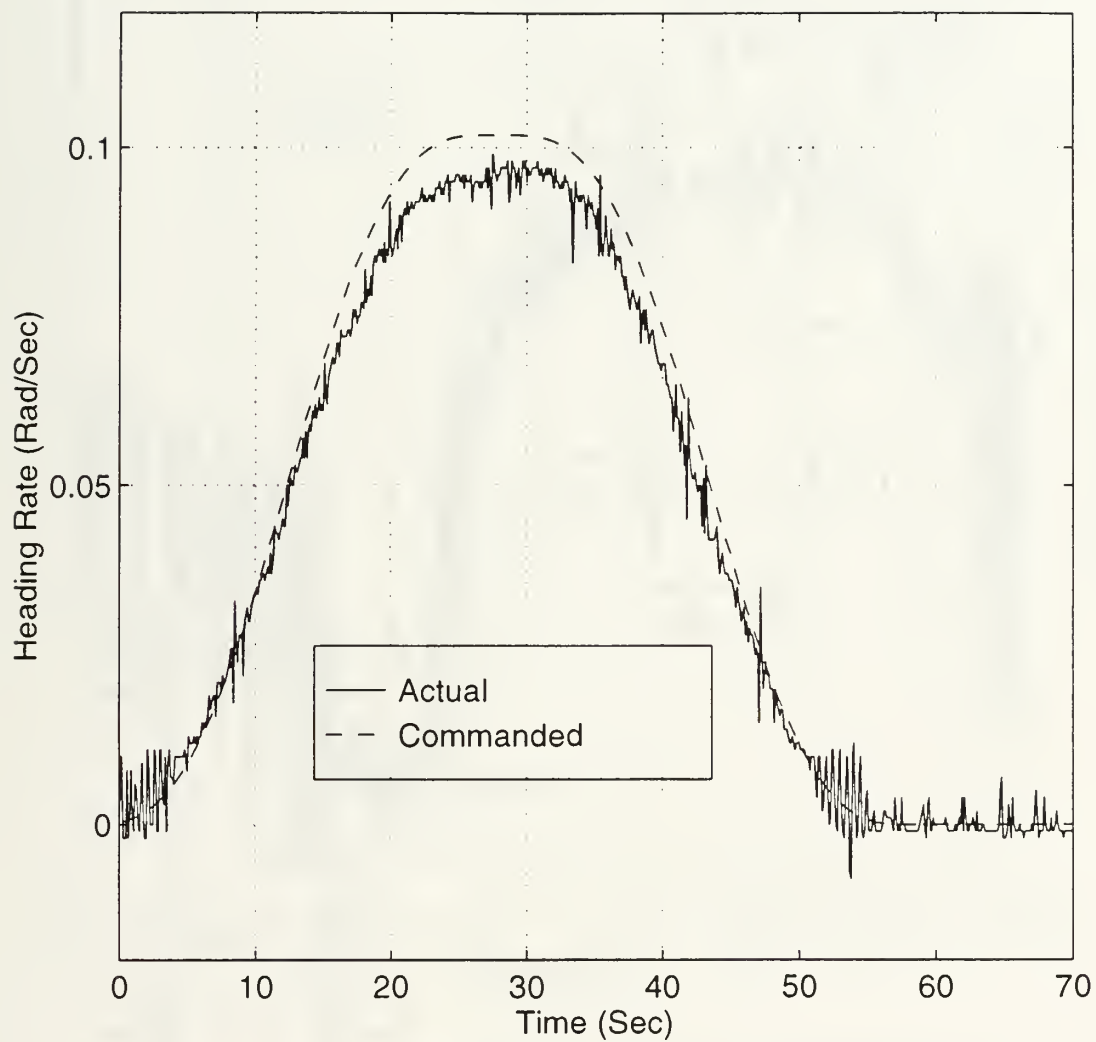
Figure 5.47 Heading Rate Response vs. Time Using Command Generator Design 2.
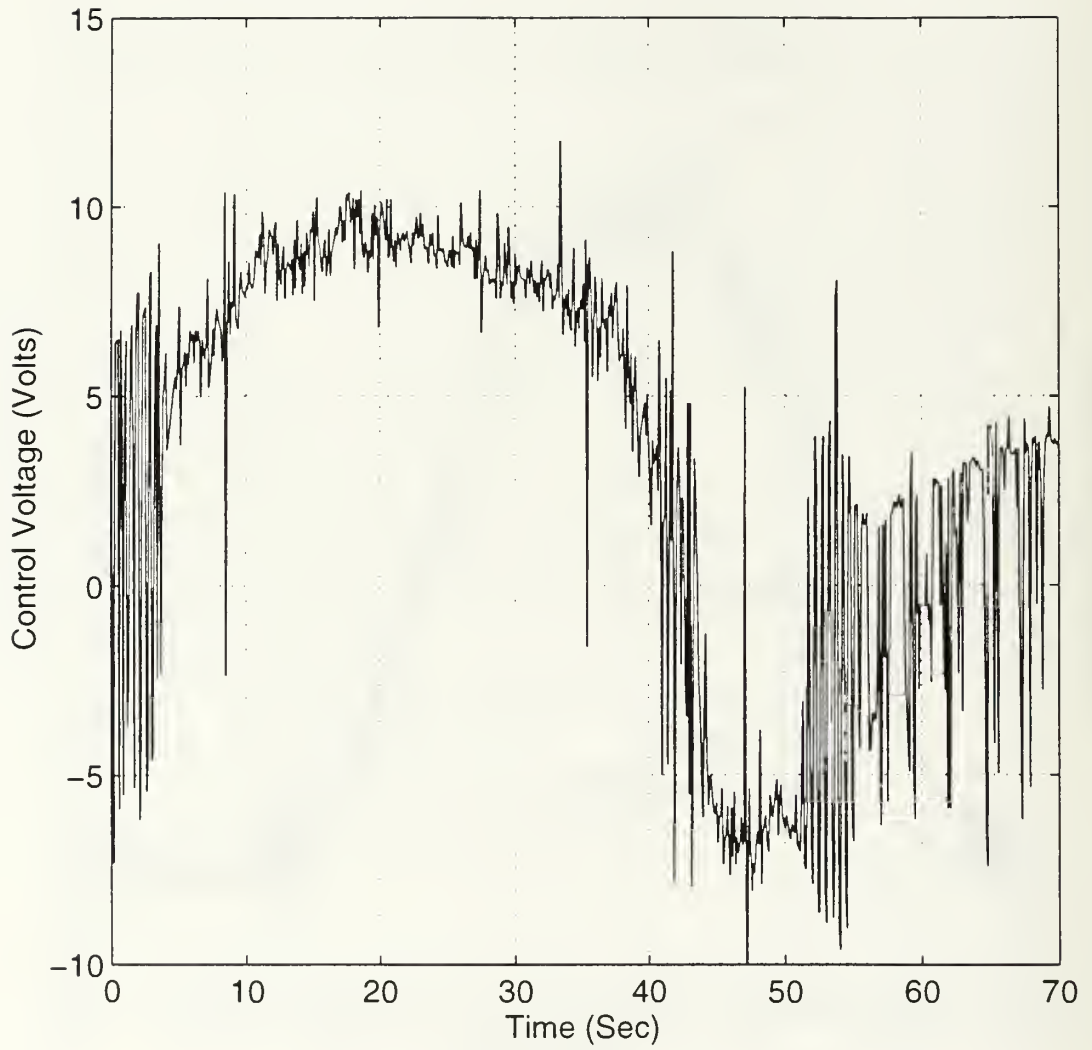
Figure 5.48 Lateral Thruster Control Voltage vs. Time Using Command
Generator Design 2.

228

WALL

R_{com}

$R(t)$

ST1000
SONAR
HEAD

SONAR
BEAM

R_0

x

y

POSITION AT
TIME t

$x(t)$

x

POSITION AT $t = 0$

y

$x = 0$

NOTE: NO LATERAL TRANSLATION
OCCURS. THE SECOND POSITION
IS OFFSET FOR CLARITY.
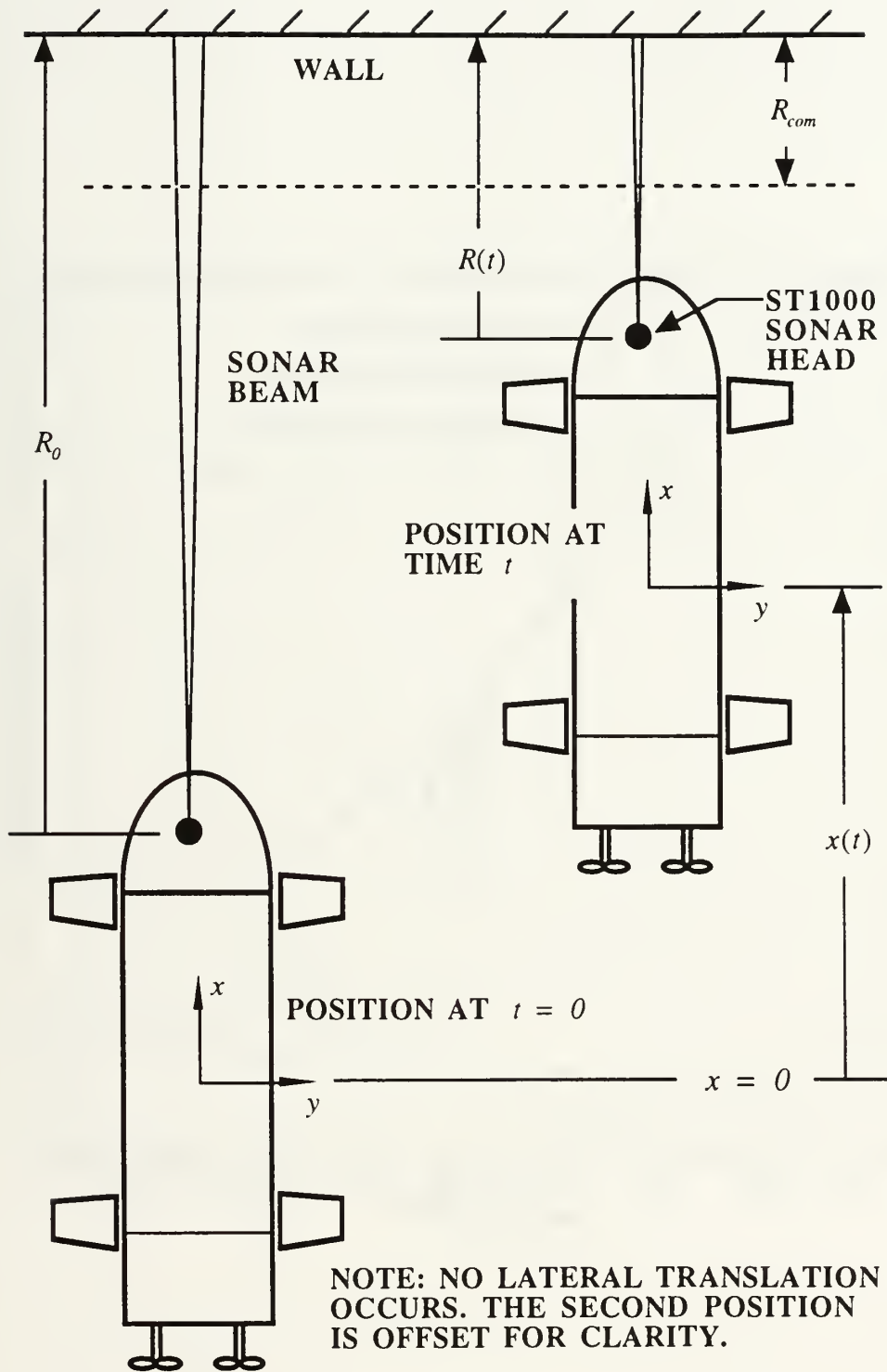
Figure 5.49 Wall Servoing Experimental Setup.
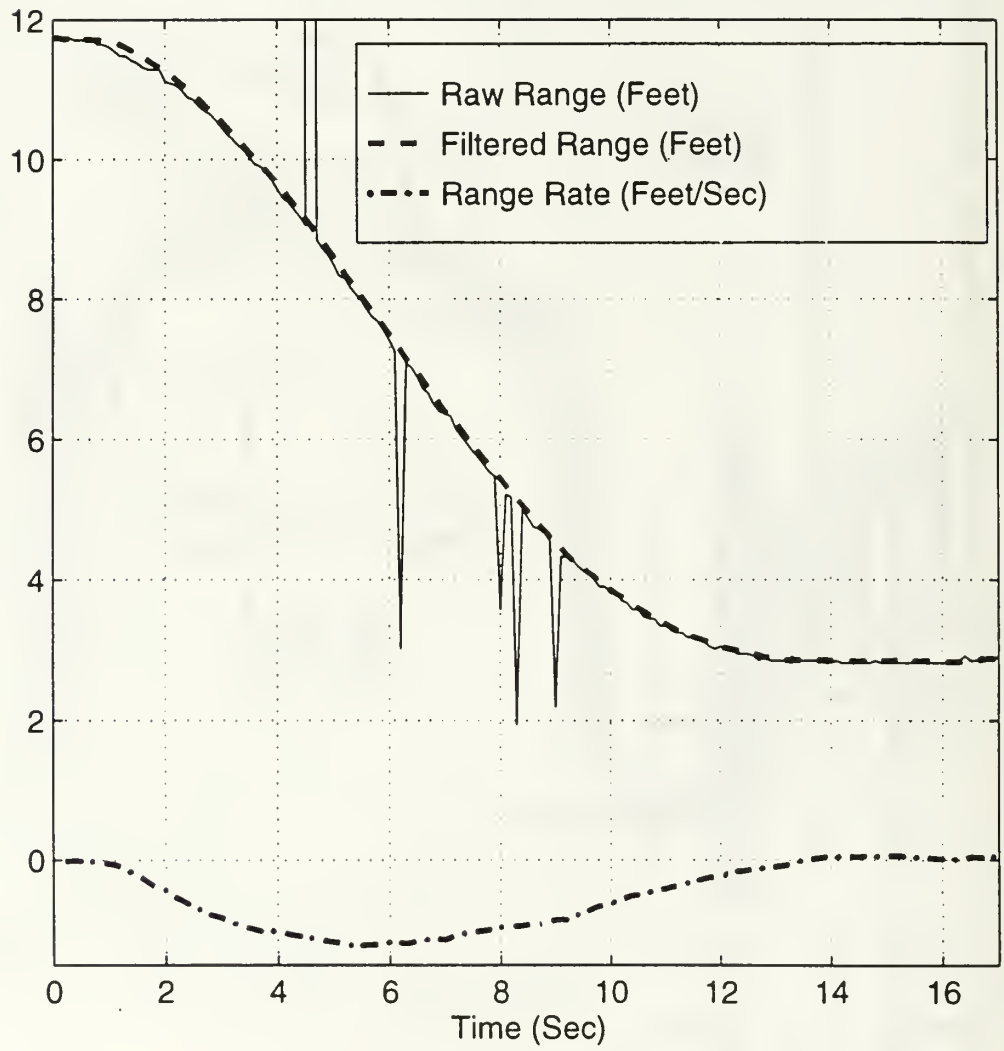
Figure 5.50 Filtered and Unfiltered Range.

230
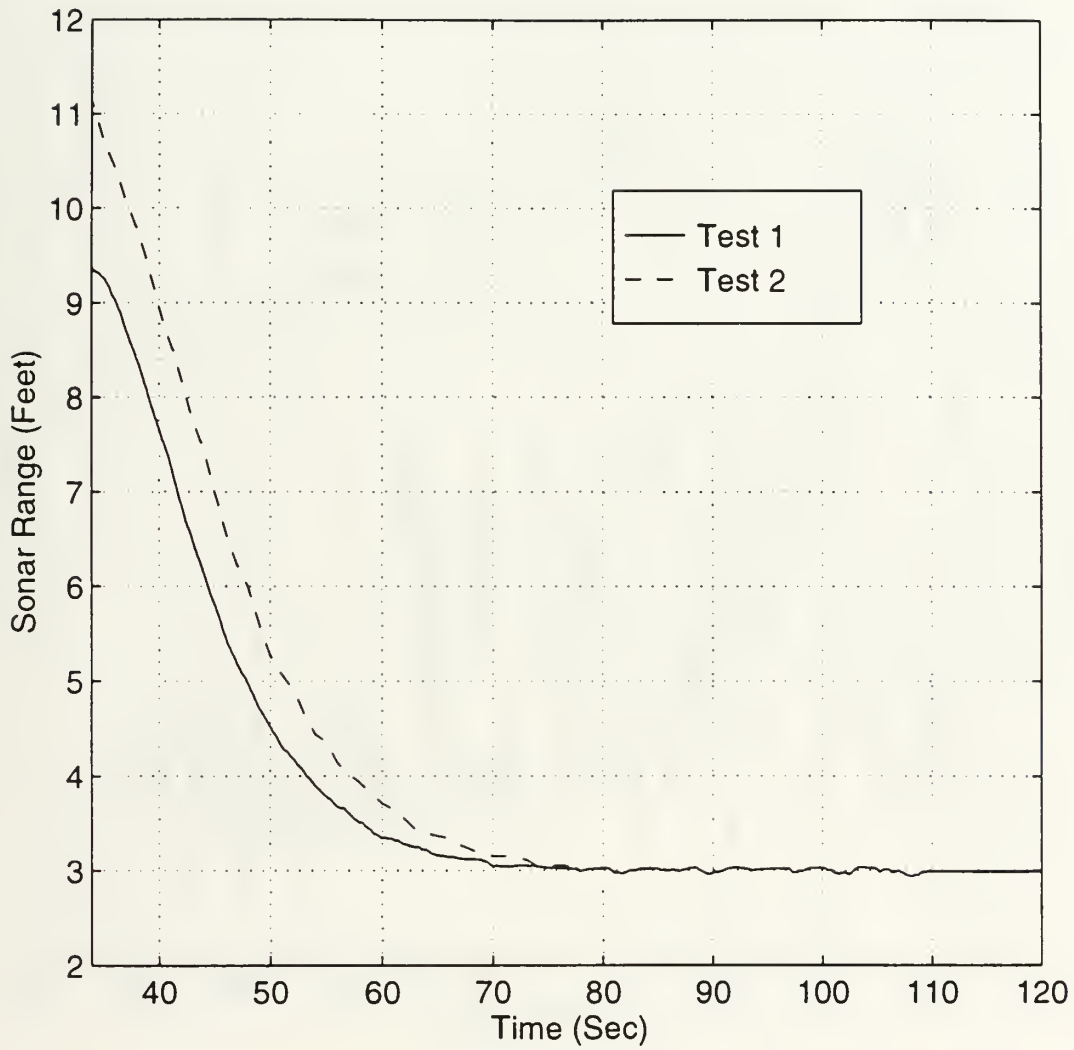
Figure 5.51 Position Response vs. Time for Tests 1 and 2.

231
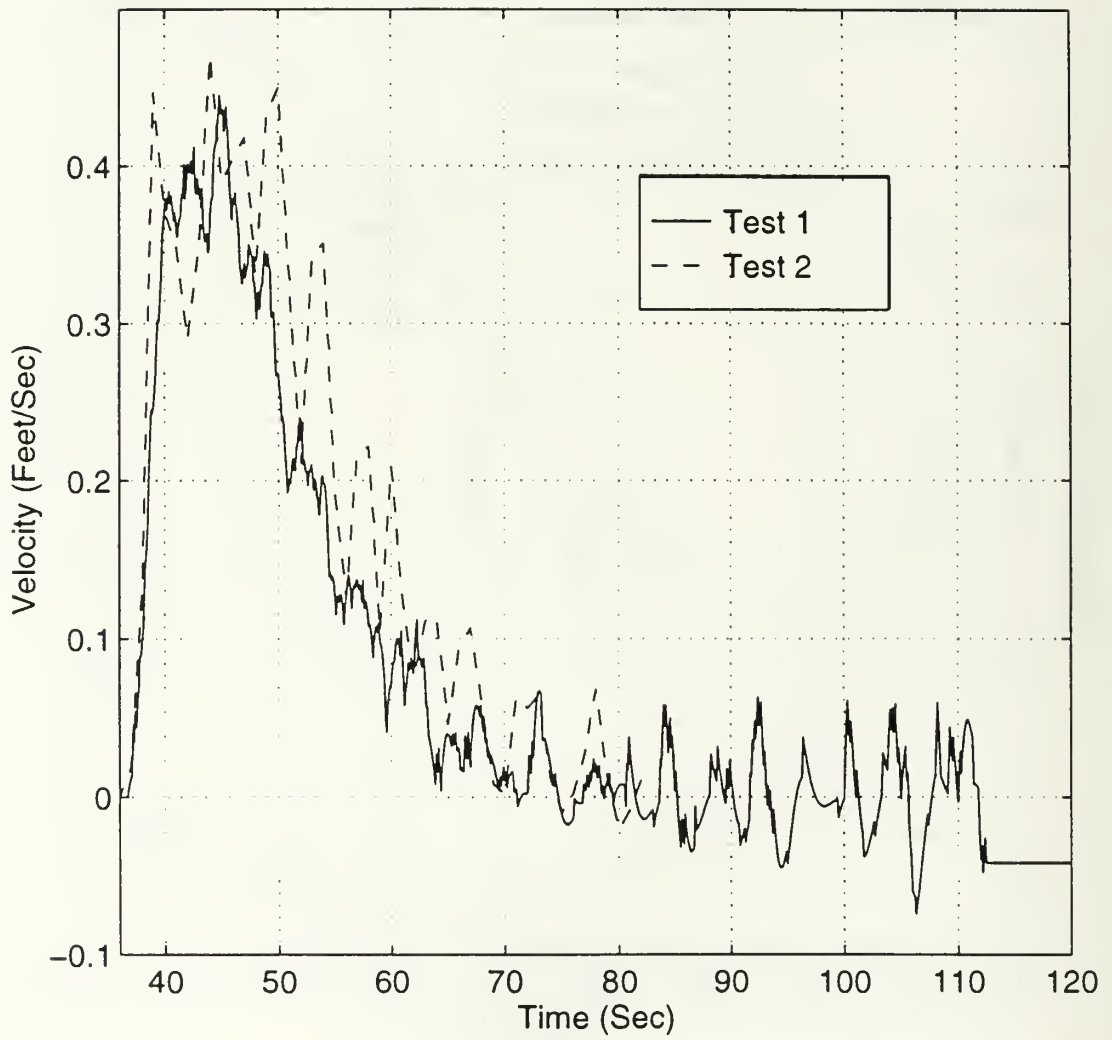
Figure 5.52 Position Rate Response vs. Time for Tests 1 and 2.

