NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

TRAJECTORY PLANNING FOR THE ARIES AUV

by

John J. Keegan

June 2002

Thesis Advisor: Anthony J. Healey

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

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 blank)***	**2. REPORT DATE** June 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Trajectory Planning for the ARIES AUV			**5. FUNDING NUMBERS**
6. AUTHOR(S) John J. Keegan			
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) Office of Naval Research, 800 N. Quincy St., Arlington, VA 22217-5660			**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 This thesis supports ongoing ONR research in the area of Autonomous Underwater Vehicles (AUVs) and Mine Warfare. It shows a simulation of a two-vehicle autonomous rendezvous using both along track and cross track position controllers. Conducting open water experiments with the ARIES AUV identified the added mass matrix and hydrodynamic coefficients of the longitudinal equation of motion. The results indicate that it will be possible to maneuver an AUV to a specific rendezvous point at a specified time. Two-vehicle rendezvous maneuvers are likely to be needed in multi-vehicle operations when data transfer between range-limited communications modems are used.			
14. SUBJECT TERMS Underwater Vehicle, AUV, Trajectory Planning, Control			**15. NUMBER OF PAGES** 116
			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. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

TRAJECTORY PLANNING FOR THE ARIES AUV

John J. Keegan
Lieutenant Commander, United States Navy
B.E.E., Villanova University, 1990

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2002

Author: John J. Keegan

Approved by: Anthony J. Healey
Thesis Advisor

Terry R. McNelley
Chairman, Department of Mechanical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis supports ongoing ONR research in the area of Autonomous Underwater Vehicles (AUVs) and Mine Warfare. It shows a simulation of a two-vehicle autonomous rendezvous using both along track and cross track position controllers. Conducting open water experiments with the ARIES AUV identified the added mass matrix and hydrodynamic coefficients of the longitudinal equation of motion. The results indicate that it will be possible to maneuver an AUV to a specific rendezvous point at a specified time. Two-vehicle rendezvous maneuvers are likely to be needed in multi-vehicle operations when data transfer between range-limited communications modems are used.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge Professor Anthony J. Healey for his guidance and patience throughout the thesis process. His knowledge on this subject area is limitless and truly inspirational. Additionally, I would like to thank CDR Bill Marr for his technical assistance and support in obtaining and interpreting data for this thesis. My former Commanding Officers on USS DEFENDER (MCM-2), CDR Andy Fuller and CDR Tom Negus, deserve special mention for endorsing my career path and having faith in my abilities. Their unwavering support was crucial for my attendance at NPS. Hopefully this work pays off part of the debt I owe to the MIW community. Finally, and most importantly, I would like to thank my wife, Sue, and daughters, Caitlin and Caroline for their love and support. The work was conducted in concert with work funded by ONR under Dr. T. Swean.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Research in the field of autonomous underwater vehicles (AUVs) at the Naval Postgraduate School (NPS) has progressed steadily since the inception of the Center for AUV Research in 1987. The operational capabilities and sophistication of software and hardware has greatly increased with each new generation of vehicle. From humble beginnings in swimming pools to open ocean operation, these vehicles have been at the forefront of AUV research.

The unique ability of AUVs to operate in very shallow water (VSW), where depths range from ten to thirty feet, enables the Mine Warfare (MIW) Commander to search, detect, and classify mines effortlessly and safely when compared to current practice Navy Special Operations or Special Warfare Teams. In an effort to expand the search area and provide the MIW Commander with near real-time information, the use of multiple vehicles having the ability to communicate between each other is the logical solution. The Acoustic Radio Interactive Exploratory Server (ARIES) AUV is designed to operate as a communications server vehicle. It is outfitted with the Florida Atlantic University (FAU) acoustic modem and has the capability to act as a command and control vehicle for numerous vehicles while operating in the search area. Equipped with a radio modem, when surfaced, ARIES has the ability to send and receive data to the MIW Commander embarked offshore.

Current modem technology has limitations, requiring increases in both data rate and range. This will also require an increase in power and size of the modem. By using a mobile server, such as ARIES, close proximity, high-speed data transfers will achieve the same results as a long-range modem. Current employment of the ARIES is shown in Figure 1. Command and control of ARIES is accomplished using a Boston Whaler as a relay station. Radio ranges of four to six nautical miles with data transfer rates of 30,000 bits/sec are routine.