232
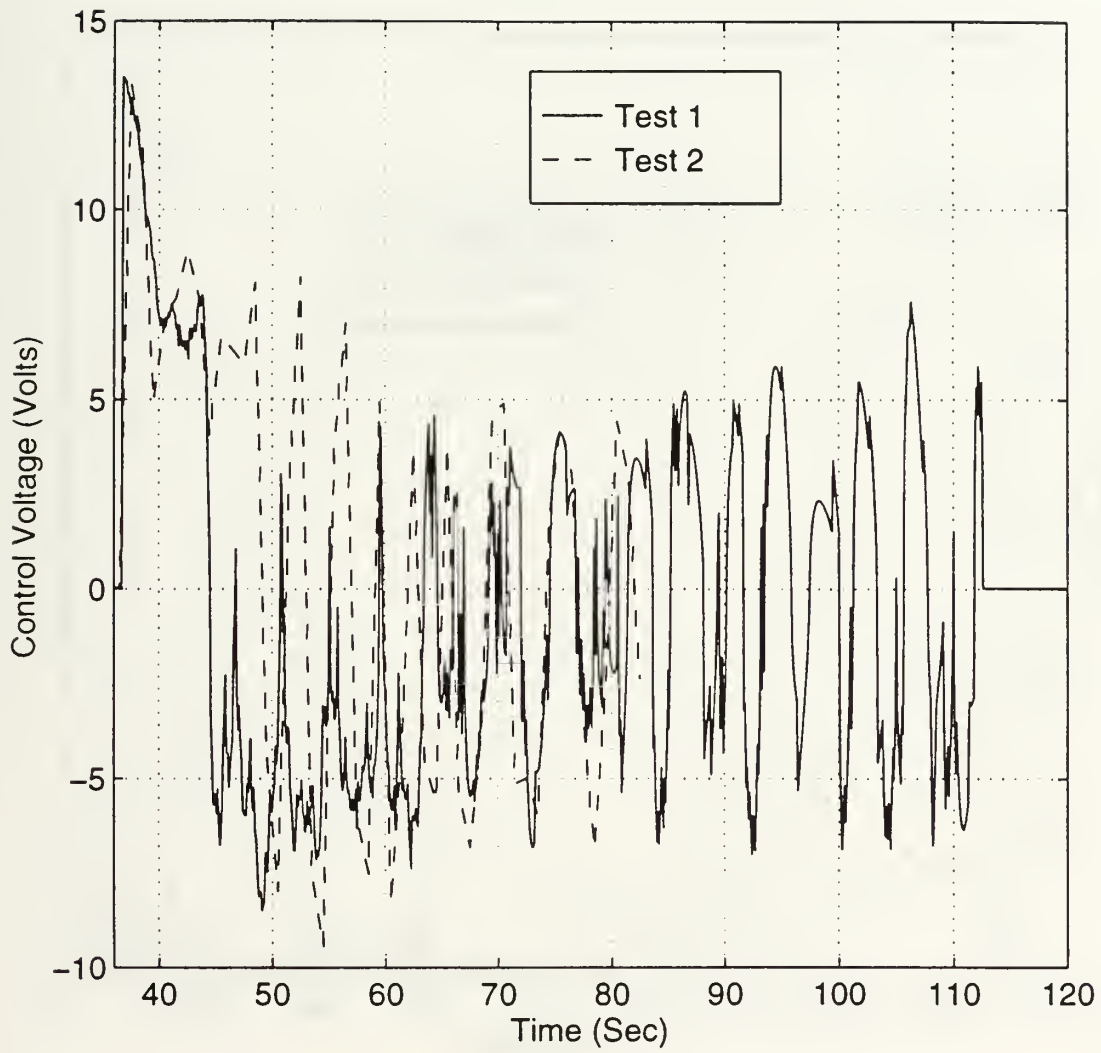
Figure 5.53 Stern Screw Control Voltage vs. Time for Tests 1 and 2.
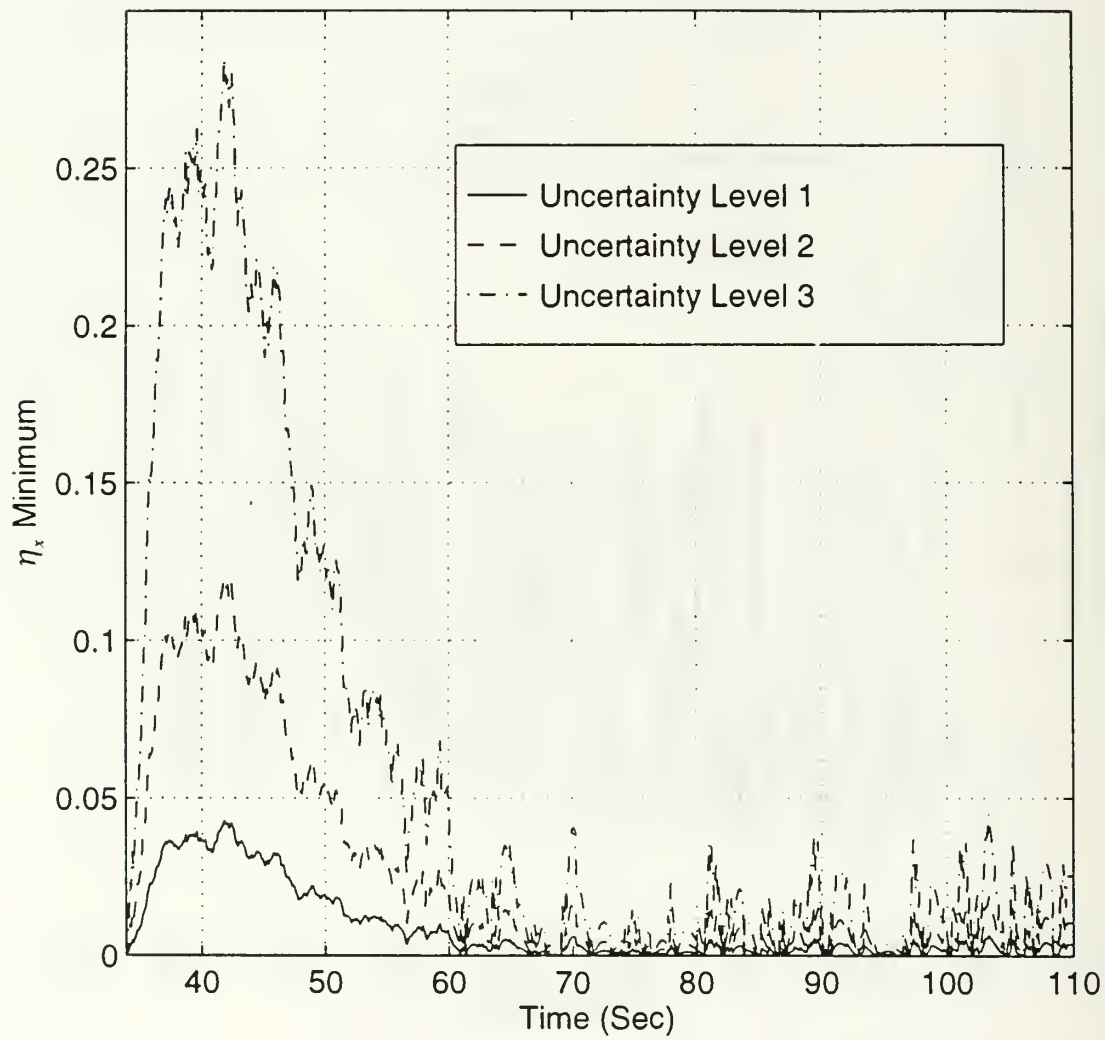
Figure 5.54 Effect of Parameter Uncertainty on Gain η_x.
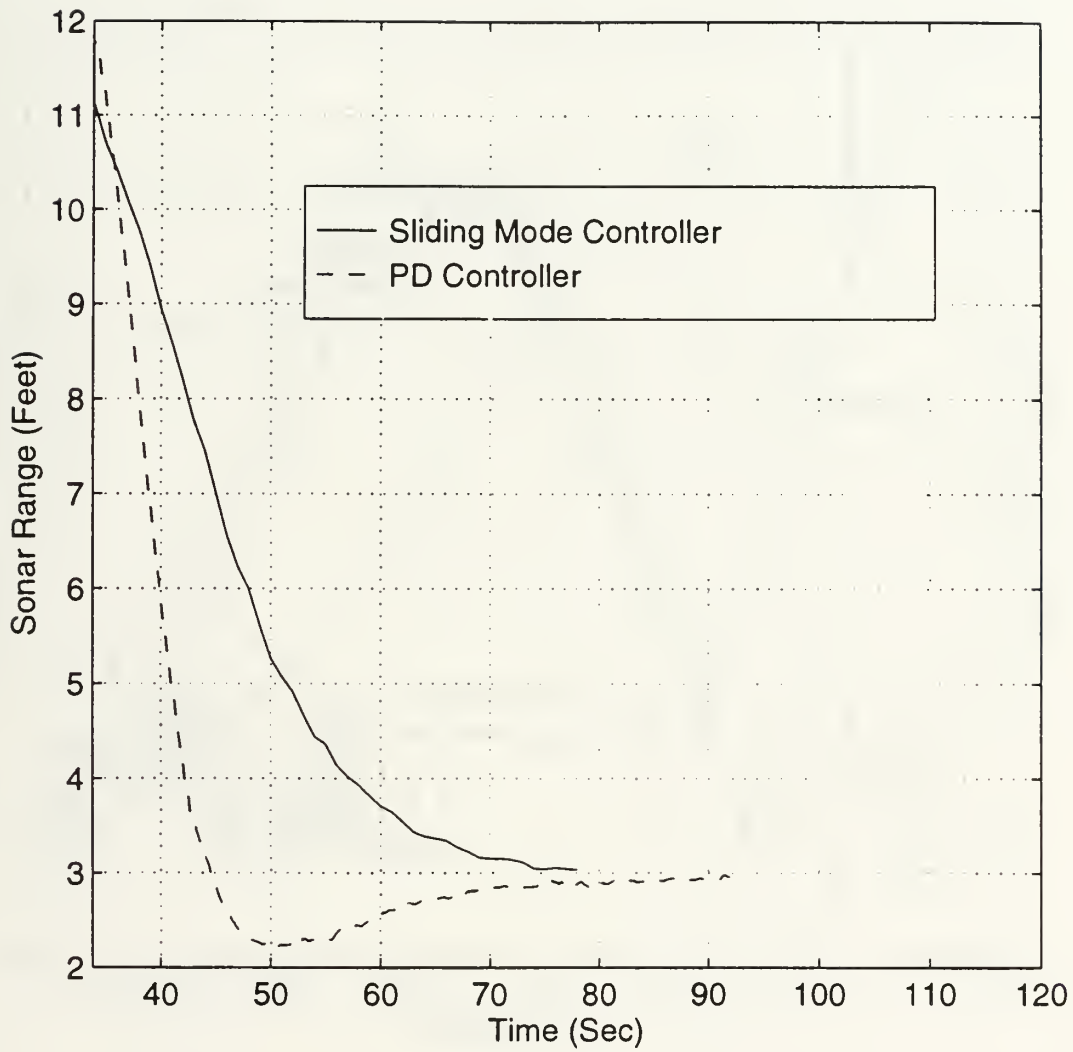
Figure 5.55 Sliding Mode Verses PD Controller Position Response.
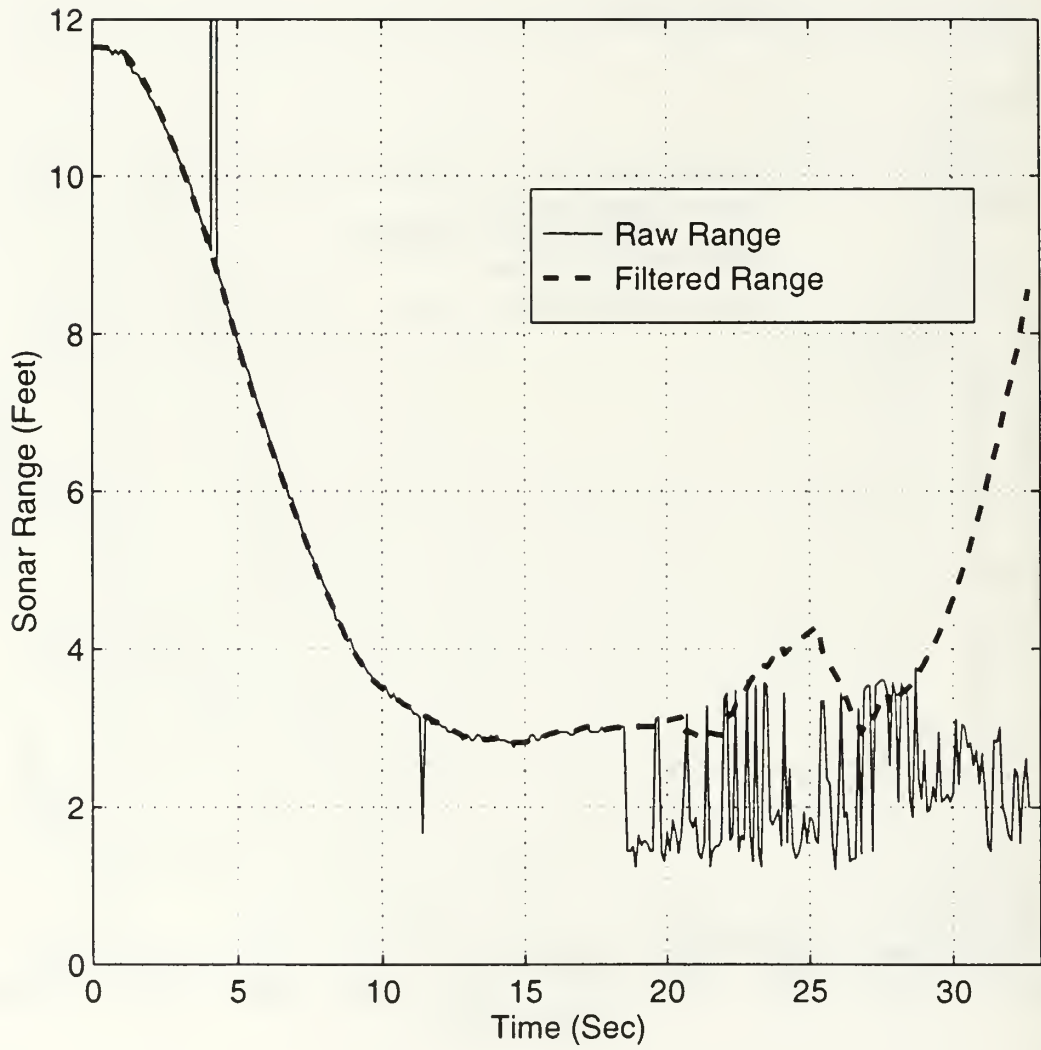
Figure 5.56 Sonar Ensonification in the NPS Test Tank.
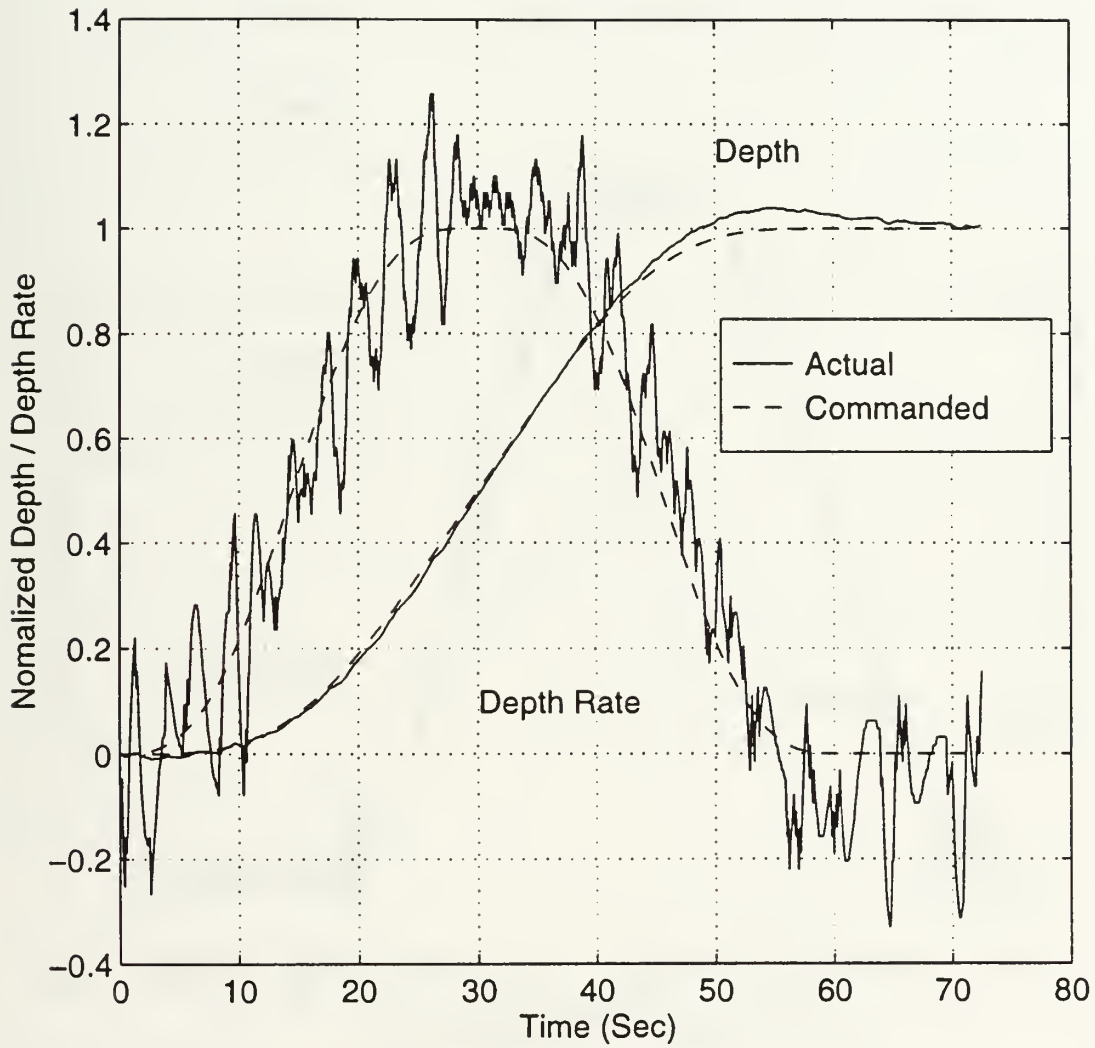
Figure 5.57 Normalized Depth and Depth Rate Response vs. Time of the
Coordinated Maneuver.
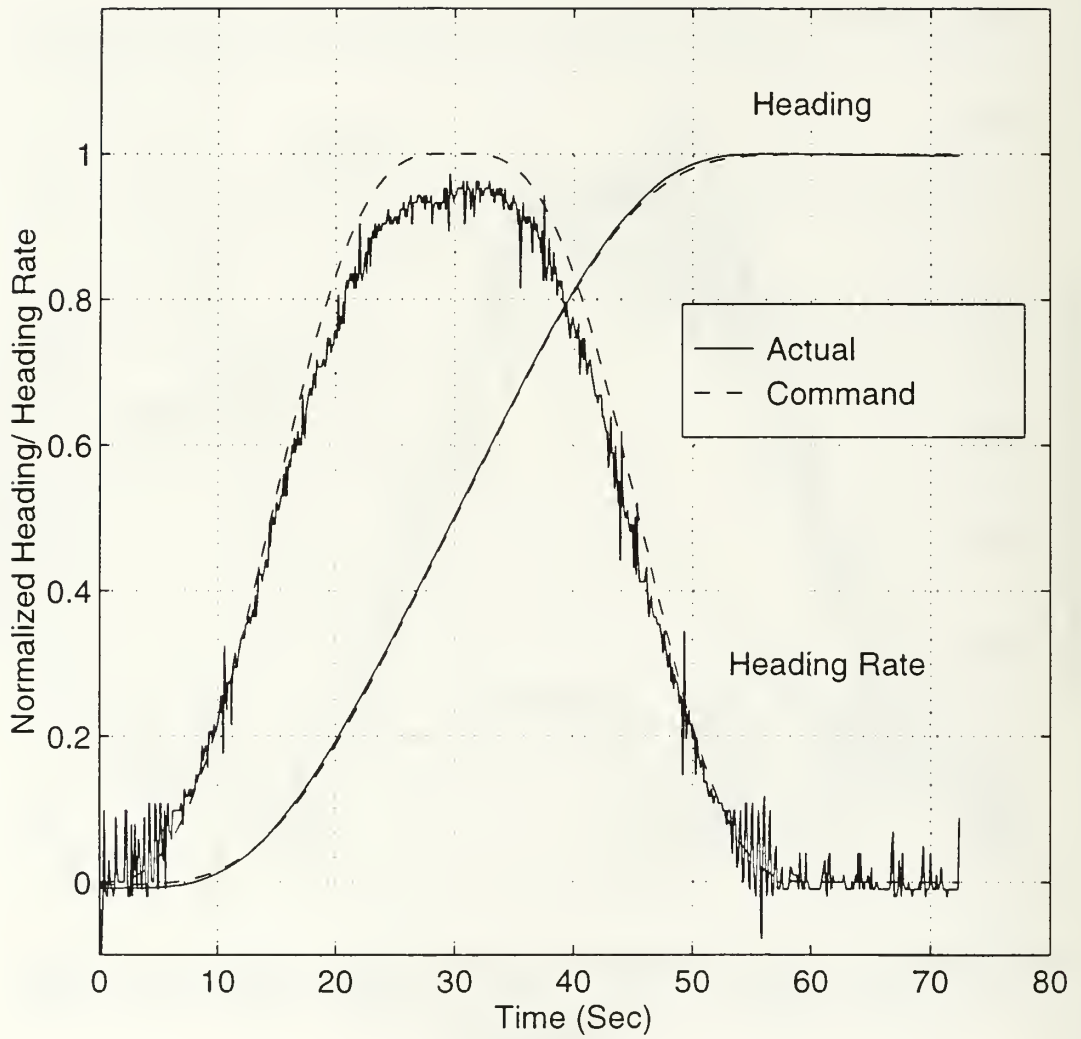
Figure 5.58 Normalized Heading and Heading Rate Response vs. Time of the Coordinated Maneuver.
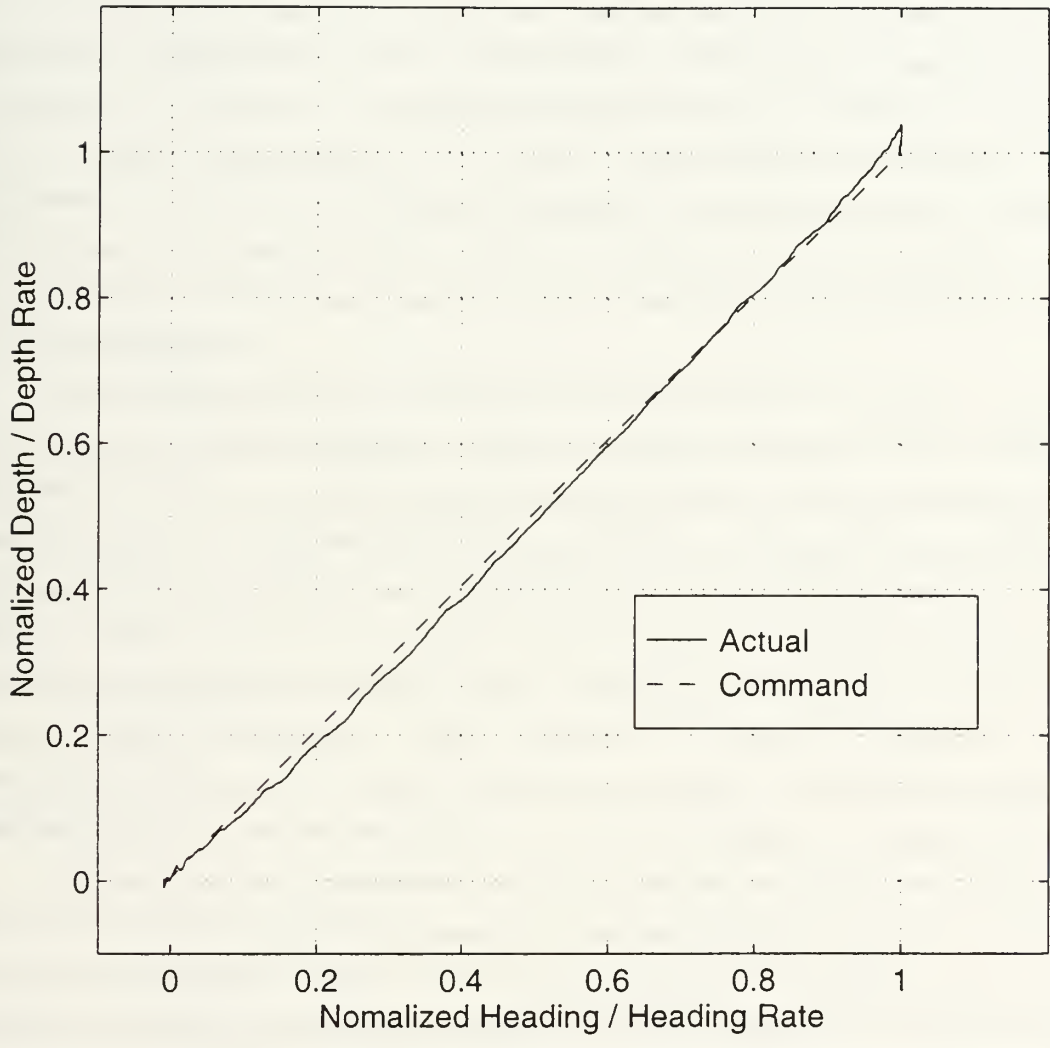
Figure 5.59 Normalized Depth vs. Heading Response of the Coordinated Maneuver.
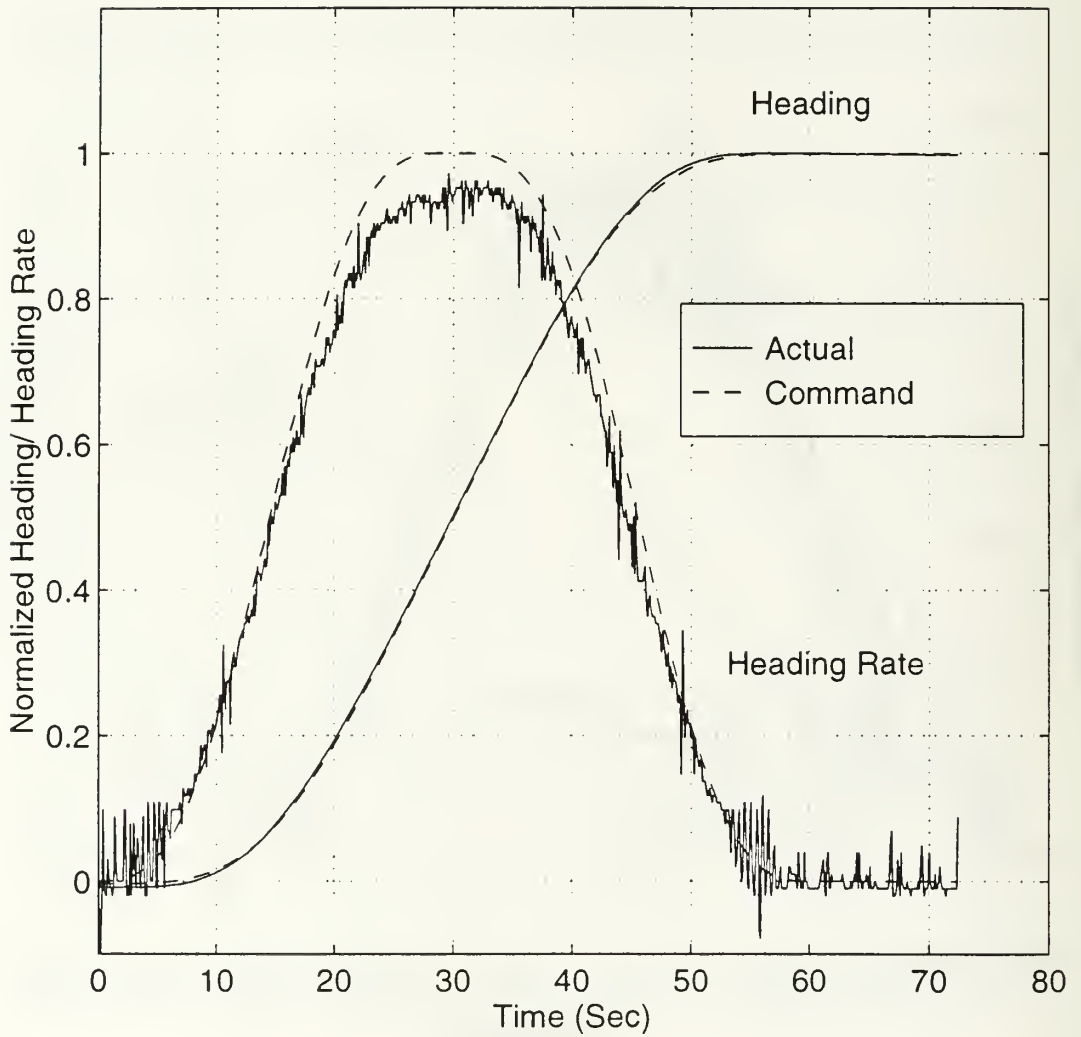
Figure 5.58 Normalized Heading and Heading Rate Response vs. Time of the Coordinated Maneuver.
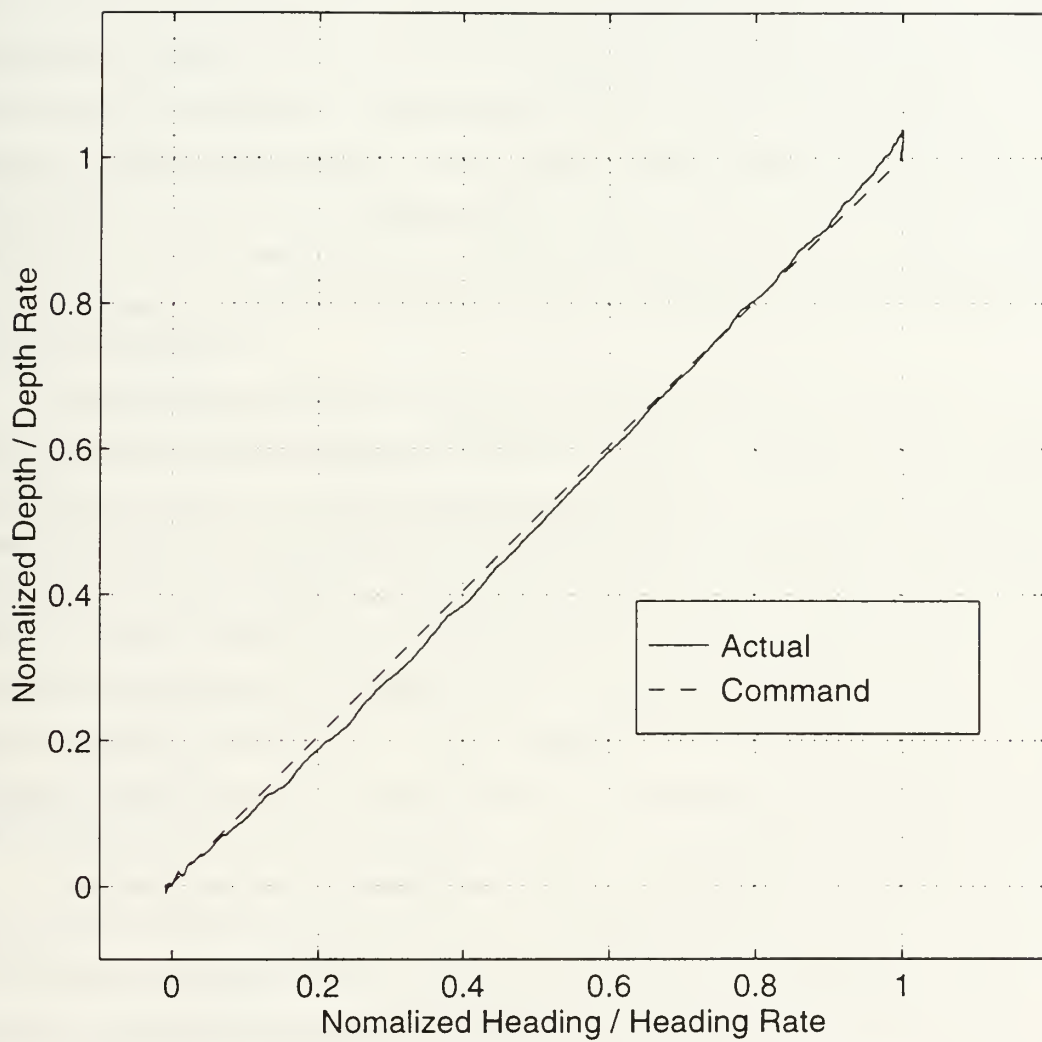
Figure 5.59 Normalized Depth vs. Heading Response of the Coordinated Maneuver.

VI. LOCAL AREA NAVIGATION

While there has always been a need to determine the global position of an underwater vehicle, in some missions involving search, mapping, and intervention with objects, navigation to local area landmarks is more appropriate and precise. In minefield reconnaissance operations, there is also a need to determine position relative to a target for the purpose of taking video pictures, viewing from different aspects, and even charge placement. If the object lies in an unstructured environment, the vehicle must use active sensors to perform these operations. Once maneuvering control around objects in the local area scene is understood to a satisfactory degree, intervention using manipulators and other tactile devices will be enabled. Such activities as changing out a battery pack for a bottom mounted sensor or finding and entering an underwater garage for re-powering will then become commonplace.

This chapter concerns the analysis of local area maneuvering using sonar based feature detection from the local scene. As this process takes a significant time to complete, a mathematical model of the vehicle response is used to provide control inputs during periods when sonar updates are not available. In the class of vehicles designed for the intervention mission, (Marks, et. al., 1994) have studied the problem of servo positioning the OTTER vehicle to visual cues from stereoscopic cameras, although monocular video data was used to perform edge detection and servo control of the pan and tilt mounting coupled to the vehicle platform. Smith, et al. have proposed the use of an acoustic single-transmitter/multiple receiver design for local area navigation, although preliminary data from the sonars alone seem encouraging, it has yet to implemented in an actual vehicle.

Section A. covers the formulation of the three degree-of-freedom equations of motion for the vehicle (longitudinal, lateral, and heading), which will be used as part of the a model based navigation control. The next section address the algorithms used to locate and track a navigation target with the ST1000 sonar, and is followed by a description of the control methodology which incorporates model based control with position updates from the sonar. Simulation and experimental results of the control performance are given, and the chapter concludes with a discussion of an improved target tracking technique.

A. MODEL BASED CONTROL FORMULATION

Absent of an inertial position reference, and where sonar position data arrive asynchronously, and infrequent, a dynamic model of the vehicle is used for state information between updates. A three degree-of-freedom model (longitudinal, lateral, and heading) is used since the motion for this experiment is restricted to the horizontal plane with the depth maintained by a separate controller. The model is obtained by including drag, added mass, and steady state thrust, and for surge is

$$M_x \dot{u}(t) + b_x u(t)|u(t)| = 2\alpha_x v_x(t)|v_x(t)| \tag{6.1}$$

The sway directional equation of motion is

$$M_y \dot{v}(t) + b_y v(t)|v(t)| = \alpha_y v_{blt}(t)|v_{blt}(t)| + \alpha_y v_{slt}(t)|v_{slt}(t)| \tag{6.2}$$

and finally the equation for the yaw dynamics becomes

$$I_z \dot{r}(t)(t) + b_\psi r(t)|r(t)| = \alpha_\psi v_{blt}(t)|v_{blt}(t)| - \alpha_\psi v_{slt}(t)|v_{slt}(t)| \tag{6.3}$$

where

$$M_x = m + m_{ax}, \qquad M_y = m + m_{ay}, \quad I_z = I_{zz} + I_{azz}$$

and

$$v_x(t)|v_x(t)| = (v_{ls}(t)|v_{ls}(t)| + v_{rs}(t)|v_{rs}(t)|)/2. \tag{6.4}$$

m is the vehicle mass, I_{zz}, the mass moment of inertia about the body-fixed z-axis, and the subscript "a" refers to the added mass or inertia of the body. $u(t)$, $v(t)$, and $r(t)$ are the body-fixed relative velocities for longitudinal (x-axis), lateral (y-axis), and heading (ψ) directions. b_x, b_y, b_ψ are the square-law damping coefficients, $v_{ls}(t)$, $v_{rs}(t)$, and $v_{blt}(t)$, $v_{slt}(t)$ are the thruster motor input voltages for the left/right stern screws, and the bow/stern lateral thrusters respectively. The voltage to force/moment coefficients are given by α_x, α_y, and α_ψ.

242

The above dynamic equations can be formulated using matrix notation as

$$M\dot{x}(t) = f(x(t),b) + g(\alpha)u(t)$$ (6.5)

and vehicle kinematics are defined by

$$\dot{z}(t) = h(\psi)x(t) + u_c(t).$$ (6.6)

The body-fixed relative velocities are

$$x(t) = \{\, u(t) \ v(t) \ r(t)\,\}^T,$$ (6.7)

and the global position and orientation is given by

$$z(t) = \{\, X(t) \ Y(t) \ \psi(t)\,\}^T.$$ (6.8)

The vector describing the hydrodynamic drag that is a function of the relative velocity and square-law damping coefficients, $b = \{\, b_x \ b_y \ b_\psi\,\}$ is

$$f(x(t),b) = \{-b_x u(t)|u(t)| \ -b_y v(t)|v(t)| \ -b_\psi r(t)|r(t)|\}^T,$$ (6.9)

the mass matrix is

$$M = \begin{bmatrix} M_x & 0 & 0 \\ 0 & M_y & 0 \\ 0 & 0 & I_z \end{bmatrix},$$ (6.10)

and input gain matrix, which is solely a function of the thruster coefficients, $\alpha = \{\, \alpha_x \;\; \alpha_y \;\; \alpha_\psi \,\}$, is

$$g(\alpha) = \begin{bmatrix} 2\alpha_x & 0 & 0 \\ 0 & \alpha_y & \alpha_y \\ 0 & \alpha_\psi & -\alpha_\psi \end{bmatrix}. \tag{6.11}$$

Finally, the control input vector is defined as

$$u(t) = \left\{ v_x(t)|v_x(t)| \quad v_{blt}(t)|v_{blt}(t)| \quad v_{slt}(t)|v_{slt}(t)| \right\}^T. \tag{6.12}$$

For the case of translation in X, Y and rotation ψ, the transformation matrix from the body-fixed axes to the global reference is given by

$$h(\psi(t)) = \begin{bmatrix} cos(\psi(t)) & -sin(\psi(t)) & 0 \\ sin(\psi(t)) & cos(\psi(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.13}$$

and it's time derivative is

$$\dot{h}(\psi(t), \dot{\psi}(t)) = \begin{bmatrix} -\dot{\psi}(t)sin(\psi(t)) & -\dot{\psi}(t)cos(\psi(t)) & 0 \\ \dot{\psi}(t)cos(\psi(t)) & -\dot{\psi}(t)sin(\psi(t)) & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{6.14}$$

Any current disturbances are represented by

$$u_c(t) = \{\, u_{c_x}(t) \;\; u_{c_y}(t) \;\; 0 \,\}^T \tag{6.15}$$

244

The sliding mode control law can now be formulated defining the tracking error vector in terms of global coordinates as

$$
\begin{bmatrix} \dot{\tilde{z}}(t) \\ \tilde{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{z}_{com}(t) \\ z_{com}(t) \end{bmatrix} - \begin{bmatrix} \dot{z}(t) \\ z(t) \end{bmatrix}.
\tag{6.16}
$$

Now that the tracking error has been formulated, an equation defining the sliding surface in terms of this error can be written as

$$
\sigma(\tilde{z}(t)) = \begin{bmatrix} S_1 & S_2 \end{bmatrix} \begin{bmatrix} \dot{\tilde{z}}(t) \\ \tilde{z}(t) \end{bmatrix}.
\tag{6.17}
$$

where

$$
\sigma(\tilde{x}(t), \tilde{z}(t)) \in \Re^{3 \times 1}; \quad S_1, S_2 \in \Re^{3 \times 3}
$$

The elements of S_1 and S_2 can be selected to provide the desired performance of the closed loop system. For the case of planar control, and choosing S_1 as the identity, they are

$$
S_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad S_2 = \begin{bmatrix} \lambda_x & 0 & 0 \\ 0 & \lambda_y & 0 \\ 0 & 0 & \lambda_\psi \end{bmatrix}.
$$

If S_1 is identity, and $u_c(t)$ is assumed zero, the sliding mode control becomes

$$
\begin{aligned}
u_1(t) &= \hat{g}(\bullet)^{-1}(\hat{M}h^{-1}\ddot{z}_{com}(t) - \hat{f}(\bullet)) \\
u_2(t) &= \hat{g}(\bullet)^{-1}\hat{M}\hat{h}^{-1}(\bullet)\left(-\dot{\hat{h}}(\bullet)\hat{h}(\bullet)^{-1}\dot{z} + S_2\dot{\tilde{z}}(t) \right) \\
u_3(t) &= \hat{g}(\bullet)^{-1}\hat{M}\hat{h}^{-1}(\bullet)\,\eta(\sigma(\bullet),\phi)
\end{aligned}
\tag{6.18}
$$

245

where

$$\phi = \left\{ \phi_x \ \phi_y \ \phi_\psi \right\}^T \quad \text{and} \quad \sigma = \left\{ \sigma_x \ \sigma_y \ \sigma_\psi \right\}^T,$$

In Eqn. 6.18, the switching term $\eta(\sigma(\bullet), \phi)$ is more clearly defined as a 3 by 1 column vector where each element contains the appropriate individual response mode switching function, so that

$$\eta(\sigma(\bullet), \phi) = \begin{bmatrix} \eta_x sat(\sigma_x / \phi_x) \\ \eta_y sat(\sigma_y / \phi_y) \\ \eta_\psi sat(\sigma_\psi / \phi_\psi) \end{bmatrix} \tag{6.19}$$

and $u_1(t)$, $u_2(t)$, and $u_3(t)$ contain the acceleration, velocity and switching terms respectively. The inverse of the input gain matrix for this case is found directly as

$$\hat{g}(\alpha)^{-1} = \begin{bmatrix} 1/2\hat{\alpha}_x & 0 & 0 \\ 0 & 1/2\hat{\alpha}_y & 1/2\hat{\alpha}_\psi \\ 0 & 1/2\hat{\alpha}_y & -1/2\hat{\alpha}_\psi \end{bmatrix}. \tag{6.20}$$

Since the motion is restricted to the horizontal plane,

$$\hat{h}^{-1}(\psi(t)) = \hat{h}^T(\psi(t)) = \begin{bmatrix} cos(\psi(t)) & sin(\psi(t)) & 0 \\ -sin(\psi(t)) & cos(\psi(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.21}$$

246

The control vectors in terms of the individual parameters and gains are

$u_1(t) =$

$$
\left\{
\begin{array}{l}
\left(1/2\hat{\alpha}_x\right)\left(\hat{M}_x \cos\psi(t)\ddot{X}_{com}(t) + \hat{M}_x \sin\psi(t)\ddot{Y}_{com}(t) + \hat{b}_x u(t)|u(t)|\right) \\[2ex]
\left(1/2\hat{\alpha}_y\right)\left(\hat{M}_y\left(-\sin\psi(t)\ddot{X}_{com}(t) + \cos\psi(t)\ddot{Y}_{com}(t)\right) + \hat{b}_y v(t)|v(t)|\right) \\[1ex]
\qquad + \left(1/2\hat{\alpha}_\psi\right)\left(\hat{I}_z\ddot{\psi}_{com}(t) + \hat{b}_\psi r(t)|r(t)|\right) \\[2ex]
\left(1/2\hat{\alpha}_y\right)\left(\hat{M}_y\left(-\sin\psi(t)\ddot{X}_{com}(t) + \cos\psi(t)\ddot{Y}_{com}(t)\right) + \hat{b}_y v(t)|v(t)|\right) \\[1ex]
\qquad - \left(1/2\hat{\alpha}_\psi\right)\left(\hat{I}_z\ddot{\psi}_{com}(t) + \hat{b}_\psi r(t)|r(t)|\right)
\end{array}
\right\}
$$

$u_2(t) =$

$$
\left\{
\begin{array}{l}
\left(\hat{M}_x/2\hat{\alpha}_x\right)\left(\dot{\psi}\left(-\sin\psi(t)\dot{X}(t) + \cos\psi(t)\dot{Y}(t)\right) + \left(\lambda_x \cos\psi(t)\dot{\tilde{X}}(t) + \lambda_y \sin\psi(t)\dot{\tilde{Y}}(t)\right)\right) \\[2ex]
\left(1/2\hat{\alpha}_y\right)\hat{M}_y\left(\dot{\psi}\left(-\cos\psi(t)\dot{X}(t) - \sin\psi(t)\dot{Y}(t)\right) - \lambda_x \sin\psi(t)\dot{\tilde{X}}(t) + \lambda_y \cos\psi(t)\dot{\tilde{Y}}(t)\right) \\[1ex]
\qquad + \left(1/2\hat{\alpha}_\psi\right)\hat{I}_z\lambda_\psi\dot{\tilde{\psi}}(t) \\[2ex]
\left(1/2\hat{\alpha}_y\right)\hat{M}_y\left(\dot{\psi}\left(-\cos\psi(t)\dot{X}(t) - \sin\psi(t)\dot{Y}(t)\right) - \lambda_x \sin\psi(t)\dot{\tilde{X}}(t) + \lambda_y \cos\psi(t)\dot{\tilde{Y}}(t)\right) \\[1ex]
\qquad - \left(1/2\hat{\alpha}_\psi\right)\hat{I}_z\lambda_\psi\dot{\tilde{\psi}}(t)
\end{array}
\right\}
$$

$$u_3(t) \;=\;$$

$$\left\{ \begin{array}{l} \left(1/2\alpha_x\right)M_x\Big(\eta_x\cos\psi(t)sat\big(\sigma_x/\phi_x\big) \;+\; \eta_y\sin\psi(t)sat\big(\sigma_y/\phi_y\big)\Big) \\[2em] \left(1/2\alpha_y\right)M_y\Big(-\eta_x\sin\psi(t)sat\big(\sigma_x/\phi_x\big) \;+\; \eta_y\cos\psi(t)sat\big(\sigma_y/\phi_y\big)\Big) \\[1em] \quad +\; \left(1/2\alpha_\psi\right)I_z\eta_\psi sat\big(\sigma_\psi/\phi_\psi\big) \\[2em] \left(1/2\alpha_y\right)M_y\Big(-\eta_x\sin\psi(t)sat\big(\sigma_x/\phi_x\big) \;+\; \eta_y\cos\psi(t)sat\big(\sigma_y/\phi_y\big)\Big) \\[1em] \quad -\; \left(1/2\alpha_\psi\right)I_z\eta_\psi sat\big(\sigma_\psi/\phi_\psi\big) \end{array} \right\}$$

The control voltages to each actuator is given by

$$v_x(t) \;=\; \sqrt{|\gamma_x(t)|}sgn(\gamma_x(t)) \tag{6.22}$$

$$v_{blt}(t) \;=\; \sqrt{|\gamma_{blt}(t)|}sgn(\gamma_{blt}(t)) \tag{6.23}$$

$$v_{slt}(t) \;=\; \sqrt{|\gamma_{slt}(t)|}sgn(\gamma_{slt}(t)) \tag{6.24}$$

where

$$\gamma_x(t) \;=\; u_{1_{(1,1)}} \;+\; u_{2_{(1,1)}} \;+\; u_{3_{(1,1)}}$$

$$\gamma_{blt}(t) \;=\; u_{1_{(2,1)}} \;+\; u_{2_{(2,1)}} \;+\; u_{3_{(2,1)}}$$

$$\gamma_{slt}(t) \;=\; u_{1_{(3,1)}} \;+\; u_{2_{(3,1)}} \;+\; u_{3_{(3,1)}}$$

Now that the dynamic model of the vehicle has been established, the next step required to perform local area navigation is the selection of a suitable target for reference. This topic will be covered in the next section.

B. TARGET DETECTION WITH SONAR

To perform local area navigation using sonar, it is necessary to select an easily discernible feature in the vehicle operating area and use it as a fixed reference. To enable repeatable and unambiguous detection of the reference feature, the target feature should be stationary and reasonably unique with respect to other structures in the sonar field of view. Also, to classify these features, each must be segmented into a separate object and analyzed to see if it posses the structural properties of the desired target for reference.

For the results presented, the target used for the local navigation reference was a 1.0 foot diameter, 2.5 foot long cylinder placed vertically in the water column of the NPS test tank. The Tritech ST1000 profiling sonar head was used mounted vertically in the nose of the Phoenix vehicle. The head uses a stepper motor which can mechanically rotate the transducer through 360° with respect to it's mounting at a minimum angular resolution of 0.9^O. For each step, the sonar is pinged and a single range value is returned which enables a complete profile of the area surrounding the vehicle to be constructed.

An actual scan of the cylindrical target and square tank walls is shown in Figure 6.1. A sweep width of $\pm 35^O$ and angular resolution of 1.8^O was used. Each dot or "pixel" corresponds to a discrete range value returned by the sonar for a given angular position of the transducer head. Several disjoint groups or segments of pixels are visible in the field of view: the two sections of the tank wall, and the cylinder which casts an acoustic shadow against the wall. Since sonar range drop outs and noise are common with sonars, the tank wall to the upper right of the cylinder is broken up into several segments, although in reality, it is a continuous feature. It is this nature of acoustic sensors that lead to the development of the following algorithms for cylinder detection in the NPS test tank.

Since the cylinder has been chosen as the desired target for the local area reference, returns from the tank walls will be filtered out and ignored. Separating a cylinder from a wall can be accomplished by segmenting each contiguous, disjoint group of range pixels and analyzing them for the desired characteristics of a cylinder. The basic method to isolate these segments is outlined in the flow diagram in Figure 6.2. The filter algorithm is initialized by pinging several times at a fixed bearing to obtain an average range value, $\bar{r}$. The head is then commanded to scan in a clockwise direction and each range return is first tested for feasibility. If the range is zero or if it exceeds the maximum operating range, r_{max}, it is ignored and that range, r_i, is set to the current average range, $\bar{r}$, and the scan

249

proceeds. If the range is feasible, a test is performed to see if it lies within an error band of $\pm \Delta r$ of the average and if so, the value of $\bar{r}$ is recalculated using the new range. The range and the associated bearing angle is then stored in a vector of size n, (the number of pixels defining the segment). If the range falls outside of the error band, a flag is set to examine how closely subsequent returns compare to the new range. A secondary average, $\bar{r}_{new}$, is initialized to this value and a new segment is declared if the next n_{min} adjacent ranges are consistent with this average at which time the current average is set to $\bar{r}_{new}$. The old segment is now terminated at $i - n_{min}$ and the range, bearing and pixel count values are processed to extract any shape information they may provide. If the subsequent ranges, less than n_{min} pixels are inconsistent with $\bar{r}_{new}$, and fall near the previous average, a new segment is not assumed and the scan continues using $\bar{r}$. These "false alarms" occur quite frequently due to the nature of the sonar returns which contain drop outs and false ranges. The value of n_{min} can be varied depending on the environment of operation. For the test tank which provides relatively clean signals, the value of n_{min} is typically 3, but in more noisy conditions, a larger value should be used to provide higher filtering.

Once a separate segment has been identified, the vector containing it's ranges and bearing angles is analyzed. The flow diagram for this algorithm is shown in Figure 6.3. To determine if the object defined by the segment is a cylinder, it must posses the following characteristics:

1. Consist of a sufficient number of pixels, n, that does not exceed a maximum, n_{max}. If the number of pixels is large, in this case greater than 10 it can be safely assumed the segment is a wall due the relative size of the cylinder.

2. Be in front of the tank walls. This is an obvious observation since the cylinder is assumed to be in the tank but must be included in the algorithm to avoid confusion by perceived cylindrical shaped areas on the wall due to noise.

3. Have a central range closer than it's edges. Since a cylinder appears the same from any direction in a horizontal plane, the center of the segment will always be closer the sonar than the beginning and ending edges.

The preceding algorithms have been used with much success in the NPS test tank and should operate well in an open water environment especially since the tank walls will

be absent and the reference target the most visible object in the area. This procedure can be modified to search for other geometric shapes since the idea of segmentation of each feature is retained but does not attempt to supplant more sophisticated and robust pattern recognition algorithms available. Additionally, this method was adopted since it can be executed in real time and is simply used as a means to perform the tasks described in the following sections.

C.   RELATIVE POSITION ESTIMATION

Once the reference target has been identified, it becomes the origin of the navigation coordinate frame where the X-axis is aligned with heading 0 degrees and the Y-axis along a heading of 90° as shown in Figure 6.4. The two dimensional position vector to the origin of the vehicle body-fixed reference with respect to the navigation frame at detection time T is

$$R_v(T) = \begin{Bmatrix} X_v(T) \\ Y_v(T) \end{Bmatrix} = -(r_s(T) + R_{cyl}(T))$$
(6.25)

where

$$r_s(T) = h(\psi(T))\begin{Bmatrix} x_s \\ y_s \end{Bmatrix},$$
(6.26)

and x_s, y_s is the position of the sonar head in vehicle coordinates.

$$R_{cyl}(T) = \begin{Bmatrix} (R_{cyl}(T) + r_{cyl})\cos(\psi(T) + \psi_s(T)) \\ (R_{cyl}(T) + r_{cyl})\sin(\psi(T) + \psi_s(T)) \end{Bmatrix}$$
(6.27)

where $R_{cyl}(T)$ is the sonar range to the target, $\psi_s(T)$ is the heading angle of the sonar beam, and for the case of a cylindrical target, r_{cyl} is it's radius. After the target and the location of the vehicle is found, the delay time between re-acquisitions is reduced by commanding the sonar to sweep across the target at a prescribed minimum sweep angle ψ_{sw} about a heading to the center of the target.

D. POSITION UPDATE

Because of the physical speed limits to mechanical scanning, there is a delay time of up to 10 seconds between successive detections of the target, the navigational scheme proposed employs the dynamic model between position updates. Eqn. (6.5) is integrated to obtain estimates of the vehicle position denoted $\hat{X}(t)$, and $\hat{Y}(t)$ during this time. The scan direction command angle $\psi_{sd}(t)$ <u>between</u> position updates is computed using

$$\psi_{sd}(t) = atan2\left(\frac{-(\hat{Y}(t) + x_s sin(\psi(t)) + y_s cos(\psi(t)))}{-(\hat{X}(t) + x_s cos(\psi(t)) + y_s sin(\psi(t)))}\right) - \psi(t) \qquad (6.28)$$

and a maneuver using this approach is shown in Figure 6.5. In circumstances where the scan width is too narrow and there exists a large discrepancy between the model and the actual vehicle, the scan direction calculated from the estimates of position can be in error. In these cases, and if the target has not been reacquired within a specified time, the head is then commanded to return to continuous sweep re-acquisition mode. One approach to reduce this possibility would be to increase the scan width, ψ_{sw} to say 120 degrees but this would increase the time between updates and has not been implemented.

For vehicle control in a plane, the complete state is defined by

$$X(t) = \{ u(t)\ v(t)\ r(t)\ X(t)\ Y(t)\ \psi(t) \}^T \qquad (6.29)$$

and the block diagram representation of the control scheme is shown in Figure 6.6. When the cylinder has been identified, the model is asynchronously updated at time of target detection using a Kalman filter of the form

$$\hat{X}(T) = (1 - K)\hat{X}^-(t) + KX_v(T)$$

$$\hat{Y}(T) = (1 - K)\hat{Y}^-(t) + KY_v(T)$$
$$(6.30)$$

252

where

$$K = \frac{\sigma_m^2}{\sigma_m^2 + \sigma_s^2}$$ (6.31)

and σ_m^2 is the variance of the system model estimate of position and σ_s^2 is the variance of vehicle position using the sonar. $\hat{X}^-(t)$, and $\hat{Y}^-(t)$ is the current estimate of position from the model just before the correction from the sonar is obtained. This analysis assumes the position estimate from the sonar is extremely accurate and the model very inaccurate. Therefore, the variance for position from sonar is set to 0 and infinity for the model. This causes the current estimate from the model to be disregard at the time of sonar update and reduces Eqn. (6.30) to

$$\hat{X}(T) = X_v(T)$$

(6.32)

$$\hat{Y}(T) = Y_v(T)$$

which states complete confidence in the sonar. At this time the dynamic model state is reset to the values obtained from Eqn. (6.32).

The word "Kalman" is somewhat loosely used here as the model error variance, σ_m^2 is not predicted and propagated along with the positional estimates. However, the fusion gain, K, is computed on the basis of a minimum fusion error variance for Gaussian errors assuming that the error statistics are known a priori.

The onboard gyroscopes provide the heading angle and yaw rate values at 10 Hz, which are synchronous and highly accurate and no estimation of these is required. The observation vector is defined by

$$y(t) = CX(t)$$ (6.33)

where the observation matrix is

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

With only these two measurements made, Eqn. (6.33) reduces to

$$y(t) = \begin{Bmatrix} r(t) \\ \psi(t) \end{Bmatrix} \tag{6.34}$$

which is used each time step in the vehicle controller and dynamic model along with the set point vector

$$r(t) = \begin{Bmatrix} X_{com}(t) \\ Y_{com}(t) \\ \psi_{com}(t) \end{Bmatrix}. \tag{6.35}$$

E.   SONAR ENVIRONMENT SIMULATION

Before in water testing was performed, simulation of the effectiveness of the local area navigation algorithms were studied by simulation. Since the purpose of this work is to interact with the underwater environment using the sonar, the existing dynamic model of the vehicle alone would not be sufficient to test and evaluate these algorithms. The dynamic model is simulated by control inputs to the actuators only and has no knowledge of an external environment such as submerged obstacles, the bottom, or even other vehicles in the immediate area. This prompted the creation of a simple but effective simulator which modeled the physical environment in the computer using a collection of objects made up of three-dimensional rectangular polygons. Coupling the geometric representation of the sonar head motions and the dynamic model of the vehicle, enabled the navigation algorithms to be quickly evaluated and modified before in water testing was performed, and was found to be an essential development tool.

254

The underwater environment was built from 3-D rectangular polygons which means there exists for each vertex an ordered triple (x, y, z) in a global coordinate system. The polygons which define the NPS test tank with cylindrical target is shown in Figure 6.7 and are labeled 0 through 18. The sides of the cylinder not shown are unlabeled for clarity. This gives a very good representation of the operating environment for the actual tests to be done. Once the physical shape of the environment was defined, detection of the objects is performed using a polygon intersection technique described in detail in Appendix H. The line that intersects the polygon represents the sonar beam, and the origin of the beam is $P(0)$ and the current direction the beam is pointed is described by unit vector $\mathbf{e}$. Solution of Eqn. (H.7), which is the point of intersection of the sonar beam is the range to that particular plane. This operation is repeated for each plane describing the environment to determine if the beam intersects that polygon. If it does, the range is calculated and registered for that sonar beam heading. The sonar is assumed to be attached to the vehicle with axis of rotation parallel to the body-fixed z-axis which will undergo the same motions the vehicle experiences. Therefore, the unit vector, $\mathbf{e}$, describing the beam direction in body-fixed coordinates is

$$
\mathbf{e} = \begin{Bmatrix} cos(\psi_s) \\ sin(\psi_s) \\ 0 \end{Bmatrix} \tag{6.36}
$$

where ψ_s is the heading angle of the sonar head which can be stepped in increments of 0.9, 1.8, or 3.6 degrees. Figure 6.8 shows the vector relationships between the vehicle and polygonal environment. The point of intersection, $\boldsymbol{P_c}$, in global coordinates is given by

$$
\boldsymbol{P_c} = \boldsymbol{R_0} + h(\boldsymbol{r_s} + \boldsymbol{r}) \tag{6.37}
$$

where h is the local to global transformation matrix. The vector $\boldsymbol{r_s}$ is the position of the sonar head, and $\boldsymbol{r}$ is the sonar range vector both expressed in body-fixed coordinates The vector $\boldsymbol{r}$ may also be expressed as $\mathbf{e}r$ where r is the scalar magnitude of the range which can be calculated from

$$
r = \left| h^T (\boldsymbol{P_c} - \boldsymbol{R_0}) - \boldsymbol{r_s} \right|. \tag{6.38}
$$

The range along with the sonar heading angle ψ_s, will be the sonar information available to the vehicle controller.

Since the controller relies on an uncertain dynamic model using the best estimates of the vehicle parameters, some mismatch will be present with the actual system. Therefore, the simulation developed two models, one using what is considered the "true" parameter values, and the other, estimates of these values, employed by the vehicle controller between sonar updates. The simulator more closely represents the situation for in-water tests since only estimates of the actual vehicle parameters are available and robustness issues are then elucidated. The sonar head position, and consequently the range values returned, are based on movements from the "true" model. The parameters which were allowed to differ are the masses, M_x, M_y, I_z, the damping, b_x, b_y, b_ψ, and the thruster gains, α_x, α_y, α_ψ.

A Silicon Graphics workstation was used for the simulation. During execution, the polygonal environment and vehicle movements were animated and displayed to provide a visual verification of the control technique. The simulation control modules were designed to have exactly the same structure and functionality as would execute in the vehicle Execution Level. This enabled no code changes to be required between the simulator and the vehicle, thus significantly reducing the software debugging process for the in-water tests to follow.

F.   SIMULATION RESULTS

The simulation comprised of five commanded poses with respect to the target as listed in Table 6.1 and shown graphically in Figure 6.9.

Table 6.1 Commanded Mission Poses

Pose	X_{com} (ft)	Y_{com} (ft)	ψ_{com} (rad)
1	-7.0	-3.0	0.0
2	-7.0	0.0	0.0
3	-7.0	3.0	0.0
4	-7.0	0.0	0.5236
5	-9.0	-3.0	0.0

The following tables give the parameter values used in the vehicle model and the sliding mode controller gains.

Table 6.2 Parameters for Vehicle Model   Table 6.3 Sliding Mode Controller Gains

Parameter	Value	Unit
m	435.0	lb
m_{ax}	43.5	lb
m_{ay}	348.0	lb
I_{zz}	53.60	lb-ft-sec^2
I_{zza}	53.60	lb-ft-sec^2
b_x	1.33	lb-sec^2/ft^2
b_y	17.0	lb-sec^2/ft^2
b_ψ	55.87	lb-ft-sec^2
α_x	0.025	lb/V^2
α_y	0.004	lb/V^2
α_ψ	0.006	lb-ft/V^2

Parameter	Value	Unit
λ_x	0.20	rad/sec
λ_y	0.20	rad/sec
λ_ψ	0.20	rad/sec
η_x	0.5	m/sec^2
η_y	0.3	m/sec^2
η_ψ	0.20	rad/sec^2
ϕ_x	0.2	ft/sec
ϕ_y	0.3	ft/sec
ϕ_ψ	0.20	rad/sec

Note: $\alpha_{slt} = \alpha_{blt} = \alpha_y l_{lt}$, where l_{lt} is the distance from the mass center of the vehicle to the center of the lateral thruster axes which is the same for both thrusters.

The controller used step inputs in position while the commanded rates were set to zero. The control phase transitions were based on the position and rate errors in the X, Y, and ψ directions. Using these three directions, the error surface was formulated as

$$\sigma(T) = \begin{Bmatrix} \sigma_X(T) \\ \sigma_Y(T) \\ \sigma_\psi(T) \end{Bmatrix} = w_e |e(T)| \tag{6.39}$$

where

$$e(T) = \begin{Bmatrix} X_{com}(T) - X_v(T) \\ Y_{com}(T) - Y_v(T) \\ \psi_{com}(T) - \psi(T) \end{Bmatrix}. \tag{6.40}$$

257

This formulation differs from the definition given in Chapter III for signal transition since it does not include the velocity error. This was done because the transition was based on position errors from the sonar ranges, not from the model estimates since the model will generally predict a very smooth trajectory to the set point prior to sonar update, regardless of the actual vehicle position. This led to the monitoring of the transition only at time of update, T, and the velocities near the terminal phase of the maneuver were assumed to be small. The parameters for the error equation used were $\sigma_{0X} = \sigma_{0Y} = 0.5$ feet, $\sigma_{0\psi} = 0.1$ radians, and $w_e = 1.0$.

The simulation results assume no parameter mismatch between the model for control and the true system and uses a control loop step time of 0.2 seconds (5 Hz). This value, instead of the usual 10 Hz rate was used based on preliminary timing tests using the vehicle CPU. Addition of the dynamic model and sonar control overheads required the control step size to be doubled in order to complete all the calculations. The X and Y position response with respect to the target is shown in Figure 6.10. The set points commanded in Table 6.1 are achieved quite easily and the control model is predicting very closely the actual position of the vehicle given by the sonar (asterisks). It should be noted that the slight discrepancy of position between the model and sonar is due to a time lag between target identification and model update. Recalling the identification methods of section 6.4, a new segment must be declared before the prior one is analyzed. This causes a delay of approximately 2.0 seconds before the coordinates of the cylinder are calculated and passed to the model for update, at which time the state has evolved 2.0 seconds beyond the time of observation. Fortunately, the delay is too small to cause instability for such a slow moving vehicle. The range and bearing angle of the sonar is shown in Figure 6.11. It clearly shows the dynamic tracking performance of the head throughout the maneuver.

G.   EXPERIMENTAL RESULTS

The in-water experiment used the same vehicle and controller parameters used in the simulation along with the same transition procedure. The vehicle was commanded to submerge to a depth of 1.2 feet using vertical thrusters as detailed in Chapter V. Once this depth was reached, the ST1000 sonar was activated and scanned clockwise until the target (cylinder) was identified. At this time, the first pose (1) was issued and the vehicle started the controlled maneuver.

Figure 6.12 shows the position response results where the upper trace is $\hat{Y}(t)$ and the lower $\hat{X}(t)$. The position calculated from sonar at update, $X_v(T)$ and $Y_v(T)$ are shown with circles and asterisks respectively. Examining the response for $\hat{X}(t)$ it is evident that the model for the longitudinal direction is in error since the predicted position at the time of correction is about double that calculated with the sonar. This mismatch has been attributed to errors in the development of forward thrust on the vehicle in transient conditions (Healey, et. al., 1995). The absence of shrouds around the stern screws appears to lead to an unmodeled transient force lag is present that is common with open propellers. Since this lag was uncompensated, and the control was dictated by the model predictions between position updates, large voltage commands to the screws were of too short a duration to build up sufficient force on the vehicle as shown in Figure 6.14. The performance was further degraded by the estimated position and rate feedback from the model. As these values were assumed to be nearing the set point pose, the controller actually reversed the propellers (negative voltage command) in an attempt to slow the vehicle. This effect can also be clearly seen in Figure 6.14 between the time 44.6 seconds and 55.0 seconds, the time of the position update from the sonar. The prediction of the lateral movement, $\hat{Y}(t)$, is much more precise since the cross-body thrusters are shrouded due to their tunnel design, and the model parameters are well established.

1. Thruster Lag Analysis

In an effort to confirm the effect and quantify the inherent propeller force lag, attention was returned to the simulator used before. The equation of motion for the longitudinal direction was modified to include a first order force lag, and in discrete form is

$$F_{x_l(i+1)} = F_{x_l(i)}e^{-T/\tau} + F_x(1 - e^{-T/\tau}) \tag{6.41}$$

where T is the control loop time step, τ is the time constant, F_x is the longitudinal force commanded by the controller, and F_{x_l} is the resulting lagged force applied to the "true" model. The model used by the controller did not have this lagging effect included in it's formulation since this was not the case during the in-water experiment. Various values of the time constant, τ, were used in the simulation to match the results obtained from the actual experiment. The value which provided the best match was 14.0 seconds. The comparison between the actual and simulated responses for the first pose maneuver is

shown in Figure 6.15. Very close matching is evident for this choice of time constant. As time proceeds, the match begins to diverge which is caused by two reasons. The first is due to the fact that the cylinder is only approximated with flat polygons in the simulation which causes the identification algorithm to behave differently for a given number of scans. The second and most significant is the lack of noise and outliers seen in the actual test and not modeled in the simulation. This causes the simulation to reaquire the target very rapidly which causes the update time for the simulation to slowly but consistently out pace the results from the in-water experiment. With this better understanding of the vehicle model, the lag can be compensated to improve the performance and will be the subject of future work.

H.   HIGH SPEED TRACKING

Although the results from the previous section are acceptable, they were still in need of improvement, especially in the area of position update rate. The target was found on average every 10 seconds and caused a rather long time to complete the 5 pose maneuver.

A new high speed algorithm was therefore studied using the standard target locating technique to initially acquire the cylinder but instead of activating a wide sweep of the sonar, the sweep direction was directed to the calculated center of the cylinder. To correct for the relative motion, an algorithm to maintain lock was devised and is shown pictorially in Figure 6.16. The use of a sweep width of 3.6 and step size of 1.8, consists of only a left, center, and right heading which enables the direction the target has apparently moved relative to the sonar to be determined. As seen in the figure, the scan direction ψ_{sd} is changed depending on which beam loses lock, either the left or the right. If lock is lost from the left beam, ψ_{sd} is incremented by 3.6 degrees and decremented by the same amount, if the right beam fails to hit the target. This method assumes relatively slow vehicle speed to prevent exceeding the bandwidth of the sonar. This must be seriously considered if the motion of the vehicle becomes too fast, the sonar will not be able to slew the head fast enough to maintain lock, especially near the target due to the high angular rate of change between the target and vehicle.