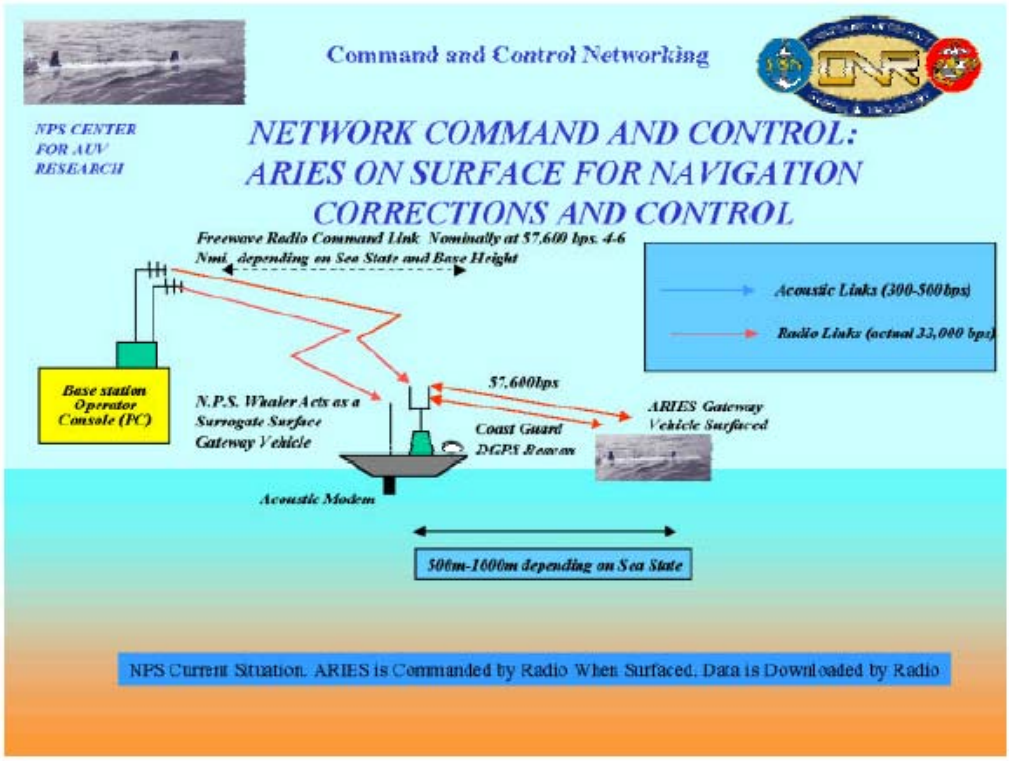

Figure 1. Current Employment of ARIES (From: Healey, 2001)

Possible employment of ARIES is shown in Figure 2. Here ARIES will rendezvous with other vehicles, such as the Remote Environmental Measuring Units (REMUS), conduct file transfer operations and then relay the information to the command and control base.

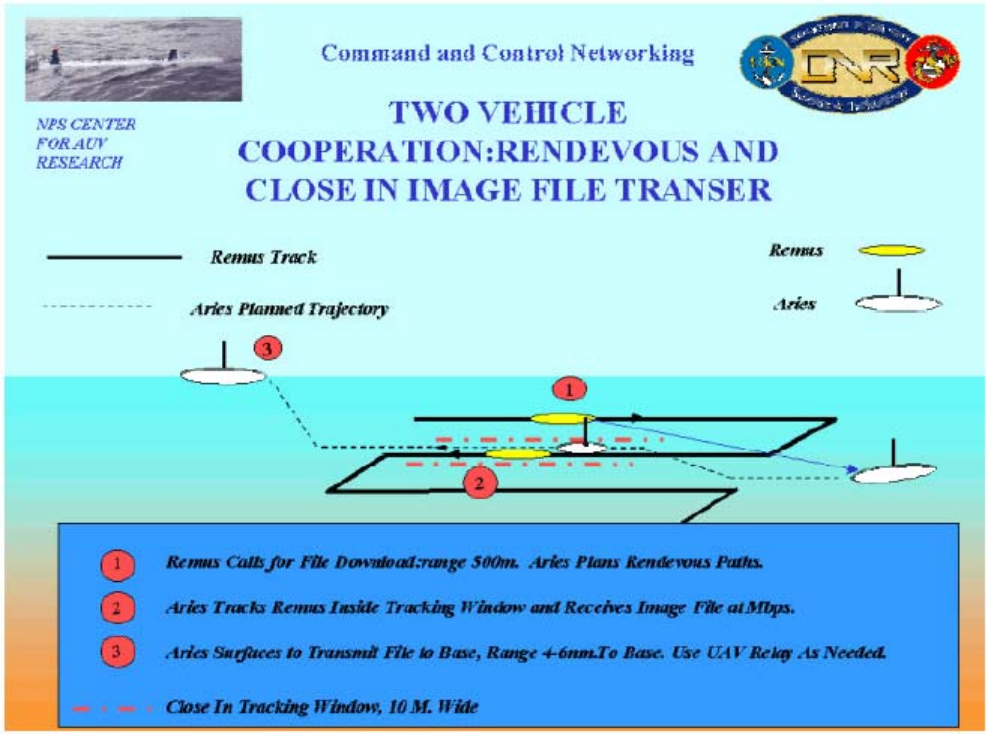

Figure 2. Possible Employment of ARIES (From: Healey, 2001)

In order to accomplish the rendezvous, ARIES must have the capability to arrive at a certain location at a certain time. This is known as trajectory planning.

B. SCOPE OF THIS WORK

Trajectory planning has not been researched in the case of AUVs and has only been touched on in the robotics community. Kant and Zucker (1986) illustrated the everyday version of the trajectory-planning problem as simply being, "How does one get from here to there"? This is a problem that humans solve so often and intuitively that the underlying complexity is given little or no thought.

The focus of this thesis is two-fold:

1. To develop a new behavior for ARIES - that of arriving at a defined place in space and time; and

2. To extend cross-track error guidance algorithms (Marco, 2000) to close along track as well as cross track errors resulting in a trajectory controller.

Chapter II will focus on the equations of motion for an AUV and the associated steering control laws. Chapter III will discuss trajectory planning theory and implementation. Chapter IV will discuss the theory and design of a sophisticated speed controller to complement the trajectory controller discussed in the previous chapter. Chapter V will present simulation results from the implementation of the complete trajectory/speed controller combination and Chapter VI will offer conclusions and recommendations for future study.

II. EQUATIONS OF MOTION AND AUV MODELING

A. GENERALIZED STEERING EQUATIONS OF MOTION

This section describes the equations of motion that are the basis for the model used to construct ARIES steering controllers.

Using a Newton-Euler approach, Healey, (1995) derives the equations of six degree of freedom motion as:

SURGE EQUATION OF MOTION

$$m[\dot{u}_r - v_r r + w_r q - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(pr + \dot{q})] + (W - B)\sin\theta = X_f \tag{1}$$

SWAY EQUATION OF MOTION

$$m[\dot{v}_r + u_r r - w_r p + x_G(pq + \dot{r}) - y_G(p^2 + r^2) + z_G(qr - \dot{p})] - (W - B)\cos\theta\sin\phi = Y_f \tag{2}$$

HEAVE EQUATION OF MOTION

$$m[\dot{w}_r - u_r q + v_r p + x_G(pr - \dot{q}) + y_G(qr + \dot{p}) - z_G(p^2 + q^2)] + (W - B)\cos\theta\cos\phi = Z_f \tag{3}$$

ROLL EQUATION OF MOTION

$$\begin{aligned} &I_x\dot{p} + (I_z - I_y)qr + I_{xy}(pr - \dot{q}) - I_{yz}(q^2 - r^2) - I_{xz}(pq + \dot{r}) + m[y_G(\dot{w} - u_r q + v_r p) \\ &- z_G(\dot{v}_r + u_r r - w_r p)] - (y_G W - y_B B)\cos\theta\cos\phi + (z_G W - z_B B)\cos\theta\sin\phi = K_f \end{aligned} \tag{4}$$

PITCH EQUATION OF MOTION

$$\begin{aligned} &I_y\dot{q} + (I_z - I_z)pr - I_{xy}(qr + \dot{p}) + I_{yz}(pq - \dot{r}) + I_{xz}(p^2 - r^2) - m[x_G(\dot{w} - u_r q + v_r p) \\ &- z_G(\dot{u}_r - v_r r + w_r q)] + (x_G W - x_B B)\cos\theta\cos\phi + (z_G W - z_B B)\sin\theta = M_f \end{aligned} \tag{5}$$

YAW EQUATION OF MOTION

$$\begin{aligned} &I_z\dot{r} + (I_y - I_x)pq - I_{xy}(p^2 - q^2) - I_{yz}(pr + \dot{q}) + I_{xz}(qr - \dot{p}) + m[x_G(\dot{v}_r + u_r r - w_r p) \\ &- y_G(\dot{u}_r - v_r r + w_r q)] - (x_G W - x_B B)\cos\theta\sin\phi - (y_G W - y_B B)\sin\theta = N_f \end{aligned} \tag{6}$$

Where:

u_r, v_r, w_r = component velocities for a body fixed system with respect to the water

p, q, r = component angular velocities for a body fixed system
W = weight
B = buoyancy
I = mass moment of inertia terms
x_B, y_B, z_B = position difference between geometric center of ARIES and center of buoyancy
x_G, y_G, z_G = position difference between geometric center of ARIES and center of gravity
X_f, Y_f, Z_f, K_F, M_f, N_f = sums of all external forces acting on ARIES in the particular body fixed direction

Healey (1995) further simplifies Equations 1 thru 6 by assuming that the center of mass of the vehicle lies below the origin (z_G is positive) while x_G and y_G are zero, and that the vehicle is symmetric in its inertial properties. It is also assumed that motions in the vertical are negligible (i.e. [w_r, p, q, r, Z, ϕ, θ] = 0) and that u_r equals the forward speed, U_o. The simplified equations of motion are thus:

$$u_r = U_o \tag{7}$$

$$m\dot{v}_r = -mU_o r + \Delta Y_f(t) \tag{8}$$

$$I_{zz}\dot{r} = \Delta N_f(t) \tag{9}$$

$$\dot{\psi} = r \tag{10}$$

$$\dot{X} = U_o \cos\psi - v_r \sin\psi + U_{cx} \tag{11}$$

$$\dot{Y} = U_o \sin\psi - v_r \cos\psi + U_{cy} \tag{12}$$

Johnson (2001) defines $\Delta Y_f(t)$ and $\Delta N_f(t)$ as forces that are functions of the vehicles dynamic parameters. Through the assumption of 'small' motions 'hydrodynamic coefficients' can be defined related to the individual motion components. The expression for the transverse force is:

$$Y_f = Y_{\dot{v}_r}\dot{v}_r + Y_{v_r}v_r + Y_{\dot{r}}\dot{r} + Y_r r \tag{13}$$

and for the expression for the rotational force is:

$$N_f = N_{\dot{v}_r}\dot{v}_r + N_{v_r}v_r + N_{\dot{r}}\dot{r} + N_r r \tag{15}$$