Since the control will rely on the sonar range signal, the velocity of the vehicle must be determined by using the standard form of the kinematic Kalman filter with thresholding used throughout this work. The formulation will also be useful for rejecting the large range values sensed when the sonar beam drops off the target and for any outliers caused by

260

multi-path or noise. The position of the vehicle is determined by using the transformation Eqn. (6.27), but in this case, it is used each time step for update, not at the asynchronous time T. Another modification is the deletion of the cylinder radius, r_{cyl}, in the calculation of the range to the cylinder, R_{cyl}, (Eqn. (6.27)). The new form of Eqns. (6.25) and (6.27) become

$$R_v(t) = \begin{Bmatrix} X_v(t) \\ Y_v(t) \end{Bmatrix} = -(r_s(t) + R_{cyl}(t))$$

(6.42)

where the sonar range to the target is

$$R_{cyl}(t) = \begin{Bmatrix} R_{cyl}(t)\cos(\psi(t) + \psi_s(t)) \\ R_{cyl}(t)\sin(\psi(t) + \psi_s(t)) \end{Bmatrix}$$

(6.43)

With these modifications and the fact that the update is now assumed to be synchronous, the block diagram for the control changes from the representation in Figure 6.6 describing asynchronous updates, to appear as in Figure 6.17. The values of $X_v(t)$ and $Y_v(t)$ are individually filtered and if either residual exceeds a value of 1.0 feet, the corresponding range is assumed to be an outlier or caused by loss of lock. In this case the residual is zeroed and the estimates for position and velocity are obtained from the kinematic model alone, as was done for wall servoing described earlier. The filtering generates estimates for the position and rate vectors given by

$$\hat{R}_v(t) = \begin{Bmatrix} \hat{X}_v(t) \\ \hat{Y}_v(t) \end{Bmatrix}$$

(6.44)

and

$$\dot{\hat{R}}_v(t) = \begin{Bmatrix} \dot{\hat{X}}_v(t) \\ \dot{\hat{Y}}_v(t) \end{Bmatrix}$$

(6.45)

respectively. The range was transformed to $\hat{X}_v(t)$, $\hat{Y}_v(t)$ before filtering instead of filtering the range then transforming, since previous work indicated this ordering provides a much smoother result, (Zinni, 1995, Scrivener, 1996).

261

Previously it was assumed that the range information that arrived asynchronously from the cylinder identification algorithm was extremely accurate. This is not a bad assumption since the algorithm is quite robust and a single value of range and sonar heading is obtained per update. However, attempting to use the ranges obtained synchronously from a beam that strikes a cylindrical object at discrete angles can cause relatively large range changes from heading to heading. This is especially acute near the edges, and the filter predicts very large range rates from these. Reduction of the Kalman gain to remove this undesirable effect can introduce lags which can lead to instability of the entire control system. A more appropriate method is to continuously fuse the position and velocity values from the kinematic filter with the estimates obtained from the model. Therefore, the variances from the model and sonar should be non-zero to provide additional conditioning of the position and velocity. These assumptions form the basis for a new set of Kalman filter equations supplanting Eqn. (6.30) given by

$$\hat{X}_f(t) = (1 - K)\hat{X}(t) + K\hat{X}_v(t)$$

$$\hat{Y}_f(t) = (1 - K)\hat{Y}(t) + K\hat{Y}_v(t)$$

(6.46)

where the velocity is also calculated using

$$\dot{\hat{X}}_f(t) = (1 - K)\dot{\hat{X}}(t) + K\dot{\hat{X}}_v(t)$$

$$\dot{\hat{Y}}_f(t) = (1 - K)\dot{\hat{Y}}(t) + K\dot{\hat{Y}}_v(t)$$

(6.47)

where the subscript "f" indicates a fused value and the gain K is obtained from Eqn. (6.31).

If target lock is lost for more than three consecutive pings, the tracking algorithm has lost lock for a longer period than can be remedied using a sweep width of 3.6 degrees and reacquisition of the target must be undertaken. During this time, the variances of the sonar range is set to 1.0 and 0.0 for the model respectively to force reliance solely on the model estimates for control. At this time the kinematic Kalman filter for the sonar range is deactivated to prevent divergence. Currently, the action taken to reacquire the target involves expanding the sweep width to ever increasing multiples of 3.6 degrees. During this time, the model is used to predict the location of the target using Eqn. (6.28) from the estimates of position. If lock is not obtained within a specified time, the mission is terminated. If lock

is re-established, the kinematic Kalman filter is reactivated and initialized to the new values of position.

Figure 6.18 shows the simulated position response in the Y direction using the new method with the commanded poses and vehicle/controller parameters of section 6.9. Since it is extremely important to regulate the vehicle speed to keep the motions within the bandwidth of the sonar movement, command generators were used for all three control directions, X, Y, and ψ. Comparing with Figure 6.10, the mission time is not significantly reduced since the maneuvers were dictated by command generators which had a 20 second elapsed time specification for each segment. This had to be done otherwise lock on the target would have been lost if a faster response was chosen. Analysis of Figure 6.19 shows that the sonar head responds to the loss of track by correcting the scan direction, ψ_{sd} in accordance to the rules defined above.

The method described above has been used to obtain some preliminary data in the test tank. The results were very promising as long as lock was maintained on the cylinder, but once lock was lost, the algorithm showed poor performance in re-acquiring lock. This was mostly due to shortcomings in the tracking software and the performance of the ST1000 sonar was degraded due to a suspected electronic temperature fault in the head. Further software enhancements and hardware repairs will be needed to overcome these problems.

I.   CONCLUSIONS

The results of these experiments have shown that it is possible to navigate an underwater vehicle in a local area using an acoustic sensor for position information. The accuracy of the model used between updates is moderately satisfactory and can allow for time varying currents. However, some additional model adjustments could be made to compensate for the force lag in the longitudinal direction during transient thrust conditions. This undesirable effect could also be alleviated physically by the addition of shrouds around the stern screws which should bring the performance up to that of the lateral thrusters. While these results were taken in a tank environment, another improvement would be to fuse the model with an INS system in between updates from the sonar and then fuse that estimate with the sonar data to obtain a smoother averaging at update time. This would allow for compensation of wave induced disturbances while retaining the positioning precision found. Since the sonars are mechanically scanned, and a delay of up

to 10 seconds between position update is common, use of an electronically scanned or multi-beam sonars may be preferable although our experience to date has been that crosstalk between beams can be a serious problem.

To increase the response time, a continuous target tracking algorithm was simulated. This provided extremely favorable results but will need further modifications to successfully operate in the actual vehicle.
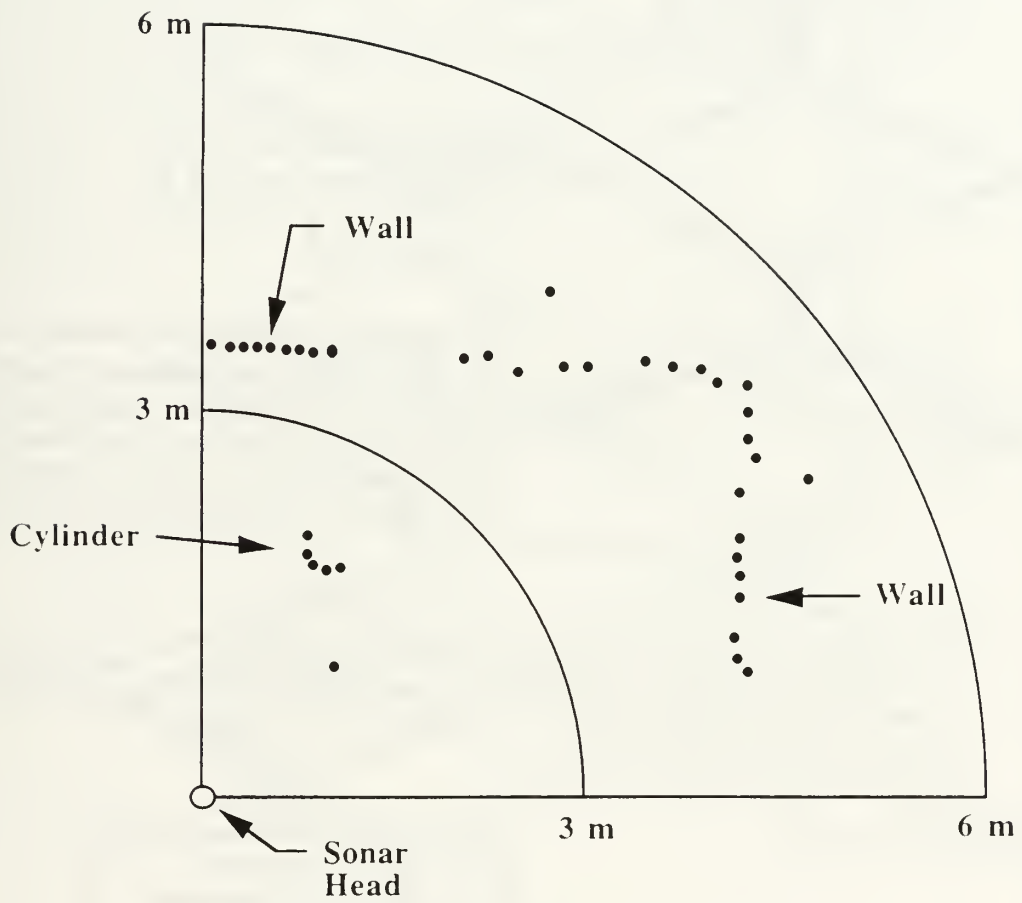
J.   CHAPTER VI FIGURES

Figure 6.1 ST1000 Sonar Image of a Cylinder and the NPS Test Tank.

Read r_i from Sonar

if($r_i == 0 \parallel r_i > r_{max}$)

TRUE

Bad return. Set r_i to current average range

FALSE

if($r_i < \bar{r} + \Delta r \ \&\& \ r_i > \bar{r} - \Delta r$)

TRUE

r_i is outside error bounds. Could be start of new segment. Begin count to check consistency of r_i with subseqent ranges. Start a secondary average $\bar{r}_{new}$

FALSE

r_i is within error bounds Recalculate $\bar{r}$ with r_i and save ranges and bearing angles in separate vector

Are there n_{min} adjacent pixels near average

FALSE

Continue

TRUE

NEW SEGMENT
Set $\bar{r} = \bar{r}_{new}$ and terminate old segment @ index $i - n_{min}$
Begin analysis of it's shape.
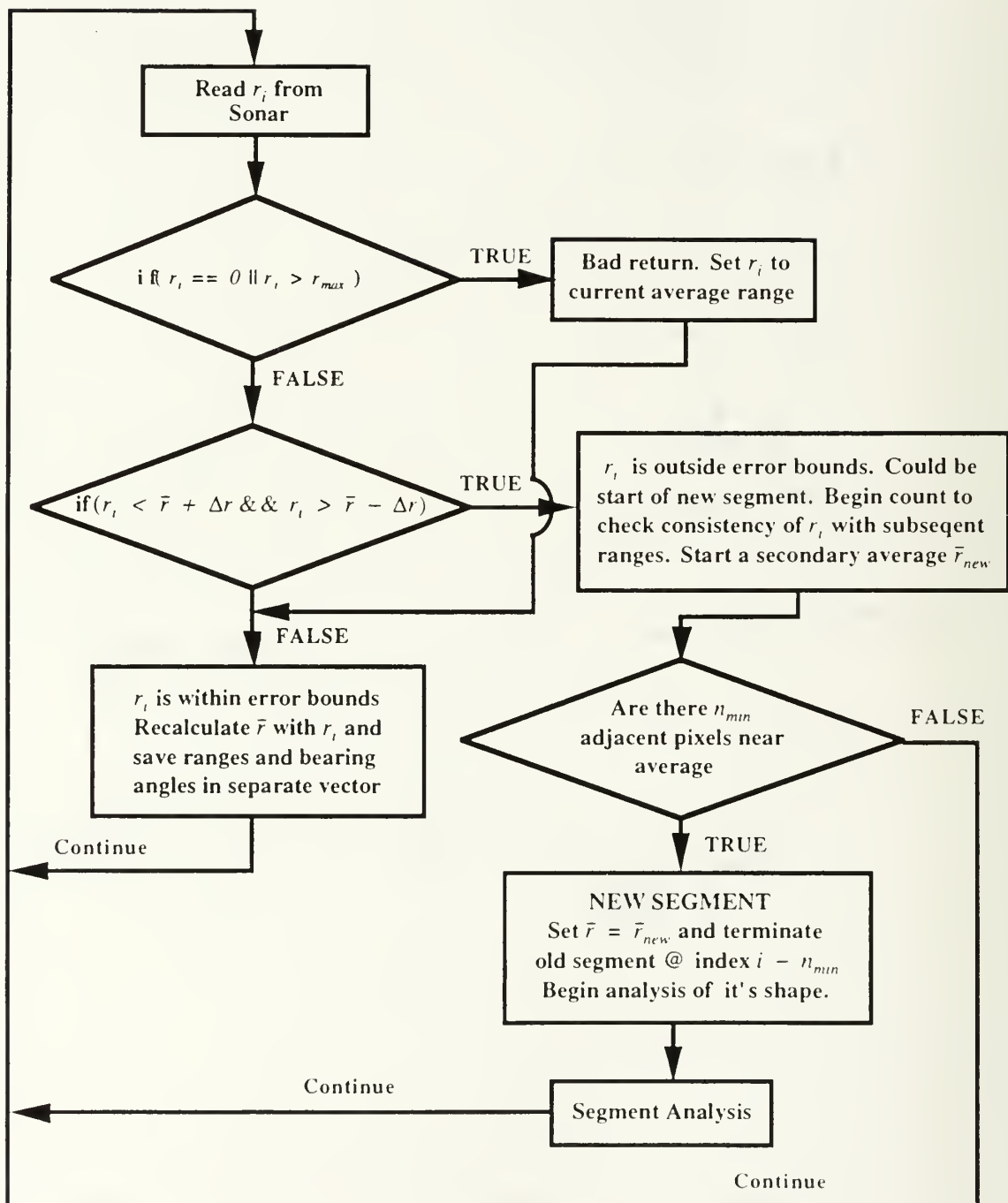
Continue

Segment Analysis

Continue

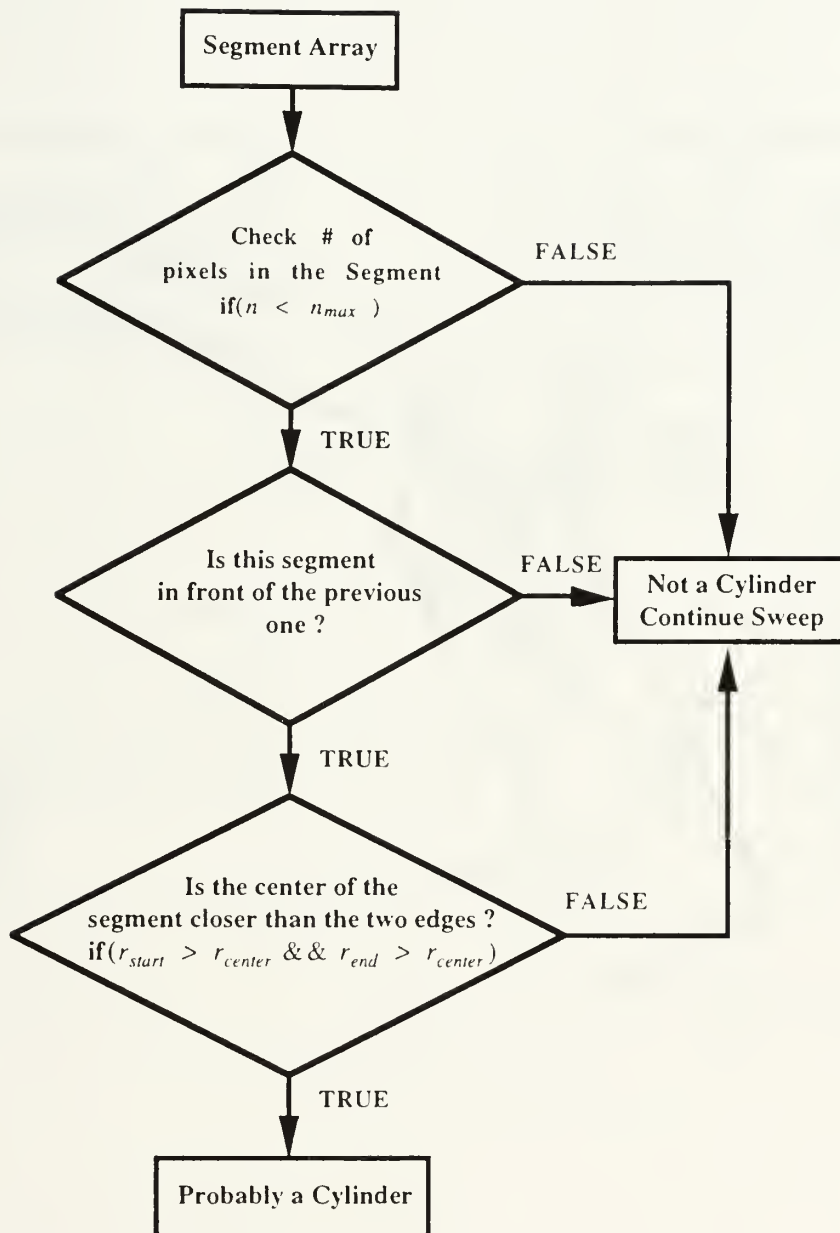Figure 6.2 Object Segmenting Algorithm Flow Diagram.

Figure 6.3 Object Segment Shape Algorithm Flow Diagram.
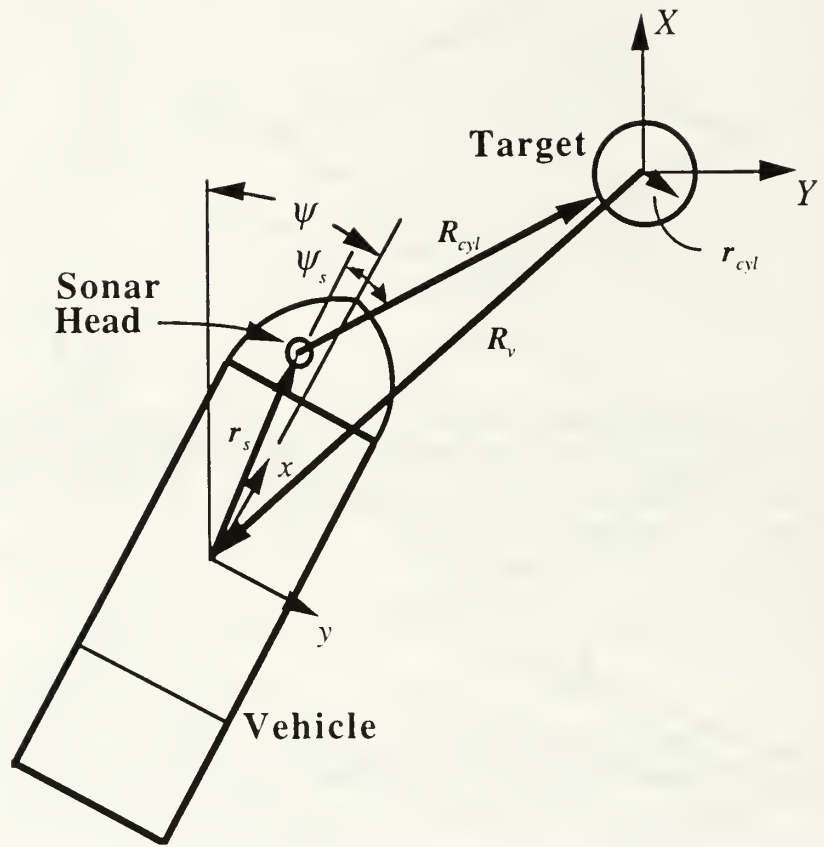
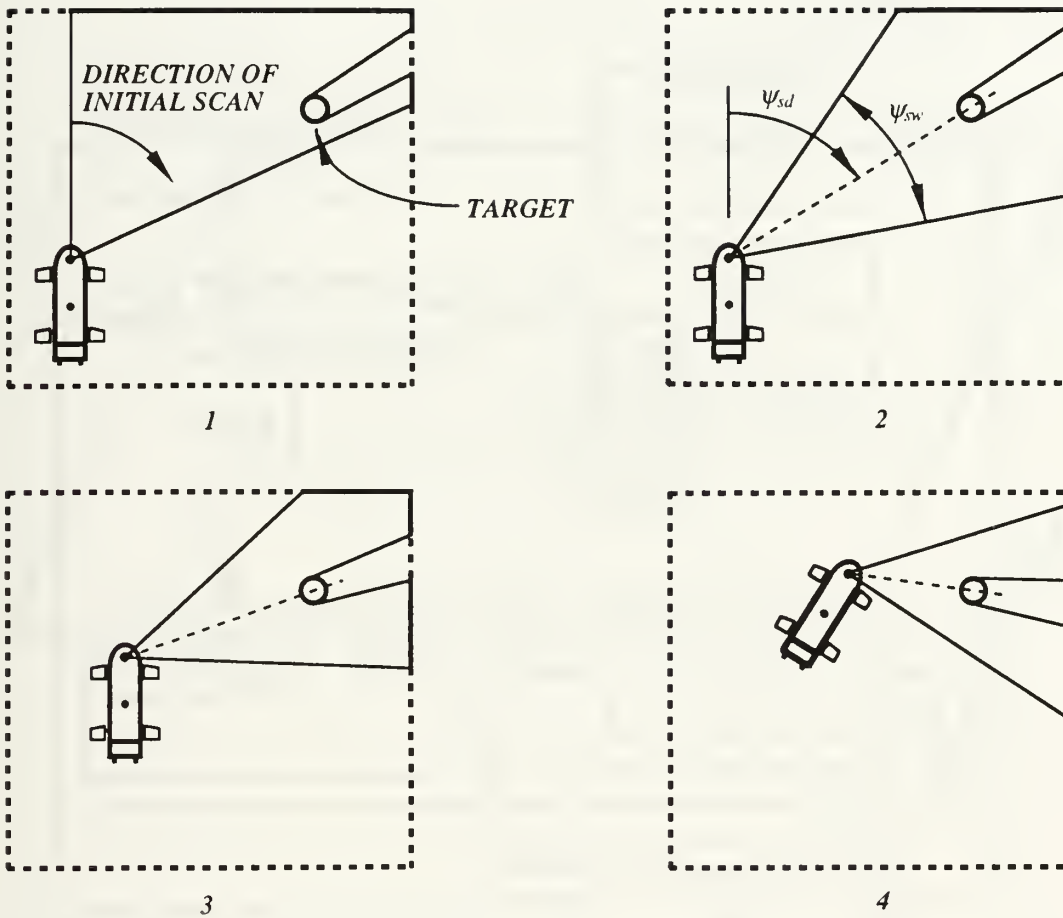Figure 6.4 Position Vector Definitions for Local Area Navigation.

270

Figure 6.5 Sonar Scan Patterns for Maneuvers with Respect to a Cylinder.
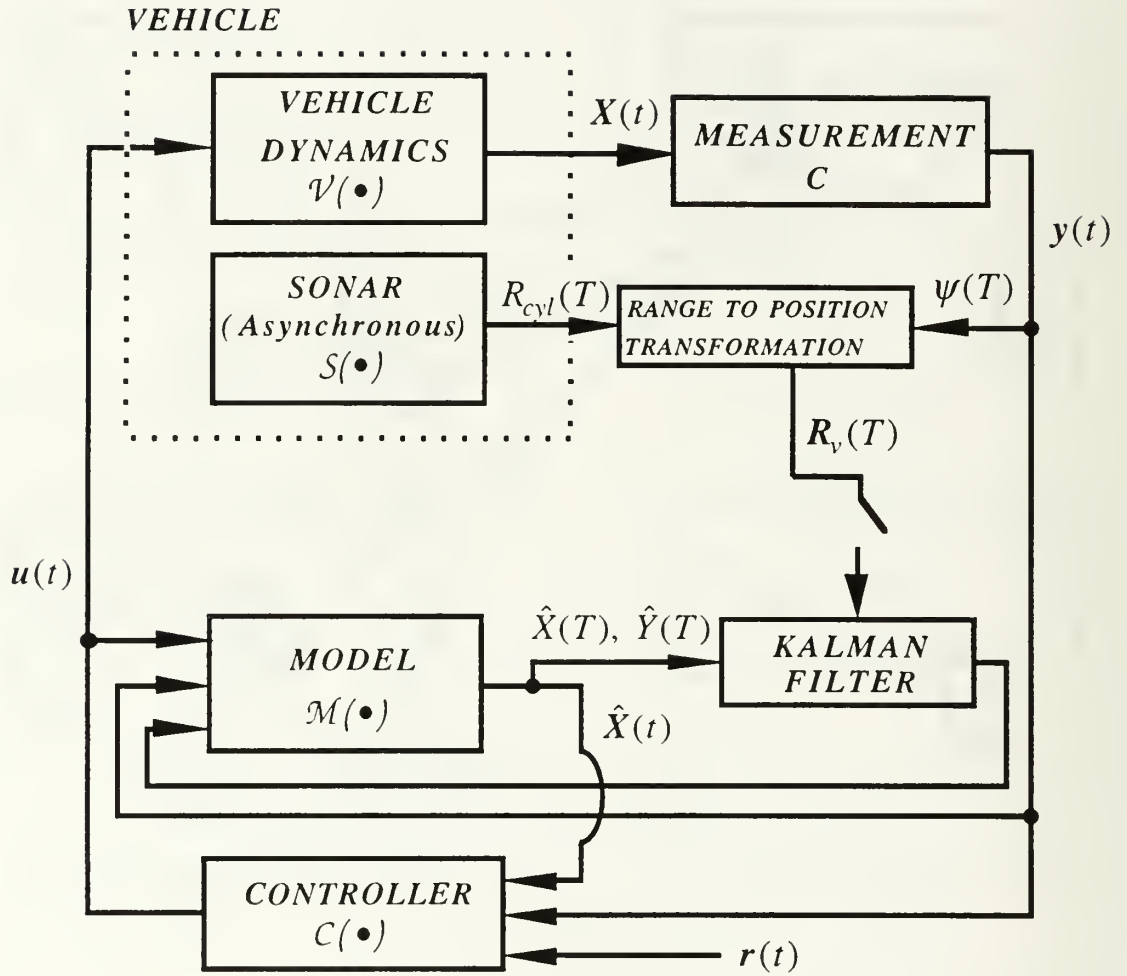
Figure 6.6 Sonar with Model Control Block Diagram.
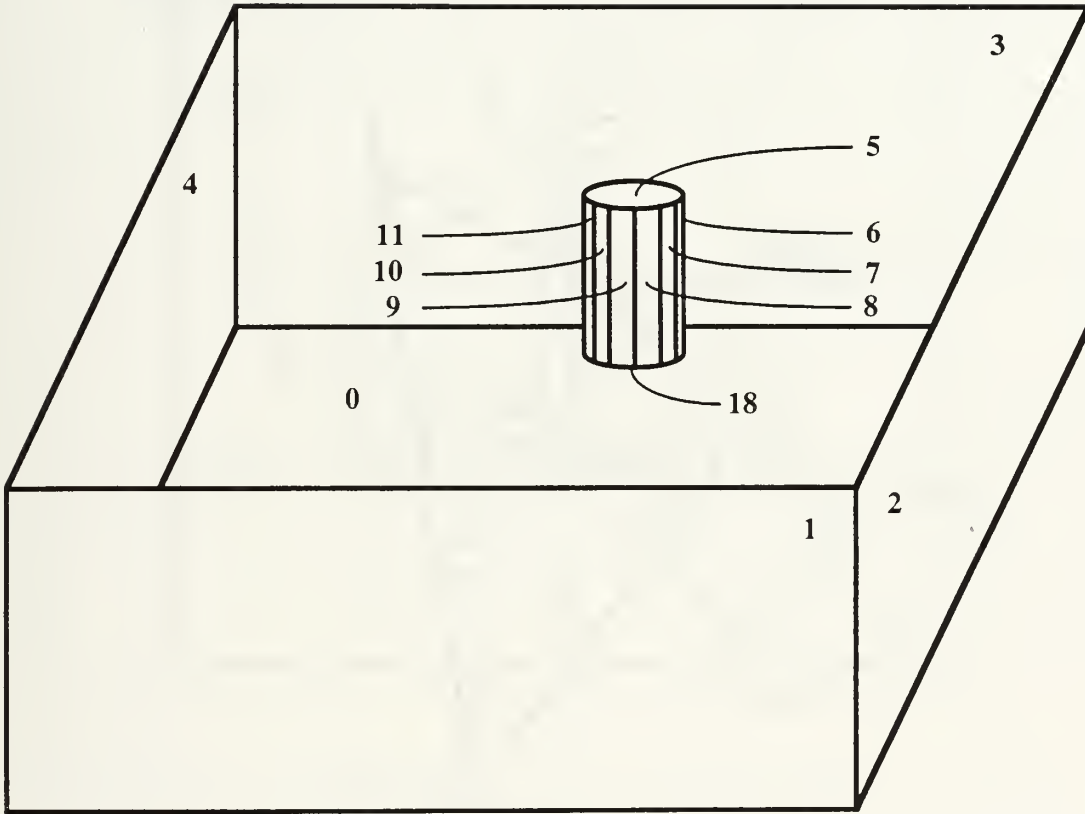
Figure 6.7 NPS Test Tank with Cylindrical Target Polygonal Representation.
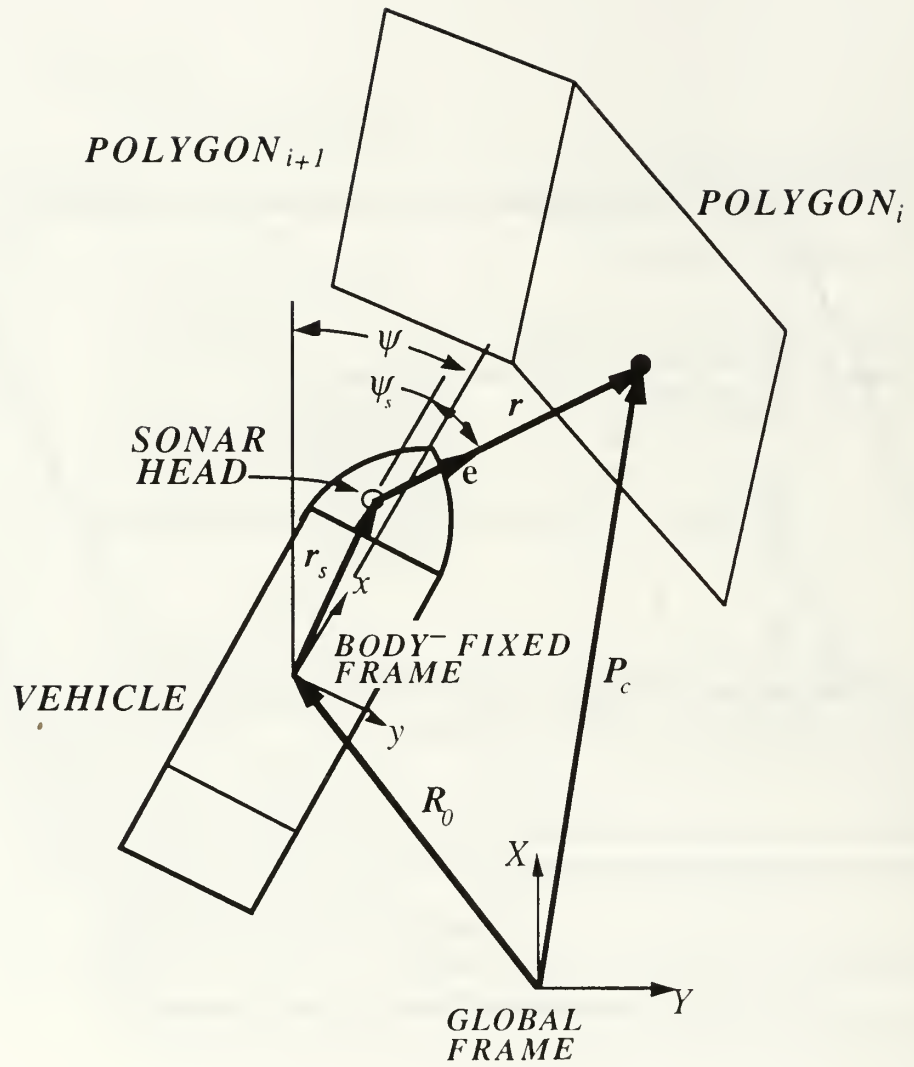
Figure 6.8 Position Vector Definitions for Simulation Environment.