This leads to:

$$Y_{\dot{v}_r} = \frac{\partial Y_f}{\partial \dot{v}_r};\ Y_{v_r} = \frac{\partial Y_f}{\partial v_r};\ Y_{\dot{r}} = \frac{\partial Y_f}{\partial \dot{r}};\ Y_r = \frac{\partial Y_f}{\partial r};$$

and

$$N_{\dot{v}_r} = \frac{\partial N_f}{\partial \dot{v}_r};\ N_{v_r} = \frac{\partial N_f}{\partial v_r};\ N_{\dot{r}} = \frac{\partial N_f}{\partial \dot{r}};\ N_r = \frac{\partial N_f}{\partial r};$$

Where:

$Y_{\dot{v}_r}$ = added mass in sway coefficient
$Y_{\dot{r}}$ = added mass in yaw coefficient
Y_{v_r} = coefficient of sway force induced by side slip
Y_r = coefficient of sway force induced by yaw
$N_{\dot{v}_r}$ = added mass moment of inertia in sway coefficient
$N_{\dot{r}}$ = added mass moment of inertia in yaw coefficient
N_{v_r} = coefficient of sway moment from side slip
N_r = coefficient of sway moment from yaw

In addition, the action of the rudder will produce forces that when linearized are: $Y_\delta \delta_r(t)$ and $N_\delta \delta_r(t)$. The dynamics of the vehicle are thus defined as:

$$m\dot{v}_r = -mU_o r + Y_{\dot{v}_r}\dot{v}_r + Y_{\dot{v}_r} v_r + Y_{\dot{r}}\dot{r} + Y_r r + Y_\delta \delta_r(t) \tag{16}$$

$$I_{zz}\dot{r} = N_{\dot{v}_r}\dot{v}_r + N_{v_r} v_r + N_{\dot{r}}\dot{r} + N_r r + N_\delta \delta_r(t) \tag{17}$$

$$\dot{\psi} = r \tag{18}$$

The kinematics of the vehicle is described by Equations (11) and (12) where U_{cx} and U_{cy} are the current velocities in the associated direction. The steering dynamics of ARIES in matrix form, $\boldsymbol{M}\dot{x} = \boldsymbol{A}x + \boldsymbol{B}u$, is:

$$\begin{bmatrix} m - Y_{\dot{v}_r} & -Y_{\dot{r}} & 0 \\ -N_{\dot{v}_r} & I_{zz} - N_{\dot{r}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{v}_r \\ \dot{r} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} Y_{v_r} & Y_r - mU_o & 0 \\ N_{v_r} & N_r & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_r \\ r \\ \psi \end{bmatrix} + \begin{bmatrix} Y_\delta \\ N_\delta \\ 0 \end{bmatrix} \delta_r(t) \tag{19}$$

As Johnson (2001) points out, $\delta_r(t)$ is a generalized command that represents the control input to both rudders. The rudders act together, but turn in opposite directions allowing for rapid turning during operation.

The final assumption made for ARIES (Johnson, 2001) is that the cross coupling terms in the mass matrix are zero. This is based on the vehicle's symmetry and the rudders being very close to being equidistant from the body center. Thus, in matrix form, the final vehicle dynamics are defined as:

$$\begin{bmatrix} m - Y_{\dot{v}_r} & 0 & 0 \\ 0 & I_{zz} - N_{\dot{r}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{v}_r \\ \dot{r} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} Y_{v_r} & Y_r - mU_o & 0 \\ N_{v_r} & N_r & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_r \\ r \\ \psi \end{bmatrix} + \begin{bmatrix} Y_\delta \\ N_\delta \\ 0 \end{bmatrix} \delta_r(t) \tag{20}$$

Johnson (2001) determined the hydrodynamic coefficients for ARIES to be as follows:

$m - Y_{\dot{v}_r} = 456.76$
$Y_{\dot{r}} = 69.90$
$Y_{v_r} = -68.16$
$Y_r = 406.30$
$N_{\dot{r}} = -35.47$
$N_{v_r} = -10.89$
$N_r = -88.34$

B. ARIES CONTROL LAWS FOR STEERING AND CROSS TRACK ERROR

This section describes the control laws used by ARIES for steering and cross track error.

ARIES has four different autopilots for flight maneuvering control: diving, steering, altitude above bottom, and cross track error. With the object of accomplishing trajectory planning, only the steering and cross track error controllers are of interest. The controllers are based on sliding mode control theory presented by Healey and Lienard (1993).

Marco and Healey (2001) argue that a second order model is sufficient for heading control. The sideslip effects are treated as disturbances that the control overcomes, so the heading model becomes:

$$\dot{r}(t) = ar(t) + b\delta_r(t) + (disturbances) \tag{21}$$

$$\dot{\psi}(t) = r(t) \tag{22}$$

From past in-water experiments, a = -0.30 sec^{-1} and b = -0.1125 sec^{-2}. $\delta_r(t)$ is the stern rudder angle. The sliding surface and stern rudder command for heading control is thus defined by Healey and Marco (2001) as:

$$\sigma(t) = -0.9499r(t) + 0.1701(\psi_{com} - \psi(t)) \tag{23}$$

$$\delta_r(t) = -1.543(2.5394r(t) + \eta \tanh(\sigma(t)/\phi)) \tag{24}$$

Where η = 1.0, ϕ = 0.5 and ψ_{com}- $\psi(t)$ is the heading error.

In order for ARIES to follow a straight-line path, Marco and Healey (2001) use a combination of line of sight guidance proposed by Healey and Lienard (1993) and a cross track error (CTE) control. The reasoning behind this is that with large heading errors, the cross track error control cannot be guaranteed stable, while a line of sight heading control will reduce heading errors to zero. Alternating between the two controllers will minimize both cross track and heading errors. Figure 3 shows the track geometry and velocity vector diagrams to support the argument.

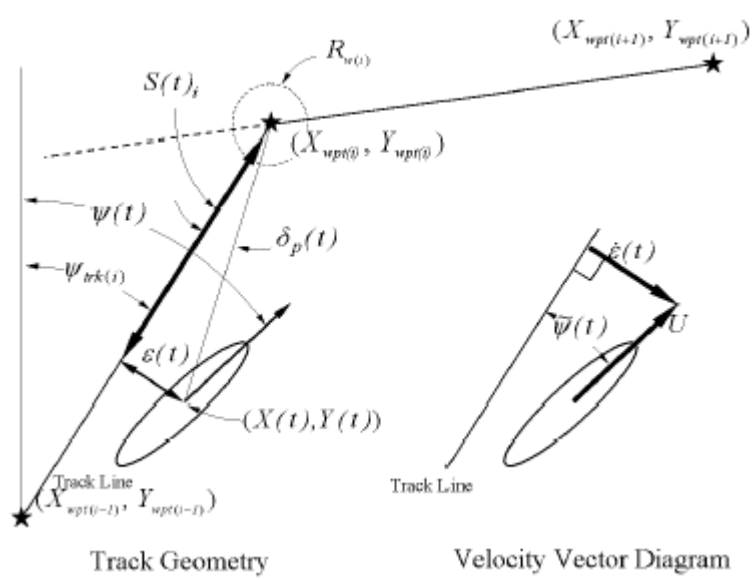

Figure 3. Track Geometry and Velocity Vector (From: Marco and Healey, 2001)

The perpendicular distance between the center of the vehicle (X(t),Y(t)) and the adjacent track line is the cross track error, ε(t). The goal of the CTE control is to

minimize ε(t). The total track length is defined as the distance between the *i*th and *i*-1 waypoints and is given by:

$$L_i = \sqrt{(X_{wpt(i)} - X_{wpt(i-1)})^2 + (Y_{wpt(i)} - Y_{wpt(i-1)})^2} \tag{25}$$

The ordered pairs ($X_{wpt(i)}$, $Y_{wpt(i)}$) and ($X_{wpt(i-1)}$, $Y_{wpt(i-1)}$) are the current and previous way points. The track angle is defined as:

$$\psi_{trk(i)} = \mathbf{arctan}\,2(Y_{wpt(i)} - Y_{wpt(i-1)}, X_{wpt(i)} - X_{wpt(i-1)}) \tag{26}$$

Where arctan2 is defined by MATLAB as the inverse tangent function. The cross track heading error for the *i*th segment is defined as:

$$\tilde{\psi}(t)_{CTE(i)} = \psi(t) - \psi_{trk(i)} \tag{27}$$

where $\tilde{\psi}(t)_{CTE(i)}$ must be normalized to lie between ± 180 degrees. The difference between the current vehicle position and the next way point is:

$$\tilde{X}(t)_{wpt(i)} = X_{wpt(i)} - X(t) \tag{28}$$

$$\tilde{Y}(t)_{wpt(i)} = Y_{wpt(i)} - Y(t) \tag{29}$$

With the above definitions, the distance to the *i*th way point projected to the track line can be defined as:

$$S(t)_i = [\tilde{X}_{wpt(i)} \quad \tilde{Y}_{wpt(i)}] \bullet [(X_{wpt(i)} - X_{wpt(i-1)}) \quad (Y_{wpt(i)} - Y_{wpt(i-1)})] / L_i \tag{30}$$

therefore, $S(t)_i$ ranges between 0-100 percent of L_i. The cross track error may now be defined as:

$$\varepsilon(t) = S(t)_i \,\mathbf{sin}(d_p(t)) \tag{31}$$

where $d_p(t)$ is the angle (normalized to lie between ±180 degrees) between the line of sight to the next way point and the current track line given by:

$$d_p(t) = \mathbf{arctan}\,2(Y_{wpt(i)} - Y_{wpt(i-1)}, X_{wpt(i)} - X_{wpt(i-1)}) - \mathbf{arctan}\,2(\tilde{Y}(t)_{wpt(i)}, \tilde{X}(t)_{wpt(i)}) \tag{32}$$