274
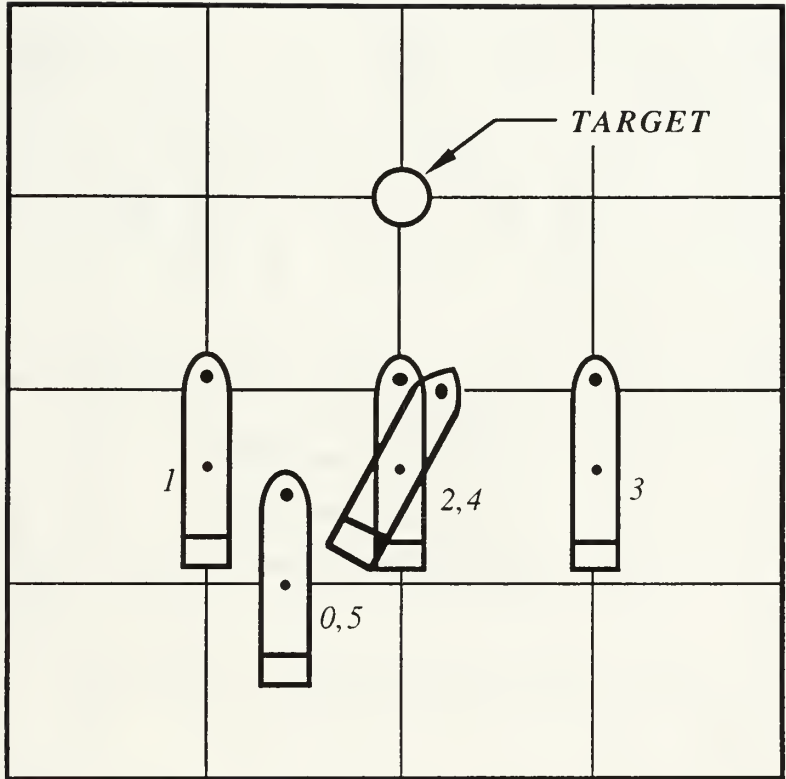
Figure 6.9 The Five Commanded Poses.
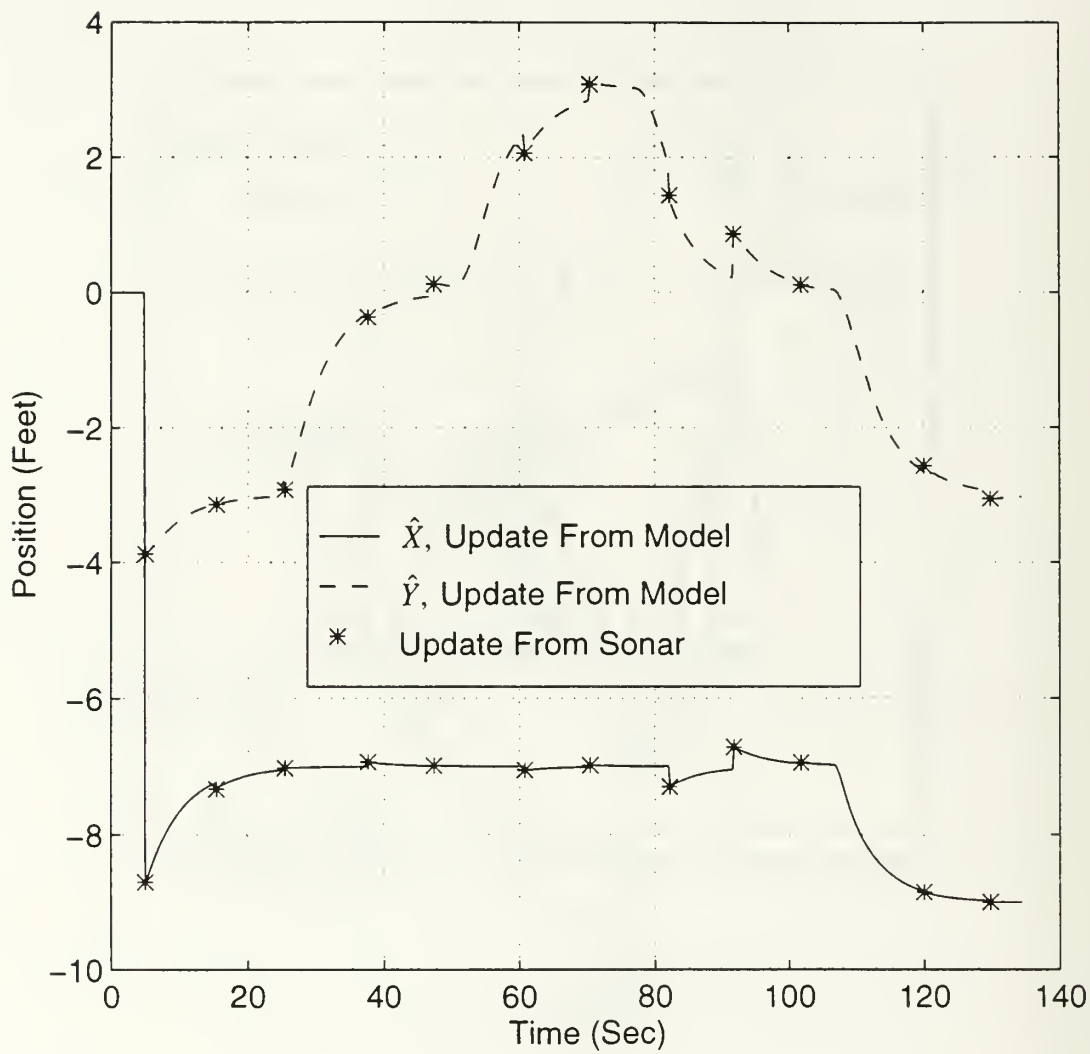
Figure 6.10 Position Response vs. Time (Simulation).
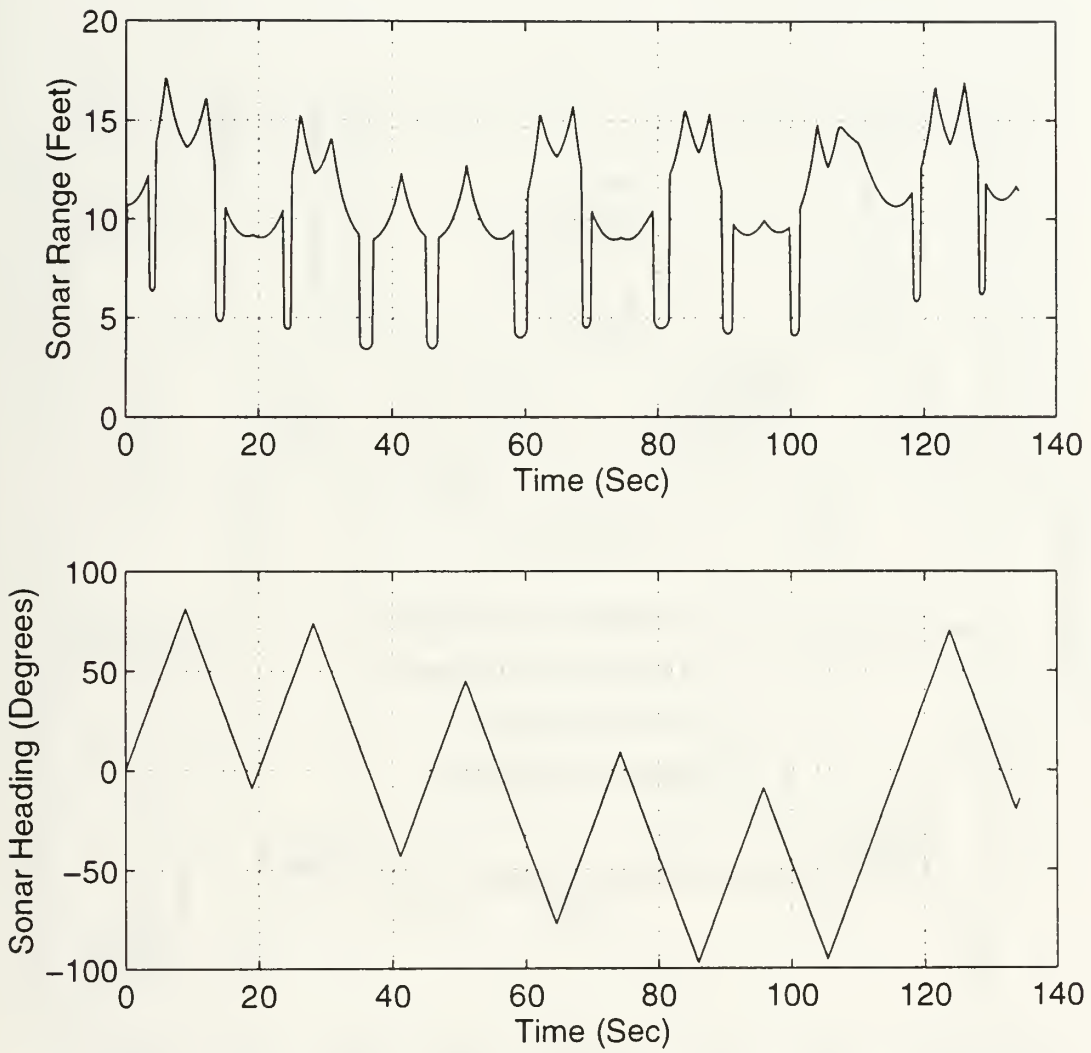
Figure 6.11 Sonar Range Returns and Heading Angle vs. Time (Simulation).

Figure 6.12 Position Response vs. Time (Experimental).
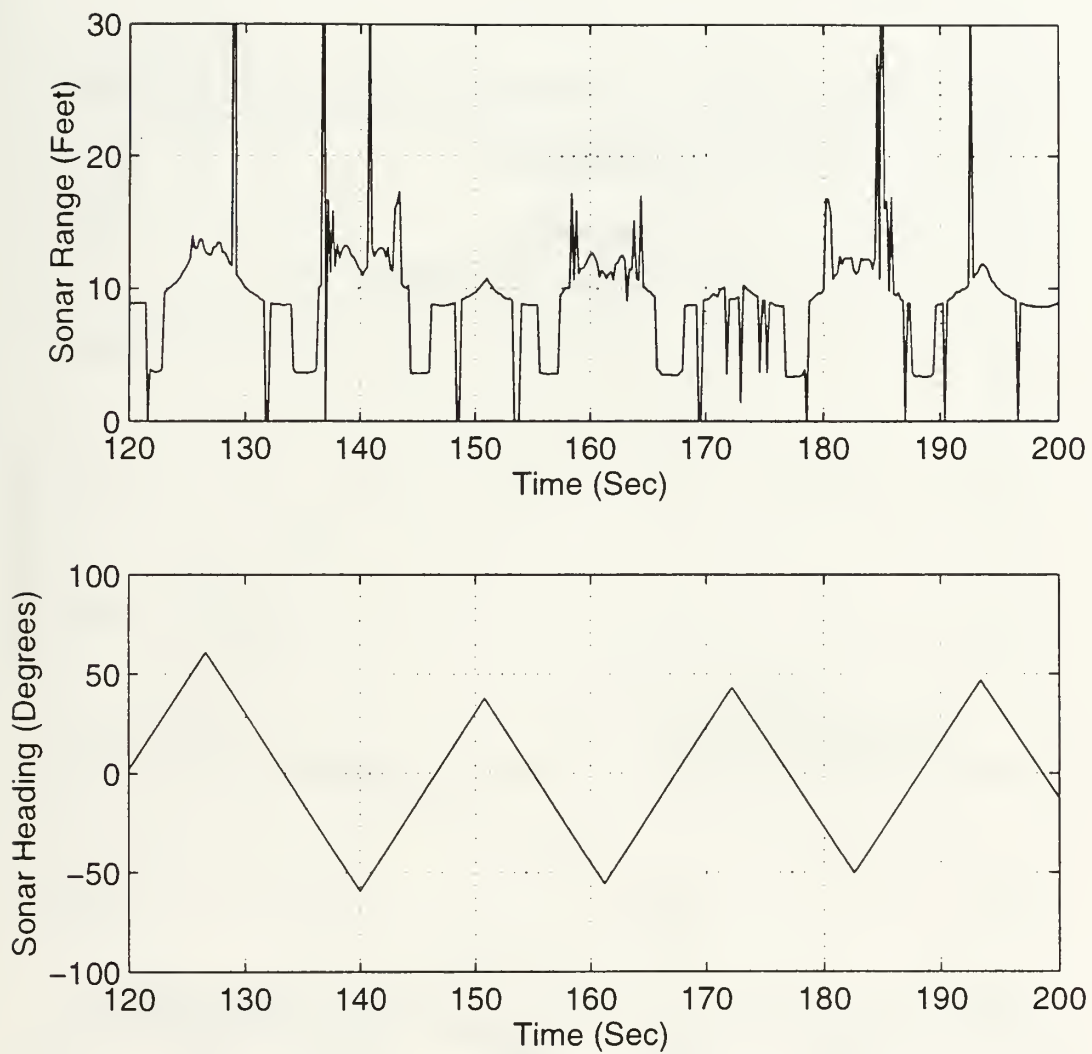
Figure 6.13 Sonar Range Returns and Heading Angle vs. Time (Experimental).

279

Figure 6.14 Stern Screw Control Voltage and Position Response vs. Time.

Figure 6.15 Comparison of Force Lag Between Simulation and Experimental vs. Time.

Figure 6.16 Sonar Target Tracking Method.

Figure 6.17 Sonar with Model Control Block Diagram.

Figure 6.18 Position Response vs. Time (Simulation).

284
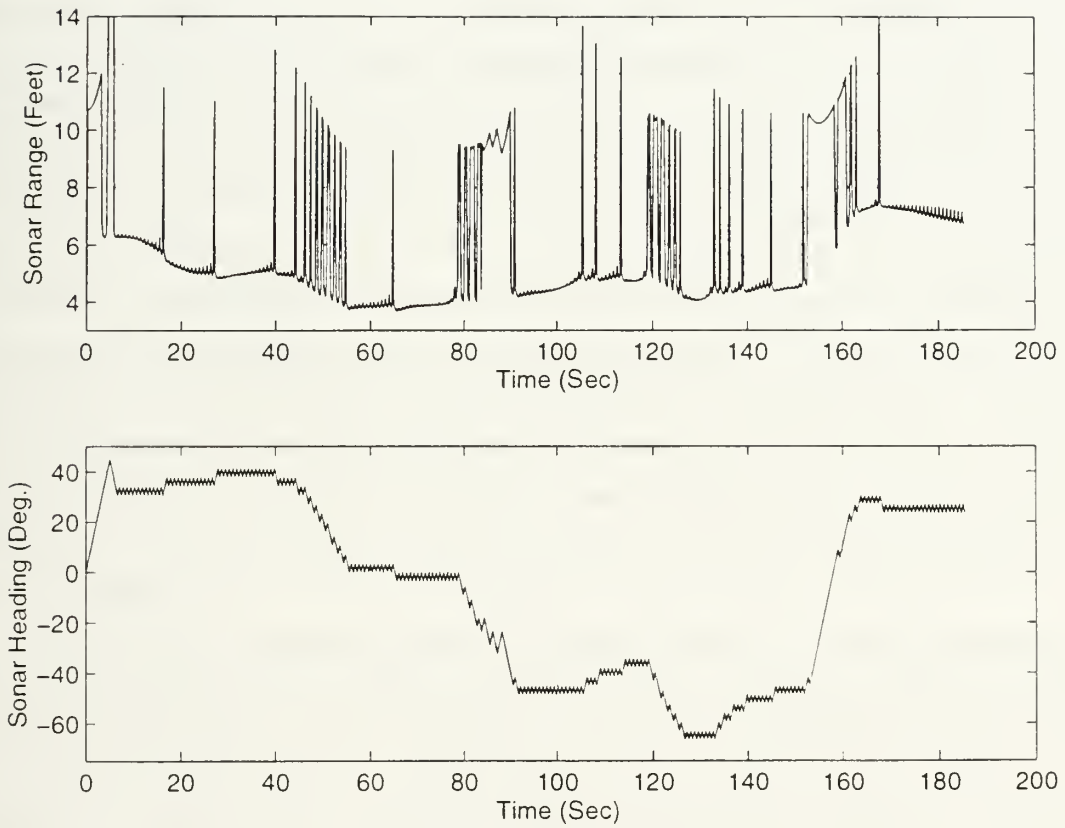
Figure 6.19 Sonar Range Returns and Heading Angle vs. Time (Simulation).
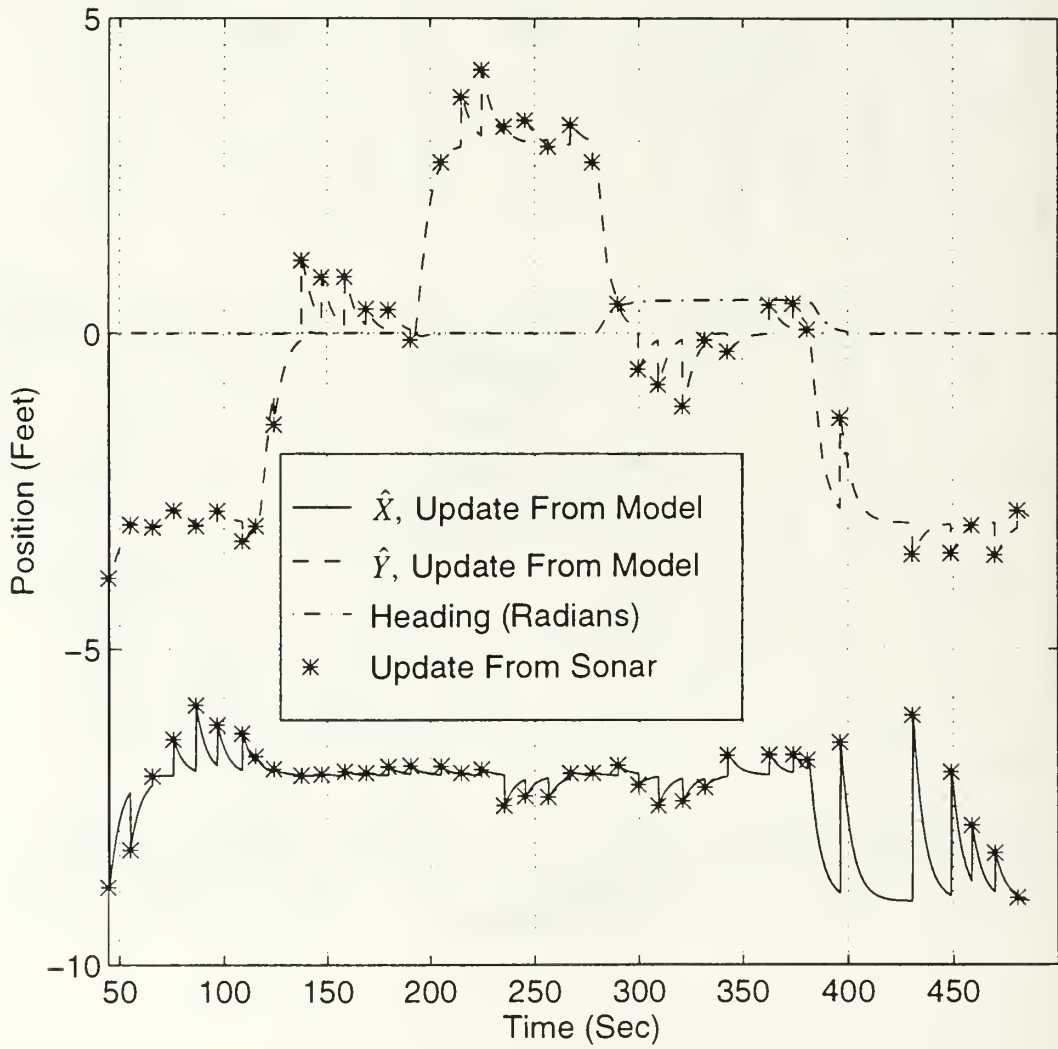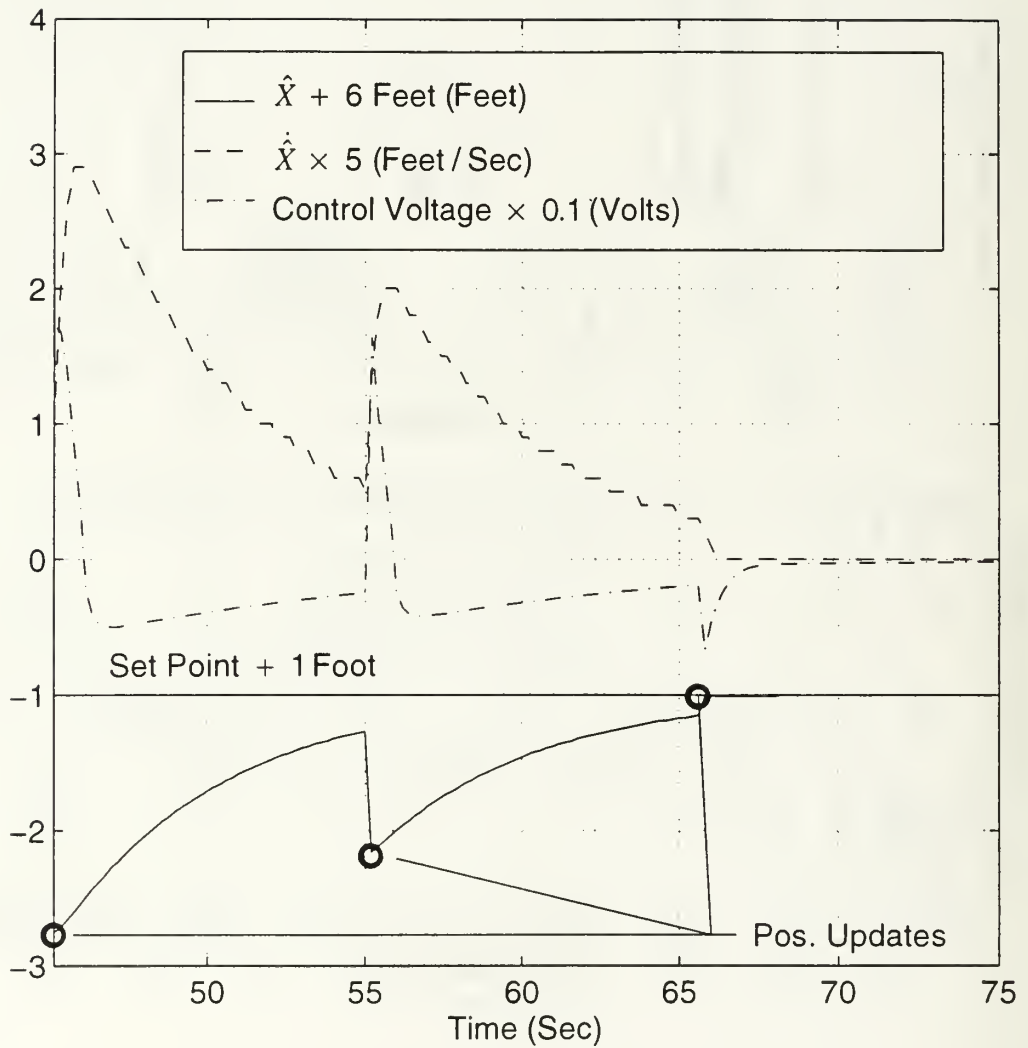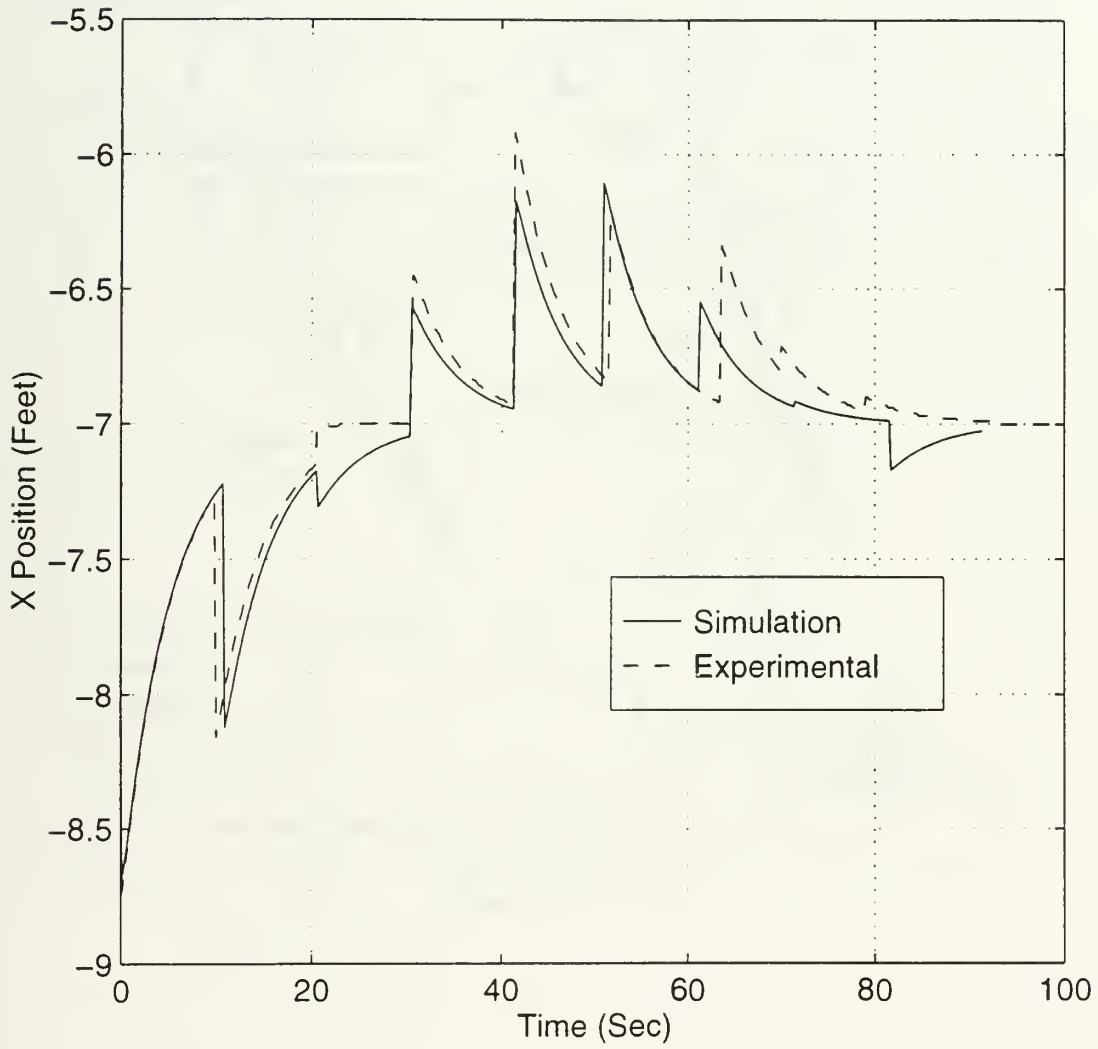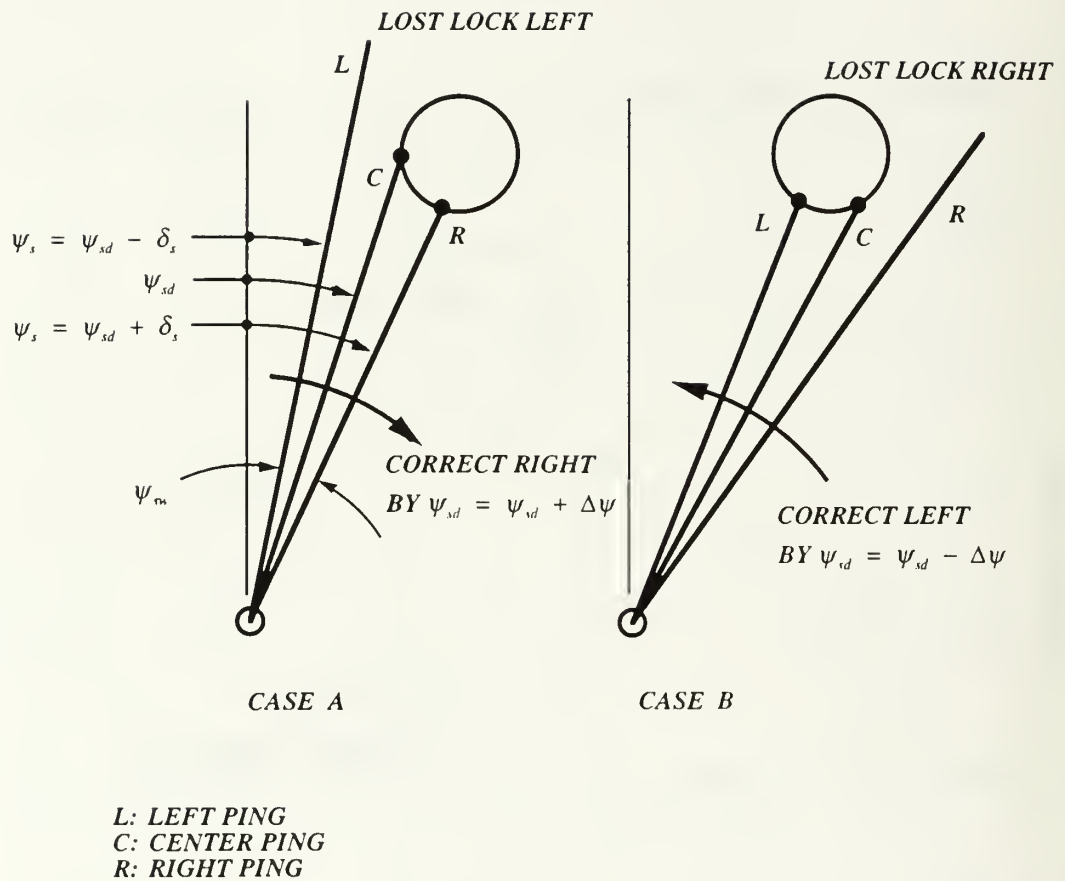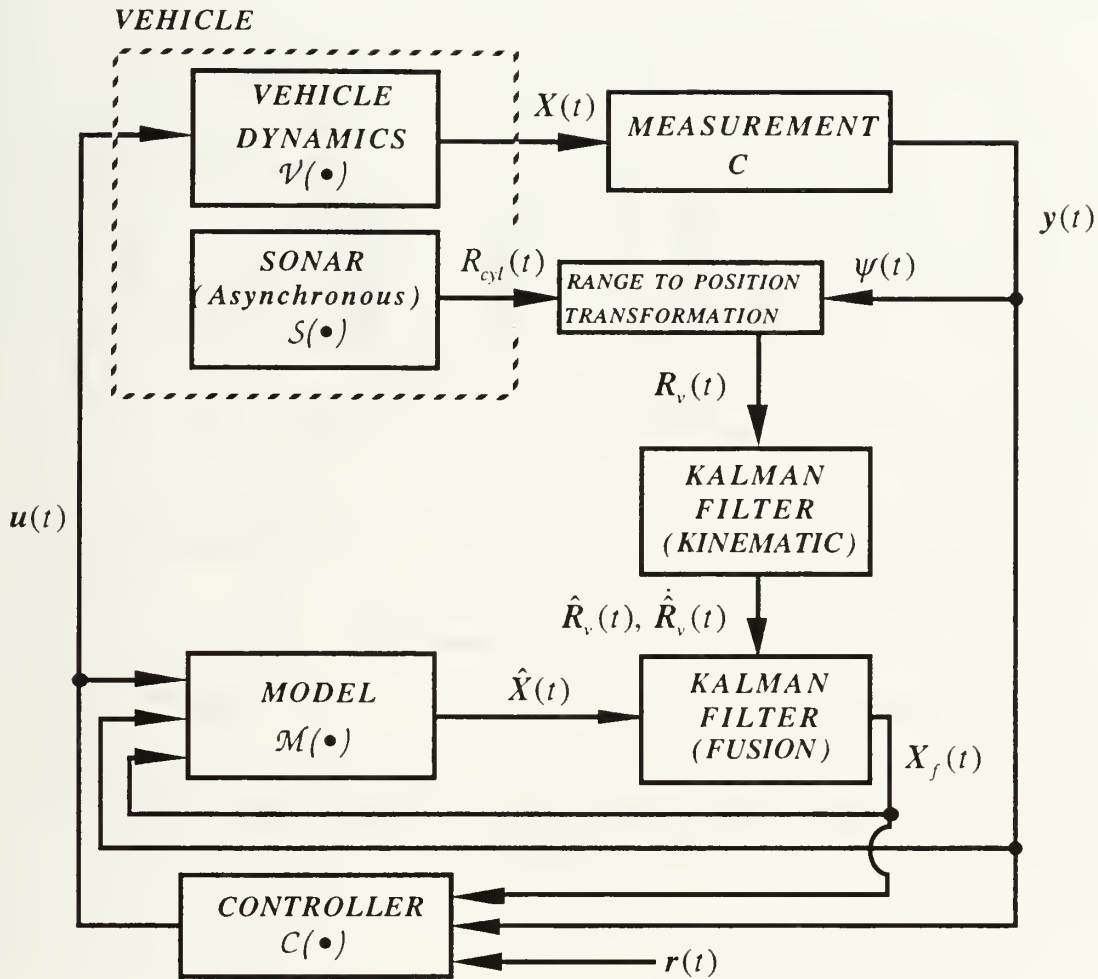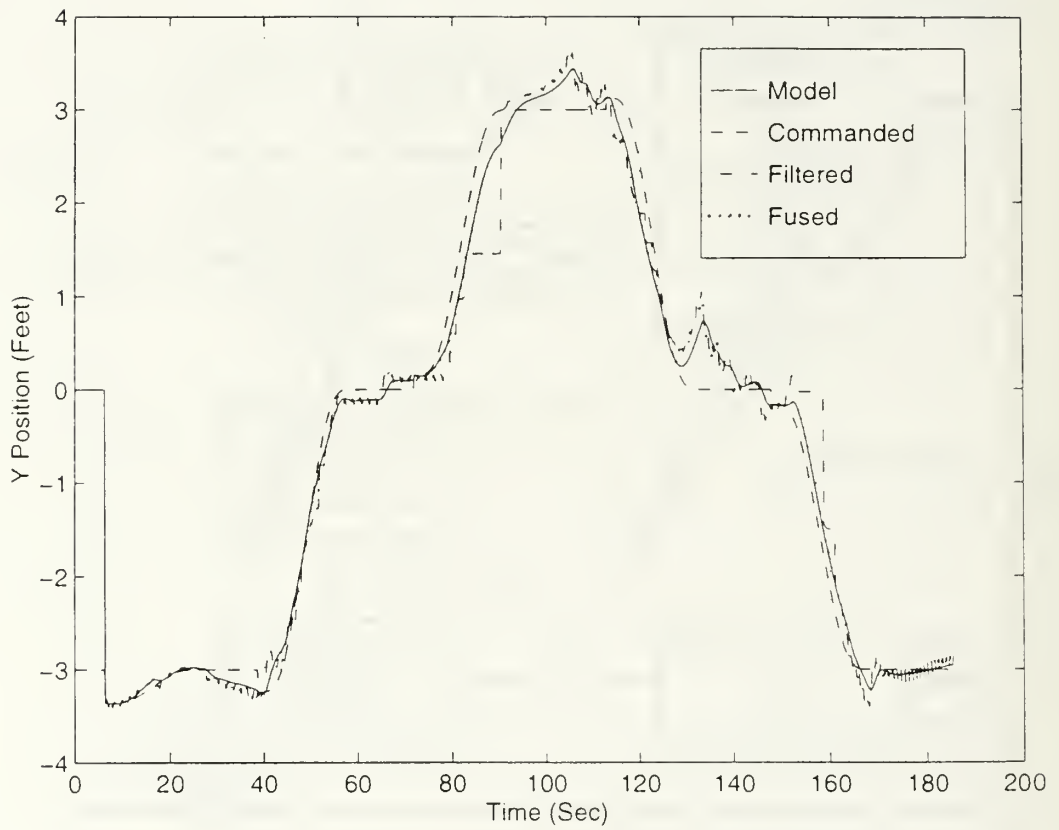
285