Marco and Healey (2001) continue by defining the sliding surface in terms of derivatives of the cross track error such that the sliding surface for the CTE controller becomes a second order polynomial of the form:

$$\sigma(t) = Ur(t)\cos(\tilde{\psi}(t)_{CTE(i)}) + \lambda_1 U \sin(\tilde{\psi}(t)_{CTE(i)}) + \lambda_2 \varepsilon(t) \tag{33}$$

The rudder input is thus expressed as:

$$\begin{aligned}\delta_r(t) = {} & (Ub\cos(\tilde{\psi}(t)_{CTE(i)})^{-1}(-Uar(t)\cos(\tilde{\psi}(t)_{CTE(i)}) + U(r(t))^2 \sin(\tilde{\psi}(t)_{CTE(i)}) \\ & - \lambda_1 Ur(t)\cos(\tilde{\psi}(t)_{CTE(i)}) - \lambda_2 U \sin(\tilde{\psi}(t)_{CTE(i)}) - \eta(\sigma(t)/\phi)\end{aligned} \tag{34}$$

where $0 < \tilde{\psi}(t)_{CTE(i)} < \frac{\pi}{2}$, λ_1 = 0.6, λ_2 = 0.1, η = 0.1 and ϕ = 0.5. If $\tilde{\psi}(t)_{CTE(i)} > \frac{\pi}{2}$, ARIES will follow the track, but travel in the opposite direction to that desired. In order to prevent this from happening in practice, a bound of 40 degrees is used as a switch to line of sight (LOS) control.

Marco and Healey (2001) determined that when the magnitude of the cross track heading error exceeded 40 degrees, a LOS controller is used and the heading command and LOS error could be determined from:

$$\psi(t)_{com(LOS)} = \arctan 2(\tilde{Y}(t)_{wpt(i)}, \tilde{X}(t)_{wpt(i)}) \tag{35}$$

$$\tilde{\psi}(t)_{LOS} = \psi(t)_{com(LOS)} - \psi(t) \tag{36}$$

The control laws for the heading controller, Equations 23 and 24, are used for heading control. The need for the LOS controller is apparent in two cases: 1) when the mission starts and ARIES' initial heading is greater than 40 degrees from the initial way point and 2) when the angle between two sequential track lines exceed 40 degrees. Once the cross track heading error reduces to less than 40 degrees, ARIES utilizes the CTE controller.

C. SIMULATION RESULTS

Appendix A contains the MATLAB .m file written by Marco (2001) that demonstrates Marco and Healey's heading and cross track error controllers for the NPS ARIES. Table 1 shows the track.out file used to plan the simulated ARIES mission. It

consists of 5 tracks (rows). The columns are defined below the table. Figure 4 shows the results of the simulation.

Table 1. track.out file (From: Marco and Healey, 2001)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
10.0	10.0	2.75	2.75	0	1.25	1	0	25.00	8.00	40.00
10.0	210.0	2.75	2.75	0	1.25	1	0	25.00	8.00	200.00
40.0	210.0	2.75	2.75	0	1.25	1	0	25.00	2.00	30.00
40.0	10.0	2.75	2.75	0	1.25	1	0	25.00	2.00	200.00
-20.0	-60.0	2.75	2.75	0	1.25	1	0	25.00	2.00	100.00

Column #	Description
1	*X* position way point (meters)
2	*Y* position way point (meters)
3	Left screw command speed (volts)
4	Right screw command speed (volts)
5	Control mode flag, 0 = Depth Control, 1 = Altitude Control
6	Commanded Altitude (feet, if applicable)
7	Commanded Depth (feet, if applicable)
8	Perform GPS popup on this track? 1 = Yes, 0 = No
9	Duration of GPS popup (seconds)
10	Watch Radius, $R_{w(i)}$ (meters)
11	Way point timeout. (seconds)