VII. CONCLUSIONS AND RECOMMENDATIONS

The main purpose of this work has been to develop, demonstrate, and validate an open architecture, three level control system for the real time control of an autonomous underwater vehicle, (AUV). As applied to underwater vehicles, the three levels have been named the Strategic, Tactical, and Execution levels. At the top is the Strategic Level It is an asynchronous, discrete event system managing the progress of the mission phases, and implemented in the rule based language, Prolog. Below this is the Tactical Level, written in the language C, which computes data necessary to coordinating the control modes required for each mission phase and is also asynchronous. Finally, the C language Execution Level performs the synchronous motion control of the vehicle. Since sequencing mission phases is inherently an asynchronous process, and operates with longer time scales than the synchronous servo controllers, there is a convenient separation to run them using different processors. The natural division of the synchronous and asynchronous tasks maps well into the separation of symbolic and numeric operations and is well suited to multiple computers. With this flexibility, each system may be chosen based on their particular attributes with regard to computer languages, operating systems, timing constraints, and processor speeds.

Combining the theories of discrete and continuous time control techniques, the Hybrid control system allows multiple task robot behaviors to be accomplished. Particular behaviors demonstrated included submerging, heading, and longitudinal control, along with local area positioning of the vehicle using acoustic sensors. Each control scenario described above was subject to simulation studies followed by experimental verification using the NPS Phoenix vehicle.

A. SUMMARY OF CONCLUSIONS

1. Chapter II Conclusions

Results from Chapter II have shown that the formulation for a MIMO Sliding Mode controller performs very well in the slow speed maneuvers supporting robotic behaviors. The control method includes the use of a weighted minimum norm for the solution of the inverse dynamics. Flexibility in adjustment of control effort between multiple redundant

actuators is thereby available and supports automated control reconfiguration for enhanced reliability. The simulations have also shown that adequate control authority is needed when performing path tracking maneuvers. So far the weights for the inverse solution of the input gain matrix have been either all unity or some unity and some zero. Further work is needed to formulate the weights for the control surfaces and thrusters to be a function of vehicle forward speed. Using this, it should be possible to avoid control saturation during any given maneuver.

A robustness analysis was also presented which included modeling inaccuracies in the mass, dynamics, and input gain parameters. The analysis provided a design procedure to ensure stability despite the uncertainties. The results of this work have emphasized the need for control designers to clearly understand the performance limits of the vehicle to be controlled.

Although, the control design methodology presented appears robust and well behaved in simulation, other real factors not included are lags in thruster response. If relatively long lags are present, these effects should be included in the general analysis.

2. Chapter III Conclusions

The conclusion of the work in Chapter III has indicated that complex behavior can be readily coordinated through Strategic Level rules, that are easily modified. Two kinds of predicates have been introduced. The first provides commands for particular vehicle action, while the second requests data for the evaluation of mission state transitions. Communication through Tactical Level software to the Execution Level controllers is a simple but convenient way of commanding stable responses of the vehicle. The design of well behaved control laws and functions at the Execution Level is essential and is accomplished through careful attention to the sliding mode control law specifications. Reactivity, failure recovery, and even human interfacing within the controller can take place at any level.

3. Chapter V Conclusions

The results of Chapter V have shown that the Phoenix can be precisely controlled in the test tank environment using thrusters. Development of sliding mode controllers for submergence, heading, and longitudinal control have been proven to be robust and have

shown to provide conveniently tuned, exceptional vehicle performance. The flexibility of the Strategic Level rules has allowed many different control scenarios to be quickly tested and evaluated without major code modifications. Using command generators provided extremely precise time based maneuvering and proved very effective when activated concurrently for multiple interacting control modes.

a. Submergence Control Studies

The results presented have shown that the depth of the Phoenix can be precisely controlled using Sliding Mode control of the vertical thrusters. Although discretization noise from the depth cell produced more control action than desired, the overall positioning performance is exceptional, and this problem could be rectified using a finer resolution A/D converter. Applying integral control has shown to be very effective in compensating for any deviation of the vehicle buoyancy from neutral, especially if activated at the proper time. Using command generators provided extremely precise, time based maneuvering, even in the presence of buoyancy disturbances.

b. Heading Control

The results of Sliding Mode heading control for the Phoenix using the lateral thrusters has been exceptional in both step response and command generator performance. The tendency of the lateral thrusters to chatter has not adversely affected the vehicle motions. Nevertheless, using post filtering and a dead band on the thruster input voltage lead to a significant reduction of this undesirable phenomena. The command generator performance was highly accurate, demonstrated by almost perfect tracking of the input profiles.

c. Longitudinal Motion Control

The results of sliding mode wall servoing control have shown highly satisfactory accuracy in longitudinal positioning of the vehicle. Using a Kalman filter modified to reject range outliers provided very accurate and stable signal conditioning from the ST1000 sonar data. A comparison between sliding mode and proportional derivative control has been presented which demonstrates the superiority of the non-linear approach.

Future work in this area could include the use of two sonars to enable both longitudinal and lateral positioning of the vehicle for more general control scenarios.

d. Coordinated Control

It has been shown that is a relatively easy task to program coordinated vehicle maneuvers since the generality of the Strategic Level rules allow the method of control to be determined by the Tactical Level. Accurate, simultaneous tracking of the command generator trajectories proved that the servo level controllers are currently very well tuned for the vehicle maneuvers shown.

4. Chapter VI Conclusions

The results of Chapter VI have shown that it is possible to navigate an underwater vehicle in a local area using an acoustic sensor for position information. The accuracy of the model used between updates is moderately satisfactory and can allow for time varying currents. However, some additional model adjustments could be made to compensate for the force lag in the longitudinal direction during transient thrust conditions. This undesirable effect could also be alleviated physically by the addition of shrouds around the stern screws which should bring the performance up to that of the lateral thrusters. While these results were taken in a tank environment, another improvement would be to fuse the model with an INS system in between sonar updates and then, at update time, to fuse that estimate with the sonar data. This would allow for better compensation of wave induced disturbances while retaining the positioning precision found. Since the sonars are mechanically scanned, and a delay of up to 10 seconds between position update is common, use of an electronically scanned or multi-beam sonars may be preferable although our experience to date has been that cross-talk between beams can be a serious problem.

To increase the response time, a continuous target tracking algorithm was simulated. This provided extremely favorable results in simulation but will need further modifications to successfully operate in the actual vehicle.

B. SUGGESTIONS FOR FUTURE WORK

Additional work is needed in several areas, and falls into three main categories: software improvements, upgrades of the vehicle hardware, and more sophisticated mission capabilities. One pressing need is to develop a graphical user interface (GUI) to provide an easier means of programming mission control. Such a system should include features to prevent attempts to design nonsensical missions or activation of inappropriate behaviors. The system should auto-generate executable code.

An expanded set of vehicle primitives needs to be developed to enhance vehicle capabilities, especially if new sensors or actuators are installed such as side-scan and Doppler sonar. Improved fault detection software should also be included as part of a Tactical Level engineering module for on-line diagnostic state of health monitoring. Replacement of the Tactical Level error filters with softer techniques for decision making should also be explored. Using elastic rather than crisp constraints allows decisions to be made much faster with varying levels of confidence as opposed to the more constrained and invariant filtering approach. Applying the above measures, will also enable reconfigurable servo level control schemes through automatic redistribution of control effort if certain actuators fail.

In the area of hardware improvements, implementation of radio Ethernet into the vehicle will allow tetherless remote communications while surfaced. To communicate with the vehicle while submerged, recent advances in underwater acoustic data transmission holds great promise and preliminary results using this technology have been positive. Using distributed processing in the Execution Level can allow a much better balance of computational requirements. For example, relegating the signal processing tasks to a dedicated processor associated with each sensor will significantly reduce the computational load on the controlling computer. For operations in the ocean environment, increased vehicle speed and endurance will be required, and can be achieved by using more powerful propulsion systems along with readily available, higher energy density batteries.

One of several missions that need to be investigated is the automatic garaging and re-powering underwater. While current studies funded by ONR in the AOSN program are aimed at addressing this problem, the design of capture mechanisms, precision homing, and devices for power and data transfer are in need of further research.

It is important to evaluate the capabilities of the control architecture for performing simulated mine countermeasures missions while operating in the shallow water ocean environment. In particular, it will be important to evaluate the influence of waves on sonar image distortion and on the ability of the control to stabilize the induced vehicle motions. An additional area for future research in shallow water environments will be to establish within the Tactical Level a machine leaning capability for on-line sea state estimation. Learning from the wave induced motions, control techniques for anticipatory compensation may assist in improving the precision of positioning.

APPENDIX A. EQUATIONS OF MOTION FOR THE NPS PHOENIX

The equations of motion and parameter values used to simulate the dynamic behavior of the NPS Phoenix is given in this appendix.

A.   PHYSICAL PARAMETERS

W,   Vehicle Weight   =   435.0 *lbs*

B,   Vehicle Buoyancy   =   435.0 *lbs*

l,   Characteristic length   =   7.30 *ft*

$$I_{xx} = 2.7 \; lb\text{-}ft\text{-}sec^2 \quad I_{yy} = 42.0 \; lb\text{-}ft\text{-}sec^2 \quad I_{zz} = 45.0 \; lb\text{-}ft\text{-}sec^2$$

$$I_{xy} = 0.0 \; lb\text{-}ft\text{-}sec^2 \quad I_{yz} = 0.00 \; lb\text{-}ft\text{-}sec^2 \quad I_{xz} = 0.00 \; lb\text{-}ft\text{-}sec^2$$

x_{bvt},   Bow Vertical Thruster Offset from C. G.   =   1.420 *ft*

x_{svt},   Stern Vertical Thruster Offset from C. G.   =   -1.420 *ft*

x_{blt},   Bow Vertical Thruster Offset from C. G.   =   1.920 *ft*

x_{slt},   Stern Vertical Thruster Offset from C. G.   =   -1.920 *ft*

y_{ls},   Left Screw Offset from C. G.   =   -0.330 *ft*

y_{rs},   Right Screw Offset from C. G.   =   0.330 *ft*

x_G, x Coordinate of C. G. From Body-Fixed Origin   =   0.010 *ft*

y_G, y Coordinate of C. G. From Body-Fixed Origin   =   0.000 *ft*

z_G, z Coordinate of C. G. From Body-Fixed Origin   =   0.042 *ft*

x_B, x Coordinate of C. B. From Body-Fixed Origin   =   0.010 *ft*

y_B, y Coordinate of C. B. From Body-Fixed Origin   =   0.000 *ft*

z_B, z Coordinate of C. B. From Body-Fixed Origin   =   0.000 *ft*

B. CONTROL INPUTS

F_{ls} Left Screw Force (lbs)

F_{rs} Right Screw Force (lbs)

F_{blt} Bow Lateral Thruster Force (lbs)

F_{slt} Stern Lateral Thruster Force (lbs)

F_{bvt} Bow Vertical Thruster Force (lbs)

F_{svt} Stern Vertical Thruster Force (lbs)

δ_{br} Bow Rudder Deflection (rad)

δ_{sr} Stern Rudder Deflection (rad)

δ_{bp} Bow Plane Deflection (rad)

δ_{sp} Stern Plane Deflection (rad)

C. NON-DIMENSIONALIZED HYDRODYNAMIC COEFFICIENTS

Surge Hydrodynamic Coefficients:

$$X'_{pp} = 0.0 \qquad X'_{pr} = 0.0 \qquad X'_{vp} = 0.0 \qquad X'_{q\delta_{sp}} = 0.0 \qquad X'_{vv} = -0.01743$$

$$X'_{qq} = 0.0 \qquad X'_{u} = -0.00282 \qquad X'_{vr} = 0.0 \qquad X'_{r\delta_{br}} = 0.0 \qquad X'_{ww} = 0.0$$

$$X'_{rr} = -0.00753 \qquad X'_{wq} = 0.0 \qquad X'_{q\delta_{bp}} = 0.0 \qquad X'_{r\delta_{sr}} = 0.0 \qquad X'_{\delta_r\delta_r} = -0.01018$$

$$X'_{\delta_{sp}\delta_{sp}} = -0.01018 \qquad X'_{\delta_{bp}\delta_{bp}} = -0.01018 \qquad X_{res} = -0.4024$$

Sway Hydrodynamic Coefficients:

$$Y'_{p} = 0.0 \qquad Y'_{v} = -0.03430 \qquad Y'_{wp} = 0.0 \qquad Y'_{\delta sr} = 0.01241$$

$$Y'_{r} = -0.00178 \qquad Y'_{p} = 0.0 \qquad Y'_{wr} = 0.0 \qquad Y'_{\delta br} = 0.01241$$

$$Y'_{pq} = 0.0 \qquad Y'_{r} = 0.01187 \qquad Y'_{v} = -0.10700$$

$$Y'_{qr} = 0.0 \qquad Y'_{vq} = 0.0 \qquad Y'_{vw} = 0.0$$

Heave Hydrodynamic Coefficients:

$$Z'_{\dot{q}} = -0.00253 \quad Z'_{\dot{w}} = -0.09340 \quad Z'_{w} = -0.78440$$
$$Z'_{pp} = 0.0 \quad\quad Z'_{q} = -0.07013 \quad Z'_{vv} = 0.0$$
$$Z'_{pr} = 0.0 \quad\quad Z'_{vp} = 0.0 \quad\quad Z'_{\delta sp} = -0.02110$$
$$Z'_{rr} = 0.0 \quad\quad Z'_{vr} = 0.0 \quad\quad Z'_{\delta bp} = -0.02110$$

Roll Hydrodynamic Coefficients:

$$K'_{\dot{p}} = -0.00024 \quad K'_{\dot{v}} = 0.0 \quad\quad K'_{wp} = 0.0$$
$$K'_{\dot{r}} = 0.0 \quad\quad K'_{p} = -0.00540 \quad K'_{wr} = 0.0$$
$$K'_{pq} = 0.0 \quad\quad K'_{r} = 0.0 \quad\quad K'_{v} = 0.0$$
$$K'_{qr} = 0.0 \quad\quad K'_{vq} = 0.0 \quad\quad K'_{vw} = 0.0$$

Pitch Hydrodynamic Coefficients:

$$M'_{\dot{q}} = -0.00625 \quad M'_{\dot{w}} = -0.00253 \quad M'_{w} = 0.05122$$
$$M'_{pp} = 0.0 \quad\quad M'_{q} = -0.01530 \quad M'_{vv} = 0.0$$
$$M'_{pr} = 0.0 \quad\quad M'_{vp} = 0.0 \quad\quad M'_{\delta sp} = -1.7664$$
$$M'_{rr} = 0.0 \quad\quad M'_{vr} = 0.0 \quad\quad M'_{\delta bp} = 1.3260$$

Yaw Hydrodynamic Coefficients:

$$N'_{\dot{p}} = 0.0 \quad\quad N'_{\dot{v}} = -0.00178 \quad N'_{wp} = 0.0 \quad\quad N'_{\delta sr} = -1.7663$$
$$N'_{\dot{r}} = -0.00047 \quad N'_{p} = 0.0 \quad\quad N'_{wr} = 0.0 \quad\quad N'_{\delta br} = 1.3259$$
$$N'_{pq} = 0.0 \quad\quad N'_{r} = -0.00390 \quad N'_{v} = -0.00769$$
$$N'_{qr} = 0.0 \quad\quad N'_{vq} = 0.0 \quad\quad N'_{vw} = 0.0$$

Mass Matrix:

$$
M = \begin{bmatrix}
m - X_{\dot{u}} & 0 & 0 & 0 & mz_G & -my_G \\
0 & m - Y_{\dot{v}} & 0 & -\left(mz_G + Y_{\dot{p}}\right) & 0 & mx_G - Y_{\dot{r}} \\
0 & 0 & m - Z_{\dot{w}} & my_G & -\left(mx_G + Z_{\dot{q}}\right) & 0 \\
0 & -\left(mz_G + K_{\dot{v}}\right) & my_G & I_{xx} - K_{\dot{p}} & -I_{xy} & -\left(I_{xz} + K_{\dot{r}}\right) \\
mz_G & 0 & -\left(mx_G + M_{\dot{w}}\right) & -I_{xy} & I_{yy} - M_{\dot{q}} & -I_{yz} \\
-my_G & mx_G - N_{\dot{v}} & 0 & -\left(I_{xz} + N_{\dot{p}}\right) & -I_{yz} & I_{zz} - N_{\dot{r}}
\end{bmatrix}
$$

$$
M^{\cdot} = \begin{bmatrix}
m - X_{\dot{u}} & 0 & 0 & mz_G & -my_G & 0 \\
0 & m - Y_{\dot{v}} & 0 & 0 & mx_G - Y_{\dot{r}} & -\left(mz_G + Y_{\dot{p}}\right) \\
0 & 0 & m - Z_{\dot{w}} & -\left(mx_G + Z_{\dot{q}}\right) & 0 & my_G \\
mz_G & 0 & -\left(mx_G + M_{\dot{w}}\right) & I_{yy} - M_{\dot{q}} & -I_{yz} & -I_{xy} \\
-my_G & mx_G - N_{\dot{v}} & 0 & -I_{yz} & I_{zz} - N_{\dot{r}} & -\left(I_{xz} + N_{\dot{p}}\right) \\
0 & -\left(mz_G + K_{\dot{v}}\right) & my_G & -I_{xy} & -\left(I_{xz} + K_{\dot{r}}\right) & I_{xx} - K_{\dot{p}}
\end{bmatrix}
$$

Input Gain Matrix:

$$
g = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
u|u|Y_{\delta br} & 0 & u|u|Y_{\delta sr} & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & u|u|Z_{\delta bp} & 0 & u|u|Z_{\delta sp} & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & u|u|M_{\delta bp} & 0 & u|u|M_{\delta sp} & 0 & 0 & -x_{bvt} & 0 & -x_{svt} & 0 \\
u|u|N_{\delta br} & 0 & u|u|N_{\delta sr} & 0 & -y_{ls} & -y_{rs} & 0 & x_{blt} & 0 & x_{slt}
\end{bmatrix}
$$

$$g^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ u|u|Y_{\delta br} & 0 & u|u|Y_{\delta sr} & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & u|u|Z_{\delta bp} & 0 & u|u|Z_{\delta sp} & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & u|u|M_{\delta bp} & 0 & u|u|M_{\delta sp} & 0 & 0 & -x_{bvt} & 0 & -x_{svt} & 0 \\ u|u|N_{\delta br} & 0 & u|u|N_{\delta sr} & 0 & -y_{ls} & -y_{rs} & 0 & x_{blt} & 0 & x_{slt} \end{bmatrix}$$

D.   EQUATIONS OF MOTION

Surge Motion Equation:

$$m[\dot{u} - vr + wq - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(pr + \dot{q})]$$

$$= \frac{\rho}{2}l^4\left[X'_{pp}p^2 + X'_{qq}q^2 + X'_{rr}r^2 + X'_{pr}pr\right]$$

$$+ \frac{\rho}{2}l^3\left[X'_{\dot{u}}\dot{u} + X'_{wq}wq + X'_{vp}vp + X'_{vr}vr + uq(X'_{q\delta_{bp}}\delta_{bp} + X'_{q\delta_{sp}}\delta_{sp})\right.$$

$$\left. + ur(X'_{r\delta_{br}}\delta_{br} + X'_{r\delta_{sr}}\delta_{sr})\right]$$

$$+ \frac{\rho}{2}l^2\left[X'_{vv}v^2 + X'_{ww}w^2 + u|u|(X'_{\delta_r\delta_r}(\delta_{sr}^2 + \delta_{br}^2) + X'_{\delta_{sp}\delta_{sp}}\delta_{sp}^2 + X'_{\delta_{bp}\delta_{bp}}\delta_{bp}^2)\right]$$

$$- (W - B)\sin\theta + F_{ls} + F_{rs} + X_{res}u|u|$$

Sway Motion Equation:

$$m[\dot{v} + ur - wp + x_G(pq + \dot{r}) - y_G(p^2 + r^2) + z_G(qr - \dot{p})]$$

$$= \frac{\rho}{2}l^4\left[Y'_{\dot{p}}\dot{p} + Y'_{\dot{r}}\dot{r} + Y'_{pq}pq + Y'_{qr}qr\right]$$

$$+ \frac{\rho}{2}l^3\left[Y'_{\dot{v}}\dot{v} + Y'_{p}up + Y'_{r}ur + Y'_{vq}vq + Y'_{wp}wp + Y'_{wr}wr\right]$$

$$+ \frac{\rho}{2}l^2\left[Y'_{v}uv + Y'_{vw}vw + u|u|(Y'_{\delta sr}\delta_{sr} + Y'_{\delta br}\delta_{br})\right]$$

$$+ \frac{\rho}{2}\int_{x_{tail}}^{x_{nose}}\left[C_{dy}h(x)(v + xr)^2 + C_{dz}b(x)(w + xq)^2\right]\frac{(v + xr)}{U_{cf}(x)}dx$$

$$+ (W - B)\cos\theta\sin\phi + F_{blt} + F_{slt}$$

Heave Motion Equation:

$$m[\dot{w} - uq + vp + x_G(pr - \dot{q}) + y_G(qr + \dot{p}) - z_G(p^2 + q^2)]$$

$$= \frac{\rho}{2}l^4\left[Z'_{\dot{q}}\dot{q} + Z'_{pp}p^2 + Z'_{pr}pr + Z'_{rr}r^2\right]$$

$$+ \frac{\rho}{2}l^3\left[Z'_{\dot{w}}\dot{w} + Z'_{q}uq + Z'_{vp}vp + Z'_{vr}vr\right]$$

$$+ \frac{\rho}{2}l^2\left[Z'_{vv}v^2 + Z'_{w}uw + u|u|(Z'_{\delta sp}\delta_{sp} + Z'_{\delta bp}\delta_{bp})\right]$$

$$- \frac{\rho}{2}\int_{x_{tail}}^{x_{nose}}\left[C_{dy}h(x)(v + xr)^2 + C_{dz}b(x)(w - xq)^2\right] \frac{(w - xq)}{U_{cf}(x)}dx$$

$$+ (W - B)cos\phi cos\theta + F_{bvt} + F_{svt}$$

Roll Motion Equation:

$$I_x\dot{p} + (I_z - I_y)qr + I_{xy}(pr - \dot{q}) - I_{yz}(q^2 - r^2) - I_{xz}(pq + \dot{r})$$

$$= \frac{\rho}{2}l^5\left[K'_p\dot{p} + K'_r\dot{r} + K'_{pq}pq + K'_{qr}qr\right]$$

$$+ \frac{\rho}{2}l^4\left[K'_{\dot{v}}\dot{v} + K'_p up + K'_r ur + K'_{vq}vq + K'_{wp}wp + K'_{wr}wr\right]$$

$$+ \frac{\rho}{2}l^3\left[K'_v uv + K'_{vw}vw\right]$$

$$+ m[y_G(\dot{w} - uq + vp) - z_G(\dot{v} + ur - wp)]$$

$$+ (y_G W - y_B B)cos\phi cos\theta - (z_G W - z_B B)cos\theta sin\phi$$

Pitch Motion Equation:

$$I_y \dot{q} + (I_x - I_z)pr - I_{xy}(qr + \dot{p}) + I_{yz}(pq - \dot{r}) + I_{xz}(p^2 - r^2)$$
$$- m[x_G(\dot{w} - uq + vp) - z_G(\dot{u} - vr + wq)]$$
$$= \frac{\rho}{2}l^5\left[M'_{\dot{q}}\dot{q} + M'_{pp}p^2 + M'_{pr}pr + M'_{rr}r^2\right]$$
$$+ \frac{\rho}{2}l^4\left[M'_{\dot{w}}\dot{w} + M'_q uq + M'_{vp}vp + M'_{vr}vr\right]$$
$$+ \frac{\rho}{2}l^3\left[M'_{vv}v^2 + M'_w uw + u|u|(M'_{\delta sp}\delta_{sp} + M'_{\delta bp}\delta_{bp})\right]$$
$$- \frac{\rho}{2}\int_{x_{tail}}^{x_{nose}}\left[C_{dy}h(x)(v + xr)^2 + C_{dz}b(x)(w - xq)^2\right]\frac{(w - xq)}{U_{cf}(x)}dx$$
$$- (x_G W - x_B B)\cos\phi\cos\theta - (z_G W - z_B B)\sin\theta - x_{bvt}F_{bvt} - x_{svt}F_{svt}$$

Yaw Motion Equation:

$$I_z \dot{r} + (I_y - I_x)pq - I_{xy}(p^2 - q^2) - I_{yz}(pr + \dot{q}) + I_{xz}(qr - \dot{p})$$
$$+ m[x_G(\dot{v} + ur - wp) - y_G(\dot{u} - vr + wq)]$$
$$= \frac{\rho}{2}l^5\left[N'_{\dot{p}}\dot{p} + N'_{\dot{r}}\dot{r} + N'_{pq}pq + N'_{qr}qr\right]$$
$$+ \frac{\rho}{2}l^4\left[N'_{\dot{v}}\dot{v} + N'_p up + N'_r ur + N'_{vq}vq + N'_{wp}wp + N'_{wr}wr\right]$$
$$+ \frac{\rho}{2}l^3\left[N'_v uv + N'_{vw}vw + u|u|(N'_{\delta sr}\delta_{sr} + N'_{\delta br}\delta_{br})\right]$$
$$- \frac{\rho}{2}\int_{x_{tail}}^{x_{nose}}\left[C_{dy}h(x)(v + xr)^2 + C_{dz}b(x)(w - xq)^2\right]\frac{(v + xr)}{U_{cf}(x)}dx$$
$$- (x_G W - x_B B)\sin\phi\cos\theta - (y_G W - y_B B)\sin\theta$$
$$+ x_{blt}F_{blt} + x_{slt}F_{slt} - y_{ls}F_{ls} - y_{rs}F_{rs}$$

Euler Angle Rates and Global Positions:

$$\dot{X} = u_c + u\cos\psi\cos\theta + v[\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi]$$
$$+ w[\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi]$$

$$\dot{Y} = v_c + u\sin\psi\cos\theta + v[\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi]$$
$$+ w[\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi]$$

$$\dot{Z} = w_c - u\sin\theta + v\cos\theta\sin\phi + w\cos\theta\cos\phi$$

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta$$

$$\dot{\theta} = q\cos\phi - r\sin\phi$$

$$\dot{\psi} = \frac{(q\sin\phi + r\cos\phi)}{\cos\theta}$$

Cross-flow Velocity:

$$U_{cf}(x) = \sqrt{[(v + xr)^2 + (w - xq)^2]}$$

APPENDIX B. COMMAND GENERATORS

To control the positional motions of an underwater vehicle, position, velocity, and acceleration command generators may be required. In the one-dimensional case the commands are obtained from the specification of a desired final zero velocity and acceleration position given an initial zero velocity and acceleration position. The elapsed time to complete the maneuver, T_f, is determined from the desired distance to travel which is the difference between the initial position, s_0, and the final, s_f, along with $\dot{s}_{max}$ and $\ddot{s}_{max}$, the maximum allowed vehicle velocity, and acceleration respectively. These values also determine the duration of T_{cv}, the time of constant velocity between acceleration and deceleration. To avoid any discontinuities in either of the command arrays, the acceleration curve must be at least third order, and gives a velocity curve of order 4 and a 5th order position curve as shown in Figure B.1.

Given a third order acceleration command equation of the form

$$\ddot{s}(t) = At^2 + Bt^3 \tag{B.1}$$

the coefficients A and B must be

$$A = \frac{3\ddot{s}_{max}}{\left(T_{\dot{s}_{max}}\right)^2} \qquad B = -\frac{2\ddot{s}_{max}}{\left(T_{\dot{s}_{max}}\right)^3} \tag{B.2}$$

to force a zero slope at $t = 0$ and $t = T_{\dot{s}_{max}}$. The duration of T_{cv} is calculated using

$$T_{cv} = \frac{1}{\dot{s}_{max}}\left(\left|s_f - s_0\right| - 2\frac{\dot{s}_{max}^2}{\ddot{s}_{max}}\right) \tag{B.3}$$

If $T_{cv} < 0$ and the value of $\left|s_f - s_0\right|$ is too small for the given specifications of $\dot{s}_{max}$ and $\ddot{s}_{max}$, this requires that $\dot{s}_{max}$ be reduced until the constant velocity interval is exactly 0 given by

$$\dot{s}_{max} = \sqrt{\frac{\left|s_f - s_0\right|\ddot{s}_{max}}{2}} \qquad T_{cv} = 0 \tag{B.4}$$

The elapsed time during acceleration to $\dot{s}_{max}$ is

$$T_{\dot{s}_{max}} = \frac{2\dot{s}_{max}}{\ddot{s}_{max}}, \tag{B.5}$$

and the value of T_f is determined.

If both the distance to travel and the total time to complete the maneuver is specified, the value of T_f must be of sufficient duration such that

$$T_f \geq \sqrt{8\frac{\left|s_f - s_0\right|}{\ddot{s}_{max}}}$$

To determine the smallest possible maximum acceleration, $\ddot{s}_{max}$, for a given time, it may be calculated using

$$\ddot{s}_{max} = 8\frac{\left|s_f - s_0\right|}{T_f^2} \tag{B.6}$$

which will produce a velocity curve with no region of constant velocity (i.e. $T_{cv} = 0$).

The position, velocity, and acceleration profiles for the five time intervals shown in Figure B.1 are given by

For $0 \leq t \leq \dfrac{T_{\dot{s}_{max}}}{2}$:

$$s(t) = s_0 + \alpha\left(\frac{A}{12}t^4 - \frac{B}{20}t^5\right)$$

$$\dot{s}(t) = \alpha\left(\frac{A}{3}t^3 - \frac{B}{4}t^4\right) \tag{B.7}$$

$$\ddot{s}(t) = \alpha\left(At^2 - Bt^3\right)$$

302

For $\dfrac{T_{\dot{s}_{max}}}{2} \le t \le T_{\dot{s}_{max}}$:

$$s(t) = s_0 + \alpha\left(\frac{A}{12}\left(T_{\dot{s}_{max}} - t\right)^4 - \frac{B}{20}\left(T_{\dot{s}_{max}} - t\right)^5 + \dot{s}_{max}t - \frac{\dot{s}_{max}^2}{\ddot{s}_{max}}\right)$$

$$\dot{s}(t) = \alpha\left(-\frac{A}{3}\left(T_{\dot{s}_{max}} - t\right)^3 + \frac{B}{4}\left(T_{\dot{s}_{max}} - t\right)^4 + \dot{s}_{max}\right) \tag{B.8}$$

$$\ddot{s}(t) = \alpha\left(A\left(T_{\dot{s}_{max}} - t\right)^2 - B\left(T_{\dot{s}_{max}} - t\right)^3\right)$$

For $T_{\dot{s}_{max}} \le t \le T_1$:

$$s(t) = s_0 + \alpha\left(\dot{s}_{max}\left(t - T_{\dot{s}_{max}}\right) + \frac{\dot{s}_{max}}{\ddot{s}_{max}}\right)$$

$$\dot{s}(t) = \alpha\dot{s}_{max} \tag{B.9}$$

$$\ddot{s}(t) = 0$$

For $T_1 \le t \le T_1 + \dfrac{T_{\dot{s}_{max}}}{2}$:

$$s(t) = s_0 + \alpha\left(-\frac{A}{12}\left(t - T_1\right)^4 + \frac{B}{20}\left(t - T_1\right)^5 + \dot{s}_{max}t - \frac{\dot{s}_{max}^2}{\ddot{s}_{max}}\right)$$

$$\dot{s}(t) = \alpha\left(-\frac{A}{3}\left(t - T_1\right)^3 + \frac{B}{4}\left(t - T_1\right)^4 + \dot{s}_{max}\right) \tag{B.10}$$

$$\ddot{s}(t) = \alpha\left(-A\left(t - T_1\right)^2 + B\left(t - T_1\right)^3\right).$$

303

For $T_1 + \dfrac{T_{\dot{s}_{max}}}{2} \leq t \leq T_f$:

$$s(t) = s_0 + \alpha\left(-\frac{A}{12}(T_f - t)^4 + \frac{B}{20}(T_f - t)^5 + \dot{s}_{max}T_{cv} + 2\frac{\dot{s}_{max}^2}{\ddot{s}_{max}}\right)$$

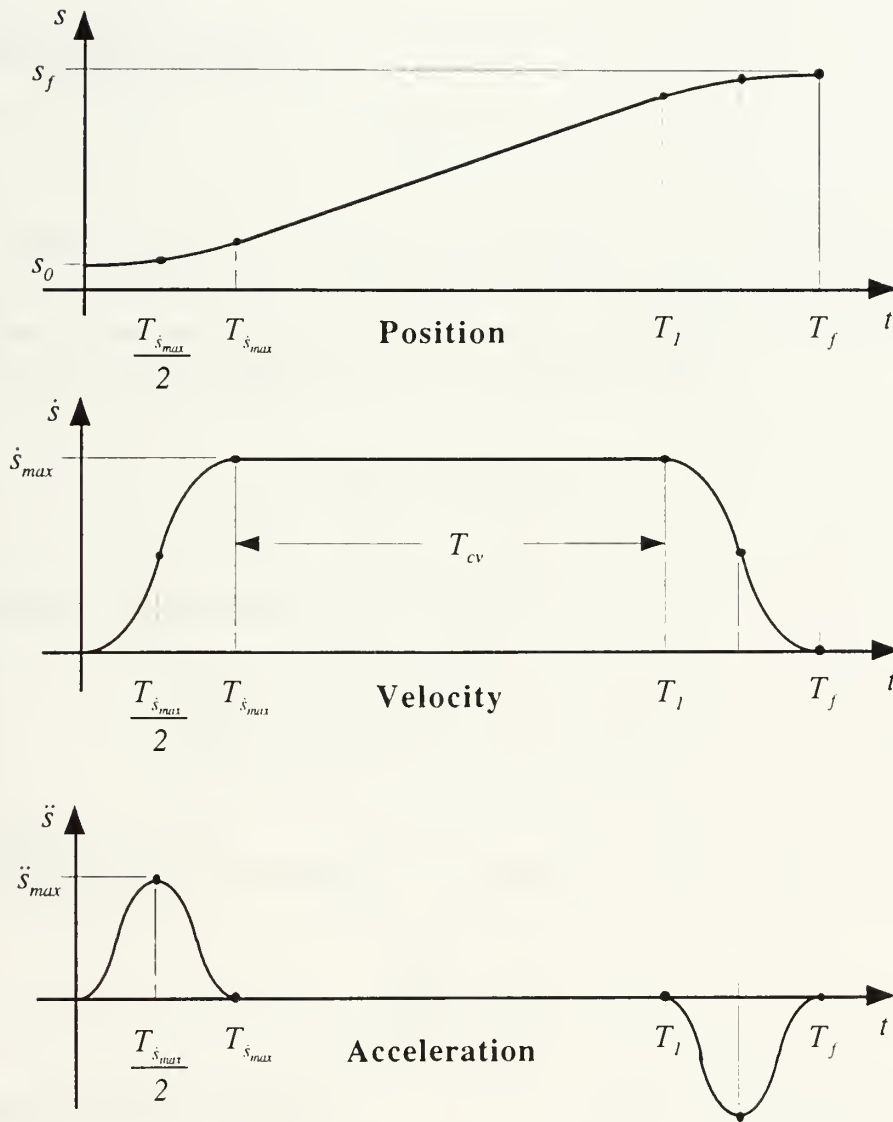$$\dot{s}(t) = \alpha\left(-\frac{A}{3}(T_f - t)^3 + \frac{B}{4}(T_f - t)^4\right) \qquad\qquad (B.11)$$

$$\ddot{s}(t) = \alpha\left(-A(T_f - t)^2 + B(T_f - t)^3\right)$$

where

$$T_1 = T_{\dot{s}_{max}} + T_{cv} \text{ and } T_f = 2T_{\dot{s}_{max}} + T_{cv},$$

and α is a sign coefficient based on

$$\alpha = sgn\left(s_f - s_0\right).$$

304

B.1 Command Generator Profiles for Position, Velocity, and Acceleration.

APPENDIX C. MINIMUM NORM SOLUTION

Given a system of equations

$$gu = f \tag{C.1}$$

where g is size $n \times m$, u is $m \times 1$, and f is $n \times 1$, and if $m > n$, an infinite number of solutions exist. The minimum norm solution can be derived using Lagrange multipliers and minimization of the norm of the solution vector, $\|u\|^2$, subject to the constraint $gu - f = 0$. This can be written in terms of a performance index as

$$J = \frac{1}{2}u^T u + \lambda(gu - f). \tag{C.2}$$

The minimum of J with respect to u can be found by

$$\frac{\partial J}{\partial u} = u + (\lambda g)^T = 0 = u + g^T \lambda^T \tag{C.3}$$

which occurs at $u = -g^T \lambda^T$.

The minimum of J with respect to λ is simply

$$\frac{\partial J}{\partial \lambda} = 0. \tag{C.4}$$

Substituting (C.3) into (C.1) yields

$$\lambda^T = -(gg^T)^{-1}f. \tag{C.5}$$

And substituting (C.5) into (C.3) provides the minimum norm solution

$$u = g^T(gg^T)^{-1}f. \tag{C.6}$$

The above treatment gives equal weighting to each u_i of the solution vector. This is not always desirable and a modified form of (C.6) can be derived which weights each u_i. Define a weighted solution vector

$$u^* = Wu \qquad \text{(C.7)}$$

where W is a diagonal weighting matrix of the form

$$W_{ii} = \frac{1}{u_{max_i}}. \qquad \text{(C.8)}$$

This implies u_i^* is a fraction of the maximum value of u_i. The weighted performance index becomes

$$J = u^{*T}u^* + \lambda(gu^* - f) = u^T W^T W u + \lambda(gWu - f) \qquad \text{(C.9)}$$

and

$$\frac{\partial J}{\partial u} = 0 = Wu + g^T \lambda^T \qquad \text{(C.10)}$$

The minimum of J occurs at $u = -W^{-1}g^T\lambda^T$

$$\frac{\partial J}{\partial \lambda} = 0 \qquad \text{(C.11)}$$

Substituting (C.10) into (C.1) yields

$$\lambda^T = -(gW^{-1}g^T)^{-1}f \qquad \text{(C.12)}$$

And substituting (C.12) into (C.10) provides the weighted minimum norm solution

$$u = W^{-1}g^T(gW^{-1}g^T)^{-1}f \qquad \text{(C.13)}$$

308

For diagonal W,

$$W_{ii}^{-1} = u_{max_i} \tag{C.14}$$

Therefore, to remove the contribution of any u_i from the solution vector, set it's u_{max} to zero in W^{-1}. To avoid division by zero in W, rewrite (C.13) as

$$u = \overline{W} g^T (g \overline{W} g^T)^{-1} f \tag{C.15}$$

where $\overline{W} = W^{-1}$.

APPENDIX D. NPS PHOENIX HARDWARE COMPONENTS

The following is a listing and description of the major hardware components currently onboard the NPS Phoenix vehicle.

A. SENSORS

Gyroscopes

To sense the vehicle roll, pitch and heading angular positions along with the three axis rates, three types of gyroscopes are used. All are manufactured Humphrey Inc. and are usually used for small aircraft/missile applications, and use a mechanical motor whose inertial angular momentum vector is fixed in a spatial direction unless acted upon by a torque.

Vertical Gyro:

This unit is used to measure the roll and pitch angle of the vehicle. The motor is gimbaled where the mechanical limits are 360° freedom of rotation about the roll axis and $\pm 80^o$ minimum freedom of rotation about the pitch axis. The output is ± 10 Vdc and is read by an A/D converter on the controlling computer. Model VG34-0301-2.

Rate Gyro:

This is a combined package of individual gimballess rate gyroscopes for measuring roll, pitch, and yaw rates. The range for roll rate is 360°/sec and pitch and yaw rate maximums are $\pm 90^o$/sec. All outputs are ± 10 Vdc and read by a A/D converter. Model RG02-2324-1.

Free Gyro

This unit is used to measure the heading angle of the vehicle. The angular range is 360° continuous and has a syncro output which is converted to 14 bit digital through the use of a converter chip from Analog Devices Inc. The digital data is read through two parallel ports on the controlling computer, 8 bits from one port and 6 from the other. The

gyroscope is also configured for remote cage/uncage of the gimbals from the computer. Model FG23-7102-1.

Depth Cell

The vehicle depth is measured using a differential pressure transducer with an operating range of 0 to 34 feet which translates to an analog signal output of 0 to 10 Vdc which is read by an A/D converter on the computer. The cell is located in the center of the forward bulkhead inside the flooded nose. The nose shields the probe from and undesirable flow effects from forward motion of the vehicle. The manufacturer is Psi-Tronix Inc., model S11-131.

Forward Speed Sensor (Turbo Probe)

The vehicle speed through the water is measured by a Turbo-Probe turbine flow meter, manufactured by Flow Technology, Inc. The transducer is an axial rotor mounted at the end of a strut which protrudes through the bottom of the nose. A cowling around the rotor houses a magnetic pick off which generates square wave electrical pulses at a frequency proportional to the rotor speed. The pulses are read by a timer card in the computer and converted to speed using the manufacturers supplied calibration.

Sonars

A single Datasonics PSA-900 Sonar Altimeter is mounted facing downward in the nose of the vehicle. This unit is used for depth above bottom measurements and emits a ten degree conical beam at a frequency of 210 kHz. The signal output is 0 - 10 Vdc proportional to the ranges detected up to approximately 90 feet.

DiveTracker

A short baseline acoustic positioning system called DiveTracker by Desert Star Systems is installed. The system consists of two surface transducers and one mounted to the vehicle. The unique capability of this system is that the vehicle location can be read by the onboard computer in addition to being tracked on the surface. The output from this unit

is read by the controlling computer through a serial link and processed for navigation. (Reimers, 1995, Scrivener, 1996, Zinni, 1995)

Global Positioning System

This unit is used for obtaining the location of the vehicle in global coordinates and is capable of standard and differential modes. Antennas for both are mounted on the top of the hull as shown in Figure 3.3 and are only usable while surfaced. The unit is connected to the Sun Voyager computer through a serial link for processing.

B.   GESPAC COMPUTER SYSTEM

The Gespac boards are referred to as "Euroboards" and each have dimensions of approximately 4.0 X 6.0 inches and 0.75 inches thick. The 12 boards are set into a 12 slot G-96 bus backplane and is powered with +5, +12, and -12 Vdc and shown in the installed positions in Figure D.1.

GESMPU-30H: Microprocessor Board

Motorola 32 bit 25 MHz 68030 microprocessor with 2.0 MBytes onboard CMOS dynamic Ram. Four EPROM's (Erasable Programmable Read Only Memory) chips are located on this board. Each have been programmed to hold essential modules for operation of the OS/9 operating system and configured to activate the TCP/IP network upon boot which enables ethernet connections from other computer systems to be made without relying on a console (serial) connection to the Gespac system. Once the system is up, any operating system modules or programs not in EPROM may be transferred to the system using ftp (File Transfer Program).

GESRAM 14B: RAM/EPROM/EEPROM Memory Module 2.0 MBytes.

2.0 MByte RAM card used for RAM disk mission data storage.

313

<u>GESSIO-1B:</u> Double Serial Interface Module.

Two port RS-232-C board. Used for serial communications with the ST725 and ST1000 sonars.

<u>GESPIA-3A:</u> Parallel Interface Board

Twin parallel input/output ports using TTL (0 to 5 Vdc) logic for enabling/disabling servo amplifier power relays for:

Port 0:

Bit PA0	-	Left screw
Bit PA1	-	Right screw
Bit PA2	-	Bow vertical thruster
Bit PA3	-	Bow lateral thruster
Bit PA4	-	Stern vertical thruster
Bit PA5	-	Stern lateral thruster

It is also used for sending signals to cage or uncage the free gyroscope and is also connected to the cage/uncage status lines.

Port 0:

Bit PB0	-	Cage free gyro
Bit PB3	-	Uncage free gyro

Port 1:

Bit PA0	-	Cage indicator
Bit PB0	-	Uncaged indicator

GESADA-1: Analog/Digital & Digital/Analog Subsystem (10 Bits)

A 16 channel, 10 bit analog/digital converter for:

Channel X	-	Computer battery voltage level indicator
Channel Y	-	Motor battery voltage level indicator
Channel 5	-	Fore leak detector
Channel 6	-	Aft leak detector

Circuitry for the fore/aft leak detectors are also connected to external LEDs for visual reference while the program is not running.

GESADC-2C: 16 Channel, 12 Bit Data Acquisition Module

A 16 channel 12 bit analog/digital converter for measurements of:

Channel 12	-	Roll angle gyroscope	± 10 Vdc
Channel 11	-	Pitch angle gyroscope	± 10 Vdc
Channel 9	-	Roll rate gyroscope	± 10 Vdc
Channel 8	-	Pitch rate gyroscope	± 10 Vdc
Channel 10	-	Yaw rate gyroscope	± 10 Vdc
Channel 7	-	Depth cell.	0 to 10 Vdc

GESTIM-1A: Multiple Timer Module (3 Modules)

Each module contains five 16 bit programmable timers (AM9513) for both event counting and frequency output. Three modules are used for:

Module 1:

Servo motor position control for:

Counter 1 (Output)	-	Top front rudder and bottom rear rudder
Counter 2 (Output)	-	Bottom front rudder and top rear rudder
Counter 3 (Output)	-	Left bow plane and right stern plane
Counter 4 (Output)	-	Right bow plane and left stern plane

315

The outputs are pulse width modulated signals which are proportional to the desired angular deflection of each servo motor.

Module 2:

Motor rotation rate measurement of:

    Counter 1 (Input) -   Bow vertical thruster
    Counter 2 (Input) -   Bow lateral thruster
    Counter 3 (Input) -   Stern vertical thruster
    Counter 4 (Input) -   Stern lateral thruster

Module 3:

Motor rotation rate measurement of:

    Counter 1 (Input) -   Left screw
    Counter 2 (Input) -   Right screw
    Counter 3 (Input) -   Turbo probe

The inputs are square wave pulse trains from each motor encoder or probe which is measured and converted to rotations per second using an appropriate calibration function in software.

EVLAN-11: Ethernet & Starlan Data Link Controller

Ethernet module for network communications between internal vehicle processors and external computers. An AUI (10Base5) to thin wire (10Base2) transceiver is attached to this unit.

<u>GESDAC-2B:</u> Digital/Analog Converter.

An 8 channel, 12 bit digital/analog converter module for input to servo amplifiers for control of:

Channel 0 - Left screw
Channel 1 - Right screw
Channel 2 - Bow vertical thruster
Channel 3 - Bow lateral thruster
Channel 4 - Stern vertical thruster
Channel 5 - Stern lateral Thruster

<u>GESMFI-1:</u> Multi-Function Interface

Contains two RS-232 serial ports and two parallel ports. 14 bits of the two parallel ports interface with the synchro to digital converter connected to the free gyroscope. 8 bits from parallel port 1 and 6 bits from port 2. One serial port is used to read position and message data from the vehicle DiveTracker unit.

<u>GESBUS-12M:</u> Interconnection Backplane for G-64 and G-96 Euroboards

Twelve slot Backplane for the boards listed above.

C.   ELECTRICAL COMPONENTS

<u>Batteries</u>

For the purposes of load balancing, two separate battery systems are used. Each system comprises of two 12 Vdc sealed lead acid batteries connected in parallel giving a total of 24 Vdc. One system powers the computer system, sonars, speed encoders, and control surface servo motors. The second system is used for the gyroscopes and propulsion motors. These have been separated since the computer is sensitive to voltage fluctuations which are caused by the propulsion motors. All batteries are rechargeable from an external power supply using a through-hull connection. The batteries are Panasonic Model LCL12V38P with nominal capacity of 38.0 Amp-hours.

ACON Power Supplies

The Gespac computer system is powered by a single ACON Model R100T2405-12TS power supply. It provides +5, +12, and -12 Vdc and is supplied by 24 Vdc directly from the computer battery system. A second supply is also installed for powering other units such as the DiveTracker module, GPS module, and ethernet transceiver for the Sun Voyager.
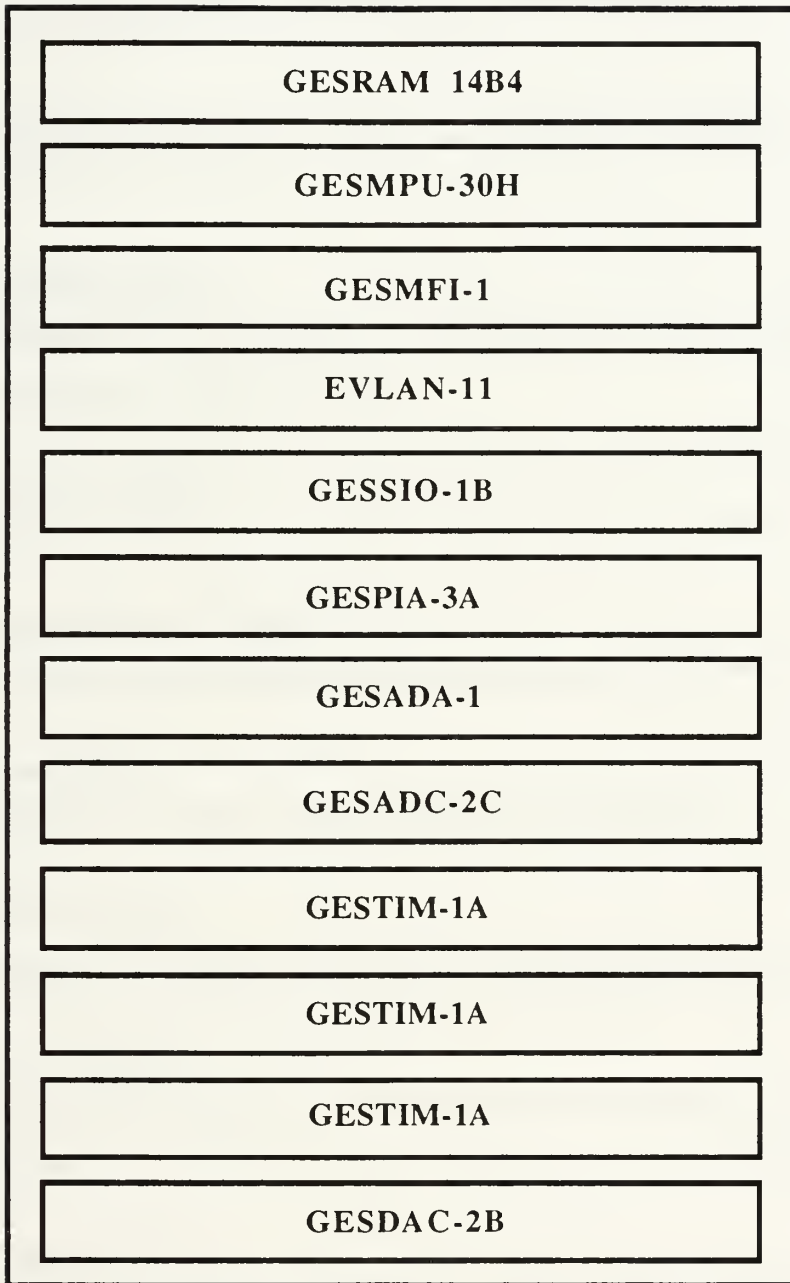
Calex Power Supplies

Calex Models 12S15, 48S15, and 12S5 power supplies provide either +5, +15, or -15 Vdc for the following units:

Reference source for the rate and vertical gyros ($\pm$ 15 Vdc)
Datasonics sonar (+15 Vdc)
Depth cell (+15 Vdc)
GESTIM-1A timer cards for control surface signal channels (+15 Vdc)
Control Surface servo motors (+5 Vdc)

Propulsion Motor Servo Amplifiers

Motor voltage control for the thrusters and stern screw motors is controlled through the use of Advanced Motion Controls PWM Model 30AD8DD servo amplifiers. One amplifier is used to control each motor and uses a 0 to 10 Vdc control signal to modulate the pulse width of a 24 Vdc, 5 to 45 kHz output signal to the motor. A control signal of 0 Vdc provides a voltage of -24 to the motor while a 10 Vdc signal corresponds to -24 Vdc to be sent.

| GESRAM 14B4 |
| GESMPU-30H |
| GESMFI-1 |
| EVLAN-11 |
| GESSIO-1B |
| GESPIA-3A |
| GESADA-1 |
| GESADC-2C |
| GESTIM-1A |
| GESTIM-1A |
| GESTIM-1A |
| GESDAC-2B |

D.1 Layout of Gespac Card Cage Slots

APPENDIX E. PROLOG PREDICATES LINKED TO TACTICAL LEVEL C FUNCTIONS

The following is a listing of the C functions callable from the Strategic Level Prolog code.

ood(+string,[-integer])

Description: This is the Officer of the Deck predicate where the value of "string" may be any one of the following:

start_sun_network

Description: Open network socket from Sun SPARC to Gespac.

start_sun_and_iris_network

Description: Open network socket from Sun SPARC to Gespac and IRIS Elan.

start_networks

Description: The same as **start_sun_and_iris_network**.

start_dive_tracker

Description: Commands the Execution Level to fork DiveTracker process.

initialize_boards

Description: Commands the Execution Level to initialize all I/O boards on Gespac.

read_mission_file

Description: Commands the Tactical Level to read the mission file which contains phase set points, time outs, etc.

turn_on_prop_power

Description: Commands the Execution Level to activate the thruster and rear screw servo amplifiers.

turn_on_sonar_power

Description: Commands the Execution Level to activate the sonar power.

turn_off_sonar_power

Description: Commands the Execution Level to deactivate the sonar power.

gyros_on

Description: Tells the Execution Level that the gyroscopes are on and should be zeroed and read each time step.

zero_sensors

Description: Commands the Execution Level to zero the depth cell and gyroscopes at this time.

initialize_st1000_sonar

Description: Commands the Execution Level to initialize the ST1000 sonar head.

initialize_st725_sonar

Description: Commands the Execution Level to initialize the ST725 sonar head.

uncage_directional_gyroscope

Description: Commands the Execution Level to uncage the directional (free) gyroscope.

initialization_done

Description: Informs the Execution that no more initialization will be performed.

shutdown_network

Description: Causes the Tactical Level to disconnect from the Execution Level also from the IRIS if connected.

engineer(+string,[-integer])

Description: Used to activate engineering functions. Not functional at this time.