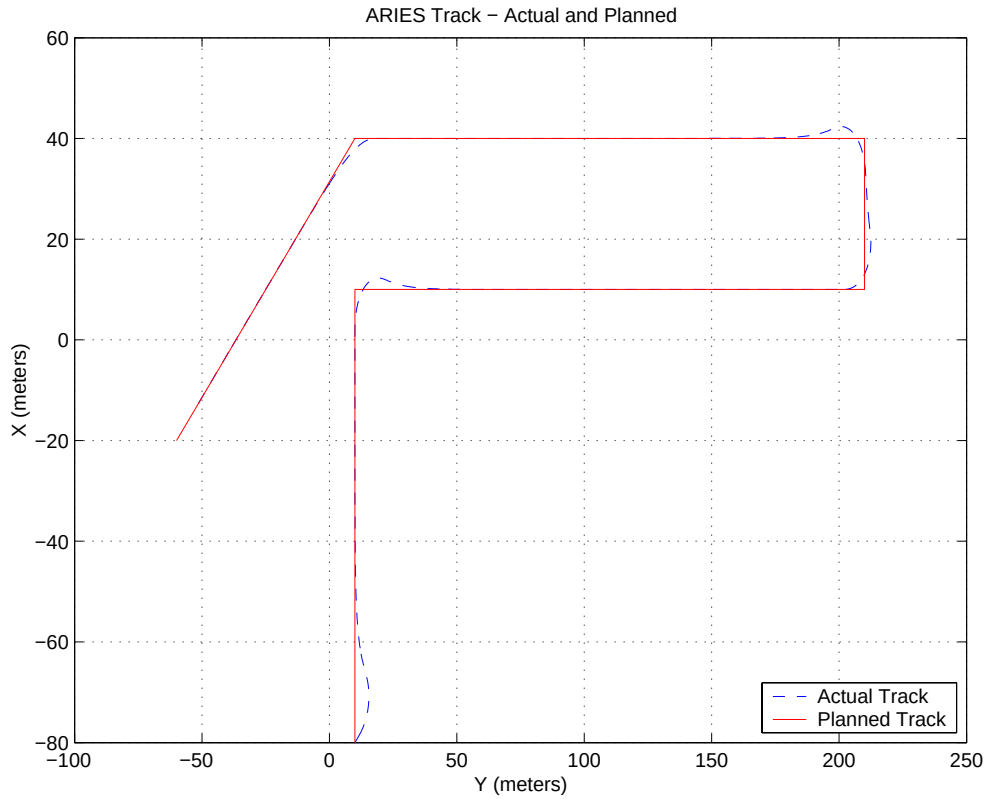

Figure 4. Simulation Results of Heading and CTE Controllers

It can be seen from Figure 4, that as ARIES starts the mission at (X = -80, Y = 10) with an initial heading of 50 degrees, LOS control is utilized until the cross track heading error falls below 40 degrees and CTE control takes over. LOS control is evident at each turn.

THIS PAGE INTENTIONALLY LEFT BLANK

III. TRAJECTORY PLANNING

A. INTRODUCTION AND BACKGROUND

As previously stated in Chapter I, trajectory tracking is defined as getting from point A to point B at a certain time. Fraichard (1993) eloquently described trajectory planning of a robot as the time history of a continuous sequence of configurations between the current configuration of the robot and its goal configuration. He argued that trajectory planning with its time dimension allows accounting of time-dependent constraints such as moving obstacles and the dynamic constraints of the robot. Dynamic constraints of robots are usually taken to be: engine force, sliding (friction between the wheels and ground), and velocity.

With the object of defining a new behavior for the NPS ARIES, that of arriving at a defined place in space and time, this chapter will seek to accomplish the object by first getting ARIES to arrive at a defined place in space and then second, getting ARIES to arrive at a defined time.

B. ARRIVAL AT A DEFINED PLACE IN SPACE

In practice, ARIES will be used to develop a communications server vehicle. While running a pre-programmed mission, she will be expected to receive modem commands from other vehicles that will direct her to a designated rendezvous location and time for file transfer operations. The first step in accomplishing this rendezvous requires ARIES to respond to a short-low bit rate modem command and then go to the rendezvous location regardless of time. This was accomplished by modifying the MATLAB .m file written by Marco (2001).

By writing a break into the .m file, thirty seconds into the simulation, a simulated modem command can be transmitted to ARIES designating a rendezvous position in the form of a one-row track.out file (Table 1). The new track.out file essentially overwrites the original track.out file and the simulation proceeds using the cross track error (CTE) and line of sight (LOS) controllers explained in Chapter II. Figure 5 shows the results of sending the vehicle to rendezvous location of (X = -20, Y = 100) thirty seconds into the

mission of Figure 4. The modifications to the original Marco (2001) MATLAB .m file can be seen in Appendix B.