Execute Predicates Always Returns 1 (TRUE)

exec_submerge([-integer])
Description: Commands the vehicle to submerge using vertical thrusters.

exec_rotate([-integer])
Description: Commands the vehicle to rotate using lateral thrusters.

exec_servo_X([-integer])
Description: Commands the vehicle to servo to wall using rear screws and sonar for range information.

exec_stop_servo_X([-integer])
Description: Deactivates wall servoing mode.

exec_sleep(+integer,[-integer])
Description: Uses the C language sleep function to halt execution for "integer" seconds.

exec_start_timer([-integer])
Description: Starts a timer for the duration of the phase time out. Used in conjunction with **ask_time_out**.

exec_surface([-integer])
Description: Commands the vehicle to surface at maximum thrust.

exec_next_setpt_data([-integer])
Description: Increments the phase set point vector index.

exec_find_sonar_target([-integer])
Description: Begin search for target of interest using object detection algorithm in the Sonar Manager

exec_start_int_control_z([-integer])

> Description: Start implementing integral control for submergence.

exec_start_send_st1000_data([-integer])

> Description: Activates sending of sonar data to Sonar Manager process.

exec_stop_send_st1000_data([-integer])

> Description: Deactivates sending of sonar data.

exec_set_st1000_mode([-integer])

> Description: Sets current phase ST1000 sweep parameters.

exec_start_ping_st1000_sonar([-integer])

> Description: Start pinging the ST1000 sonar.

exec_stop_ping_st1000_sonar([-integer])

> Description: Stop pinging the ST1000 sonar.

exec_start_sonar_filter([-integer])

> Description: Activates the kinematic Kalman filter using ST1000 sonar data.

exec_set_heading([-integer])

> Description: Used to override current heading set point.

exec_set_heading_from_target([-integer])

> Description: Commands the vehicle to point towards target of interest.

exec_start_XY_psi_control([-integer])

> Description: Commands the vehicle maneuver to set points relative to a target.

exec_calc_pos_from_target([-integer])

> Description: Calculates the vehicle position relative to a target.

exec_track_target([-integer])

Description: Commands the sonar to track a target continuously.

Query Predicates Return either a 1 (TRUE) or 0 (FALSE)

ask_depth_reached([-integer])

Description: Returns TRUE if the depth is within error bounds, otherwise FALSE.

ask_heading_reached([-integer])

Description: Returns TRUE if the heading is within error bounds, otherwise FALSE.

ask_X_reached([-integer])

Description: Returns TRUE if the longitudinal position is within error bounds, otherwise FALSE.

ask_time_out([-integer])

Description: Queried after a call to **exec_start_timer**. If the phase time-out has been exceeded, the predicate returns TRUE. If the elapsed time is less FALSE is returned.

ask_system_problem([-integer])

Description: Returns TRUE if a system problem has been encountered, FALSE if no problems.

ask_surface_reached([-integer])

Description: Used in conjunction with **exec_surface**. If the depth is within .1 feet of the surface, TRUE is returned, otherwise FALSE.

ask_sonar_ping_out([-integer])

Description: Returns TRUE if allotted time to ping the sonar has expired

ask_int_control_z_on([-integer])

Description: Returns TRUE if integral control for submergence is active.

ask_int_control_z_off([-integer])

Description: Returns TRUE if integral control for submergence is inactive.

ask_sonar_target_found([-integer])

Description: Returns TRUE if target described by the Sonar Manager algorithm is identified.

ask_XY_psi_reached([-integer])

Description: Returns TRUE if X_{com}, Y_{com}, ψ_{com} for the particular phase is within error bounds.

ask_XY_psi_control_off([-integer])

Description: Returns TRUE if local navigation to a target is inactive.

APPENDIX F. EXECUTION LEVEL C LANGUAGE FUNCTIONS

The following is a listing and description of the C language Execution Level functions used for vehicle control and environment sensing.

A. SENSORS

double **roll_angle**()

Description: Returns the roll angle in radians from the vertical gyroscope.

double **pitch_angle**()

Description: Returns the pitch angle in radians from the vertical gyroscope.

double **calc_psi**()

Description: Returns the heading angle in radians from the free gyroscope. The angle returned includes any multiples of $\pm 2\pi$ radians if the vehicle rotates beyond $\pm 360^{\circ}$.

double **roll_rate_gyro**()

Description: Returns the roll rate in radians/sec from the rate gyroscope.

double **pitch_rate_gyro**()

Description: Returns the pitch rate in radians/sec from the rate gyroscope.

double **yaw_rate_gyro**()

Description: Returns the yaw rate in radians/sec from the rate gyroscope.

void **read_gyros**()

Description: Performs all of the gyroscope reads listed above with a single function call.

void **cage_dg**()

Description: Cages the free (directional) gyroscope.

void **uncage_dg**()

Description: Uncages the free gyroscope.

int **cage**()

Description: Returns TRUE (1) if free gyroscope is caged, and FALSE (0) if not.

double **depth**()

Description: Returns the vehicle depth in feet from the depth cell.

double **read_computer_battery_voltage**()

Description: Returns the computer battery voltage.

double **read_motor_gyro_battery_voltage**()

Description: Returns motor/gyro battery voltage.

int **leak_check**()

Description: Returns TRUE (1) if a leak is detected, otherwise FALSE (0).

zero_sensors(mode)

Description: At the time the function is called the current vehicle orientation, angular rate, and depth is read and these values are used as zero offsets until mission completion. If the gyroscopes are not to be used for a mission, the value **mode** should be set to 0, to prevent reading them. If the gyroscopes are to be used, **mode** should be set to 1.

B. ACTUATORS

thruster_power(onoff)

Description: Thruster Motor Power Control, **onoff** = TRUE to activate, FALSE to deactivate.

screw_power(onoff)

Description: Rear Screw Power Control, same as above.

motor(n,value)

Description: Commands to motor number **n** voltage where **value** = 0-1023 which maps to -24 - +24 VDC applied to the motor.

n =

0 Left Screw

1 Right Screw

2 Bow Vert. Thruster

3 Bow Lateral Thruster

4 Stern Vert. Thruster

5 Stern Lateral Thruster

zero_motors()

Description: Sends a 0 voltage command to all motors.

float **motor_speed(n)**

Description: Returns motor speed in rotations/sec for motor **n**.

ls_speed_control(n_com)

Description: Control left rear screw to speed **n_com** in rotations/sec.

rs_speed_control(n_com)

Description: Control right rear screw to speed **n_com** in rotations/sec.

rudder(angle)

Description: Commands rudders to deflect to **angle** in radians.

planes(angle)

Description: Commands planes to deflect to **angle** in radians.

`zero_fins()`

> Description: Sends a command of zero angle to all fins, (rudders and planes).

C.   EXECUTION LEVEL PRIMITIVES

The following are strings passed to the Execution Level from the Tactical Level.

Initialization Primitives:

> INITIALIZE_BOARDS
> START_DIVE_TRACKER_PROCESS
> TURN_ON_PROP_POWER
> TURN_OFF_PROP_POWER
> TURN_ON_SONAR_POWER
> TURN_OFF_SONAR_POWER
> ZERO_GYROS_AND_DEPTH_CELL
> ZERO_DEPTH_CELL
> UNCAGE_DIRECTIONAL_GYROSCOPE
> INITIALIZE_ST1000_SONAR
> INITIALIZE_ST725_SONAR
> INITIALIZATION_DONE

Control Primitives:

> Command Strings:

> SUBMERGE
> ROTATE
> SERVO_X
> XY_PSI_CONTROL
> STOP_SERVO_X
> SURFACE
> START_INT_CONTROL_Z
> START_DEPTH_ERROR_FILTER

START_HEADING_ERROR_FILTER
START_X_ERROR_FILTER
RESET_HEADING_COMMAND

SET_ST1000_MODE
PING_ST1000
STOP_PING_ST1000
START_SONAR_FILTER
START_SEND_ST1000_DATA
STOP_SEND_ST1000_DATA

Query Strings:

GET_DEPTH_INFO
GET_XY_PSI_INFO
GET_ST1000_INITIALIZATION_STATUS
GET_HEADING_INFO
GET_ST725_DATA
GET_ST1000_DATA
GET_SERVO_X_DATA

APPENDIX G. STANDARD SONAR COMMANDS

The following is a list of the standard sonar commands used for controlling the Tritech ST725 and ST1000 sonars.

Dec	Chr	Command	Description
043	'+'	MVCW	Move 1 step ClockWise, current step size. Replies 'T','t','F', or 'f'
045	'-'	MVCCW	Move 1 step CounterClockWise. Replies 'T','t','F', of 'f'
065	'A'	SETAV	Set mode return range bin avg. Value. Reply is 'A'
066	'B'	SETGN	Set gain for test purposes. Send 'B' followed by Char(gain value). Reply 'B'
067	'C'	SETYMI	Set initial gain for TVG (Time Varying Gain). Call is 'C' followed by Char(Ymin) Where Ymin = 255*gain/100, 0 < gain < 100. Reply is 'A'
068	'D'	INDEF	Inquire if head is using Default settings. Reply is 'T' if yes, 'F' if not.
069	'E'	SETYMA	Set final gain for TVG. Send 'E' followed by Char(Ymax). Reply is 'E'
070	'F'	SET96	Set 9600 baud communication speed (Default)
071	'G'	SET192	Set 19200 baud communication speed
072	'H'	SETHS	Set halfstep mode (0.9 deg). Reply 'H'
073	'I'	WRPARM	Inquire sonar parameters. Replies:

> Word - TxPulse length in 1.96 μ sec units
> Byte - NSAMPL, No. A/D samples per bin
> Byte - NBINS, No. of bins to collect
> Byte - Range Code, 0-8
> Byte - DataByte checksum, (Lower 8 bits of the sum of above values)

Dec	Chr	Command	Description
075	'K'	SETPK	Set mode return range Peak. Value (Normal Use). Reply 'K'
076	'L'	SCCW	Scan CounterClockWise (Scan Left). Does a scan ping. Returns NBINS/2 bytes of data + 'TtFf', then steps CCW
077	'M'	TSTSEN	Test head direction sensor. Replies 'TtFf'

079 'O' MOTOFF	Switch scan motor off. Replies 'O'

063 '?' NA	No Action. For comms test. Replies '?'

080 '**P**' RDPARM — Send sonar parameters. Send 'P' followed by
Word - TxPulse length in 1.96 μ sec units
Byte - NSAMPL, No. A/D samples per bin
Byte - NBINS, No. of bins to collect
Byte - Range Code, 0-8
Byte - DataByte checksum, (Lower 8 bits of the sum of above values)
Replies 'T' if checksum ok, else 'F'

082 '**R**' SCW	Scan ClockWise (Scan Right). Same return as SCCW

083 '**S**' SCAN	Scan but no step. Same return as SCW,SCCW

084 '**T**' GetGn	See TVG gains

085 '**U**' CLRTVG	Disable TVG, gain 0. Replies 'U' (Don't Use)

086 '**V**' VER	Returns ASCII version number '3'

087 '**W**' SendGn	See TVG gains

088 '**X**' SETTVG	Enable TVG (Normal). Replies 'X'

089 '**Y**' CLRHS	Full step mode set (1.8 deg). Replies 'Y'

Profiler Sonar Commands

036 '$' CLRDLE	Clear DLE protocol mode. No reply

040 '(' FESCCW — Get echorange, step CounterClockWise (For Profiling). Equivalent to sending 'Z' and
MVCCW. Replies:
Word - Echorange
Byte - 'TtFf'

041 ')' FESC	Get echorange, step ClockWise. Same reply as FESCCW

042 '*' SETDLE	Set DLE protocol mode. No reply

' ' FHalfS	Set step size 0.9 degrees. No reply

049 '**1**' FFullS	Set step size 1.8 degrees. No reply

050 '**2**' FDoubS	Set step size 3.6 degrees. No reply

| 060 | '<' | SHORTR | Set Profiler to 1 mm resolution. No reply |
| 061 | '=' | RDMOTD | Set motor step delay time, in 1.96 μ sec intervals. Send '=' followed by Word - Delay interval |
| 074 | 'J' | RDESPA | Send profiler parameters. Send 'J' followed by:

Word - ECPULS - Echo TxPulse length in 1.96 μ sec units
Word - TIMOUT - Timout for max range (mm)
Word - LOKOUT - Lockout time, min range 1.96 usec units
Word - ESWAIT - Wait time AutoES mode
Byte - GECMIN - Initial gain value
Word - GAINDT - Gain increment delay for TVG
Word - ECSCLX - Scale numerator for range
Word - ECSCLY - Scale denominator for range
Word - Maxdst - Max distance in range units
Word - DACSCX - Scale numerator for DAC O/P
Word - DACSCY - Scale denominator for DAC O/P
Byte - RngUnt - 0 = 10 mm, else 1 mm range units
Byte - CHKSUM - DataByte checksum, (Lower 8 bits of the sum of above values)
Replies 'T' if Ok, else 'F' and returns head to defaults |
| 078 | 'N' | ENQES | Inquire if profiler support. Reply '1' |
| 081 | 'Q' | WRESPA | Inquire profiler parameters. Replies RDESPA ended with DataByte checksum |
| 090 | 'Z' | ESRNG | Profiler range. Replies:
Word - Profiler range in mm |
| 091 | '[' | DRON | Profiler debug on. With 'Z' replies profiler raw time, distance and DAC value (0-4095) in ASCII |
| 093 | ']' | DROFF | Profiler debug off |

APPENDIX H. POLYGON INTERSECTION ALGORITHM

The intersection of a straight line with a plane was taken from (Dewey, 1988). A line may be represented by the equation:

$$P(t) = P(0) + cet \qquad (0 \leq t \leq 1) \tag{H.1}$$

where $P(0)$ is the point which locates the start of the line, $\mathbf{e}$ is a unit vector in the direction of the line, and c is a constant which specifies the length of the line (Figure H.1). The equation of the plane is

$$P(u,v) = P(0,0) + c_1\mathbf{e}_1 u + c_2\mathbf{e}_2 v \tag{H.2}$$

where u and v have the range 0 to 1 and c_1 and c_2 are the length and width dimensions of the plane of interest. $\mathbf{e}_1$ and $\mathbf{e}_2$ are orthogonal unit vectors which lie on the plane. The system of equations which describe the point of intersection is

$$P_c = P(0,0) + c_1\mathbf{e}_1 u + c_2\mathbf{e}_2 v = P(0) + cet \tag{H.3}$$

The three unknowns u, v, and t are uniquely determined by the three equations.

The above equations may be expanded in matrix form to yield

$$\begin{bmatrix} c_1\mathbf{e}_{1x} & c_2\mathbf{e}_{2x} & -c\mathbf{e}_x \\ c_1\mathbf{e}_{1y} & c_2\mathbf{e}_{2y} & -c\mathbf{e}_y \\ c_1\mathbf{e}_{1z} & c_2\mathbf{e}_{2z} & -c\mathbf{e}_z \end{bmatrix} \begin{Bmatrix} u \\ v \\ t \end{Bmatrix} = \begin{Bmatrix} x_0 - x_{00} \\ y_0 - y_{00} \\ z_0 - z_{00} \end{Bmatrix} \tag{H.4}$$

where

$$\begin{Bmatrix} x_0 \\ y_0 \\ z_0 \end{Bmatrix} = P(0) \qquad \text{and} \qquad \begin{Bmatrix} x_{00} \\ y_{00} \\ z_{00} \end{Bmatrix} = P(0,0)$$

337

Equation (4) may be written in compact form as

$$Mu = z \qquad \text{(H.5)}$$

and the solution vector u is obtained by

$$u = \begin{Bmatrix} u \\ v \\ t \end{Bmatrix} = M^{-1}z \qquad \text{(H.6)}$$

The point of intersection is simply

$$P_c = P(0) + cet \qquad \text{(H.7)}$$

If any of the solutions to u, v, or t is out of the range 0 to 1, the intersection lies on the extended plane (i.e. outside of the plane area bounded by c_1 and c_2). If the line is parallel to the plane, M is singular and no solution exists.
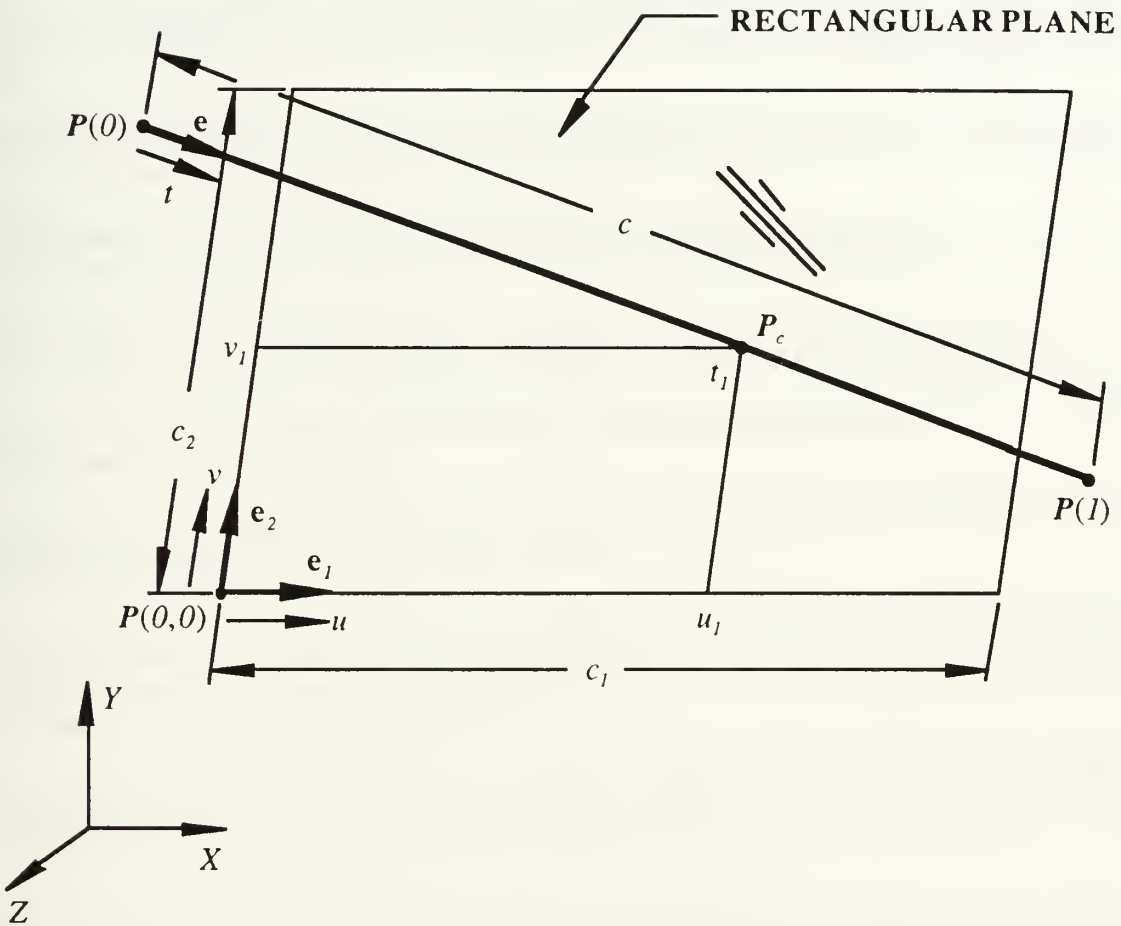
H.1 Vector Definitions for the Polygon Intersection Algorithm.

LIST OF REFERENCES

Albus, J. 1988, "System Description and Design Architecture for Multiple Autonomous Undersea Vehicles" National Institute of Standards and Technology, Technical Note 1251, September 1988.

Albus, J. and Quintero R. 1990, "Towards a Reference Model Architecture for Real-Time Intelligent Control Systems (ARTICS)", *Robotics and Manufacturing*, New York, ASME, vol. 3.

Antsaklis, P. J., Passino, K. M. 1993, *An Introduction to Intelligent and Autonomous Control.* Klewer Academic Publishers, ISBN #0-7923-9267-1.

Bellingham, J.G., Consi, T. R. 1991, "State Configured Layered Control," *Procs of Mobile Robots for Subsea Environments, IARP*, Monterey CA, pp 75-79.

Bellingham, J. G., Goudey, C. A., et. al. 1994, "A Second Generation Survey AUV", *Procs. IEEE Symp. on Auton. Underwater Vehicle Tech.*, July 1994, pp. 148-156.

Brooks , R. 1986, "A Robust Layered Control System for a Mobile Robot", *IEEE J. Rob. and Autom.*, Vol. RA-2, No. 1.

Brunner, G. M., "Experimental Verification of AUV Performance", MS Thesis. Naval Postgraduate School, Monterey, CA. Mar. 1988.

Byrnes, R. B., MacPherson, D.L. Kwak, S.H.., McGhee, R.B. and Nelson, M.L. 1992 "An Experimental Comparison of Hierarchical and Subsumption Software Architecture for Control of an Autonomous Underwater Vehicle", *Proc. IEEE Symposium on AUV Technology*, Washington DC, pp. 135-141.

Byrnes, R., "The Rational Behavior Model: A Multi-paradigm Tri-level Software architecture for the Control of Autonomous Vehicles", Ph.D. Dissertation. Naval Postgraduate School, Monterey, CA. Mar. 1993(a).

Byrnes, R., Kwak, S. H., McGhee, R. B., Healey, A. J., Nelson, M., 1993(b), "Rational Behavior Model: An Implemented Tri-Level Multilingual Software Architecture for Control of Autonomous Vehicles" *Proceedings of 8th International Symposium on Unmanned Untethered Submersible Technology*, Sept. 27-29, Durham, New Hampshire, pp. 160-179.

Byrnes, R. B., Healey, A. J., McGhee, R. B., Nelson, M. L., and Kwak, S. H., "A Rational Behavior Model Software Architecture for Intelligent Ships: An Approach to Motion and Mission Control", *Naval Engineers Journal,* American Society of Naval Engineers, March 1996, pp. 43-55.

Cassandras, C. G. 1993, *Discrete Event Systems, Modeling and Performance Analysis*, Aksen Associates, ISBN #0-256-11212-6.

Cody, S. E., "An Experimental Study of the Response of Small Thrusters To Step and Triangular Inputs", MS Thesis. Naval Postgraduate School, Monterey, CA. Dec. 1992.

Cristi, R., Healey, A. J., Papoulias, F. A., "Dynamic Output Feedback by Robust Observer and Variable Structure Control" *Proceedings of the American Control Conference*, San Diego June 1990(a).

Cristi, R., Healey, A. J., Papoulias, F. A., "Adaptive Sliding Mode Control of Autonomous Underwater Vehicles in the Dive Plane" *IEEE Journal of Oceanic Engineering*, July 1990(b).

Decarlo, R. A., Zak, S. H., and Matthews, G. P., "Variable Structure Control of Nonlinear Multivariable Systems: A Tutorial", *Proc. IEEE*, Vol. 76, pp. 212-232, Mar. 1988.

Dewey, Bruce R., *Computer Graphics for Engineers*. Harper & Row, New York, 1988.

Fossen, T. I., "Nonlinear Modelling and Control of Underwater Vehicles", Dr. Ing. Thesis, Norwegian Institute of Technology, Trondheim, 1991(a).

Fossen, T. I. and Sagatun, S., "Adaptive Control of Nonlinear Systems: A Case study of Underwater Robotic Systems", *J. of Robotic Systems*, Vol. 8, pp. 393-412, 1991(b).

Freidland, B. 1986, *Control System Design: Introduction to State Space Methods*. McGraw Hill, ISBN #0-07-022441-2.

Gelb, A., Ed. (1988), *Applied Optimal Estimation*. MIT Press, ISBN 0-262-57048-3, 10th Printing, 1988.

Hall., D. and Adams, M. 1992, "Autonomous Vehicle Software Taxonomy", *Procs. IEEE Symp. on AUV Tech.*, pp. 49-64.

Healey, A. J., Papoulias, F. A., MacDonald, G., " Design and Experimental Verification of a Model Based Compensator for Rapid AUV Depth Control ", *Proceedings of the 6th Unmanned, Untethered, Submersible Technology*, Washington DC., June 12-14 1989.

Healey, A. J., McGhee, R. B., Christi, R., Papoulias, F. A., Kwak, S.H. Kanayama, Y. and Lee, Y., 1991, "Mission Planning, Execution and Data Analysis for the NPS AUV II Autonomous Underwater Vehicle", *Procs of 1st IARP Workshop on Mobile Robots for Subsea Environments*, MBARI, Pacific Grove, CA, pp. 177-186.

Healey, A. J., Marco, D. B.1992(a), "Experimental Verification of Mission Planning by Autonomous Mission Execution and Data Visualization using the NPS AUV II." Proceedings of IEEE Oceanic Engineering Society *Symposium on Autonomous Underwater Vehicles, AUV-92* Washington DC., June 2-3, 1992.

Healey, A. J., Marco, D. B.1992(b), "Slow Speed Flight Control of Autonomous Underwater Vehicles: Experimental Results with NPS AUV II" *Proceedings of the 2nd International Offshore and Polar Engineering Conference*, San Francisco, July 14-19 1992.

Healey, A. J., Good, M., "The NPS AUV II Autonomous Underwater Vehicle Testbed: Design and Experimental Verification" *Naval Engineers Journal,* ASNE , May 1992.

Healey, A. J., Lienard, D., "Multivariable Sliding Mode Control for Autonomous Diving and Steering of Unmanned Underwater Vehicles", *IEEE J. Oceanic Eng.*, Vol. 18 No. 3, pp. 327-339, July 1993.

Healey, A. J., "Marine Vehicle Dynamics", ME 4823 Course Notes, Naval Postgraduate School, Monterey, CA. 1993.

Healey, A. J. , Pascoal, A. M. , Pereira, F. L. "Autonomous Underwater Vehicles: An Application of Intelligent Control Technology", *Proceedings of the American Control Conference*, Seattle, Washington June 21-23, 1995, pp. 2943-2949.

Healey, A. J., Kwak, S. H., and McGhee, R.B., "An Experimental Study of Software Architectures and Software Reuse for Control of Unmanned Underwater Vehicles", NSF Proposal, 1993.

Healey, A. J., et. al. 1994, "Tactical / Execution Level Coordination for Hover Control of the NPS AUV II Using Onboard Sonar Servoing" *Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology*, July 19-20, Cambridge, Mass. 1994 pp. 129-138.

Healey, A. J., Rock, S. M., Cody, S., Miles, D., Brown, J. P., " Towards an Improved Understanding of Thruster Dynamics for Underwater Vehicles" *IEEE Journal of Oceanic Engineering,* Vol. 20, No. 4, 1995, pp. 354-361.

Ingold, B. J., "AUV Navigation from Image Profile Segments using a High Frequency Sonar", MS Thesis. Naval Postgraduate School, Monterey, CA. Dec. 1992.

Kwak, S. H., Thornton, F. P. B., 1994, "A Concurrent Object-Oriented Implementation for the Tactical Level of the Rational Behavior Model Software Architecture for UUV Control", *Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology,* July 19-20, Cambridge, Mass. 1994 pp. 54-60.

Lindgren, A. G., Cretella, D. B., Bessacini, A. F., "Dynamics and Control of Submerged Vehicles", *Trans. Instrument Society of America*, Vol. 6, pp. 335-346, Dec. 1967.

MacDonald, G., "Model Based Compensator Design and Experimental Verification of Control Systems for a Model AUV", MS Thesis. Naval Postgraduate School, Monterey, CA. Mar. 1989.

Marco, D. B., Healey, A. J., "Sliding Mode Acoustic Servoing for an Autonomous Underwater Vehicle from Simulations and Experiments" *Proceedings of the 1992 Offshore Technology Conference*, Houston Texas, 1992 Paper No. OTC 6974.

Marco, D. B., Healey, A. J., McGhee, "Autonomous Underwater Vehicles: Hybrid Control of Mission and Motion", *Journal of Autonomous Robots*, Vol. 3,169-186 (1996).

Marks, R. L., Rock, S. M., Lee, M. J., "Real Time Video Mosaicking of the Ocean Floor", *Procs. IEEE Symp. on Auton. Underwater Vehicle Tech.*, July 1994, pp. 21-28.

343

Milliken, G. L., "Multivariable Control of an Underwater Vehicle", M. S. Thesis, MIT, Cambridge, Mass., 1984.

Murata, T.1989, "Petri Nets: Properties, Analysis, and Applications" *Proceedings of IEEE*, 77, pp. 541-580.

Perrier, M., Rigaud, V., Peuch, A., Coste-Maniere, E., Simon, B., " Vortex: A versatile Testbed Vehicle for Control Algorithms Evaluation", *Proceedings of the 8th. UUST*, Durham New Hampshire, September 27-29, 1993, pp. 29-36.

Saridis, G. N., "Intelligent Robotic Control", *IEEE Trans. on Automatic Control*, Vol. AC-28, No. 5, May, 1983, pp. 547-557.

Saridis, G. N. 1989, "Analytical Formulation of the Principle of Increasing Precision with Decreasing Intelligence for Intelligent Machines", *Automatica*, Vol. 25, pp. 461-467.

Sarpkaya, T., and Issacson, M. I., *Mechanics of Wave Forces on Offshore Structures.* Van Nostran, ISBN # 0-442-25402-4, 1981.

Saunders, T., "Performance of Small Thrusters and Propulsion Systems" MS Thesis. Naval Postgraduate School, Monterey, CA. Mar. 1990.

Scrivener, A., "Acoustic Underwater Navigation of the Phoenix Autonomous Underwater Vehicle Using the DiveTracker System", MS Thesis. Naval Postgraduate School, Monterey, CA. Mar. 1996.

Silva, V., Oliveira, P., Silvestre, C., Pascoal, A. 1994, "Mission Coordination Synthesis and Execution using the CORAL Language", Report 1994/08/04 Instituto de Systemas e Robotica, Instituto Superior Technico, Lisbon, Portugal.

Simon, D., Espiau, B., Castillo, E., Kapellos, K., 1993, "Computer Aided Design of a Generic Robot Controller Handling Reactivity and Real Time Control Issues", *IEEE Transactions on Control Systems Technology*, Vol. 1, No. 4, Dec. 1993, pp. 213-229.

Slotine, J. J. E., and Li, W. 1991, *Applied Nonlinear Control.* Prentice-Hall Int., ISBN # 0-13-040890-5.

Smith, S. M., Dunn, S. 1994, "The Ocean Voyager II: An AUV Designed for Coastal Oceanography", *Procs. IEEE Symp. on Auton. Underwater Vehicle Tech.*, July 1994, pp. 139-148.

Smith, R., Stevens, A., Frost, A., and Probert, P., "Developing a Sensor-based Underwater Navigation System", *Procs of Underwater Robotics, IARP '96*, Toulon-La Seyne, France.

Sousa, J. B., Pereira, F. L. and Silva, E. P. 1994, "A Dynamically Configurable Architecture for the Control of an AUV", *Procs OCEANS '94*, Brest, France, 1994, pp. 131-136.

Torsiello, K. A., "Acoustic Positioning of the NPS AUV II During Hover Conditions", MS Thesis. Naval Postgraduate School, Monterey, CA. Mar. 1994.

Yoerger, D. R., and Slotine, J. J. E., "Robust Trajectory Control of Underwater Vehicles", *IEEE J. Oceanic Eng.*, Vol. OE-10, No. 4, pp. 462-470, Oct. 1985.

Yoerger, D. R., Newman, J. B., and Slotine, J. J. E., "Supervisory Control System for the JASON ROV", *IEEE J. Oceanic Eng.*, Vol. OE-11, No. 3, pp. 392-400, July 1986.

Yuh, J., "Modeling and Control of Underwater Vehicles", *IEEE Trans. of Systems, Man, and Cybernetics*, Vol. 20, pp. 1475-1483, 1990.

Zinni, J., "Analysis of the DiveTracker Acoustical Navigation System for the NPS AUV" MS Thesis. Naval Postgraduate School, Monterey, CA. Dec. 1995.

INITIAL DISTRIBUTION LIST