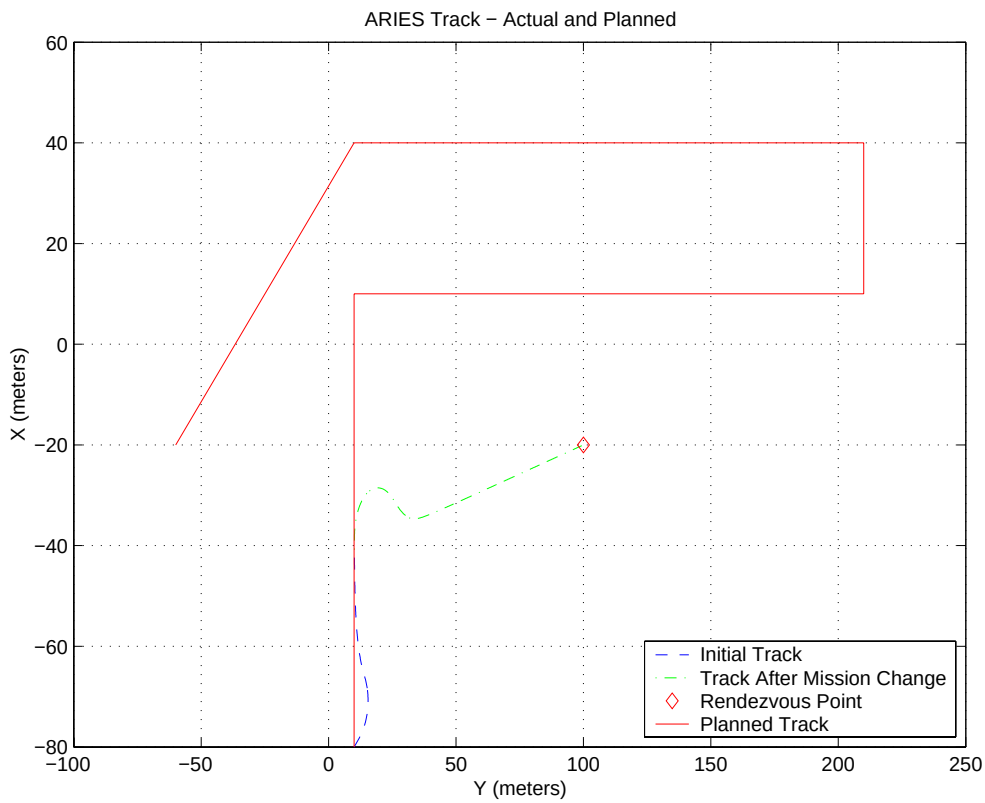

Figure 5: Simulation of a Modem Command Directing ARIES to a Rendezvous Point

C. ARRIVAL AT A DEFINED PLACE IN TIME WITHOUT ACCELERATION

Having previously defined $S(t)_i$ as the distance to the *i*th way point projected to the track line, $S(t)_i$ can be re-characterized as the vehicle path and *s* can simply be defined as the distance traveled along the path. Velocity can now be defined as $\dot{s}$. Using theory developed by Fraichard (1993), the state-time space of ARIES is thus a three-dimensional space ($s\ x\ \dot{s}\ \ \dot{x}\ \ t$), where *t* represents the time dimension. The dynamic constraints of ARIES are transformed into constraints on the velocity, $\dot{s}$, and the acceleration, $\ddot{s}$. The constraints on $\dot{s}$ translate into a velocity limit curve in the ($s\ x\ \dot{s}$) plane.

With only a top speed of 3.5 knots, ARIES is not capable of the large speed changes seen in robots. In order to use velocity to the advantage in this trajectory planning problem, it

is proposed that the velocity limit curve be defined as a step function, i.e., velocity is one of two states: maximum (3.5 knots) or minimum (0.5 knots). Making the assumption that the acceleration is zero further simplifies the problem. The basic uniform motion equations: $s(t) = s_o + vt$ and $v(t) = v$ thus hold true.

The first item that must change in order to solve the rendezvous problem is in regards to the eleven-column track.out file as shown in Table 1. In its current configuration, a rendezvous time is not mentioned. In order to work around this, at the break in the .m file that allows the operator to send a simulated modem command, a one line, twelve-column track.out file is sent instead of the old eleven-column file. The new twelfth column indicates the time to rendezvous in seconds after the modem command.

Now that ARIES has a rendezvous location and time, the mission feasibility must be determined. The mission feasibility is solely based on the velocity constraints of ARIES (since acceleration is assumed to be zero). If the time allotted for the rendezvous is greater than the distance to the rendezvous point divided by the maximum speed (3.5 knots) or if the time allotted for the rendezvous is less than the distance to the rendezvous point divided by the minimum speed (0.5 knots), the mission is deemed feasible and not constrained by the velocity. The distance to the rendezvous point (L_i) is defined by equation 25.

Once the mission has been deemed feasible, the simulation proceeds using the pre-existing CTE and LOS controllers to determine the vehicle's relative position with regard to the rendezvous point. A simple, speed controller is then implemented to control ARIES' speed with respect to time. This is accomplished by determining the overall distance traveled (ODT_i) in a given time step as defined by:

$$ODT_i = L_{i-1} - s_i \tag{37}$$

where s_i is the distance traveled along the path. The time used is simply the distance traveled divided by ARIES' speed, and the time remaining until rendezvous is the given time (column twelve of the track.out file) minus the time used.

If the time remaining until the rendezvous is less than the distance traveled along the path (s) divided by the speed (U), then the vehicle is sped up to the maximum speed

of 3.5 knots. If the time remaining is greater than or equal to $\frac{s}{U}$, then ARIES is slowed to 0.5 knots. Figure 6 is a three-dimensional, time-space plot that shows the results of sending ARIES to the rendezvous location (X = -20, Y = 100, t = 90) thirty seconds into the original mission depicted in Figure 4. Having assumed no acceleration, Figure 7 shows the step function behavior of ARIES' speed.

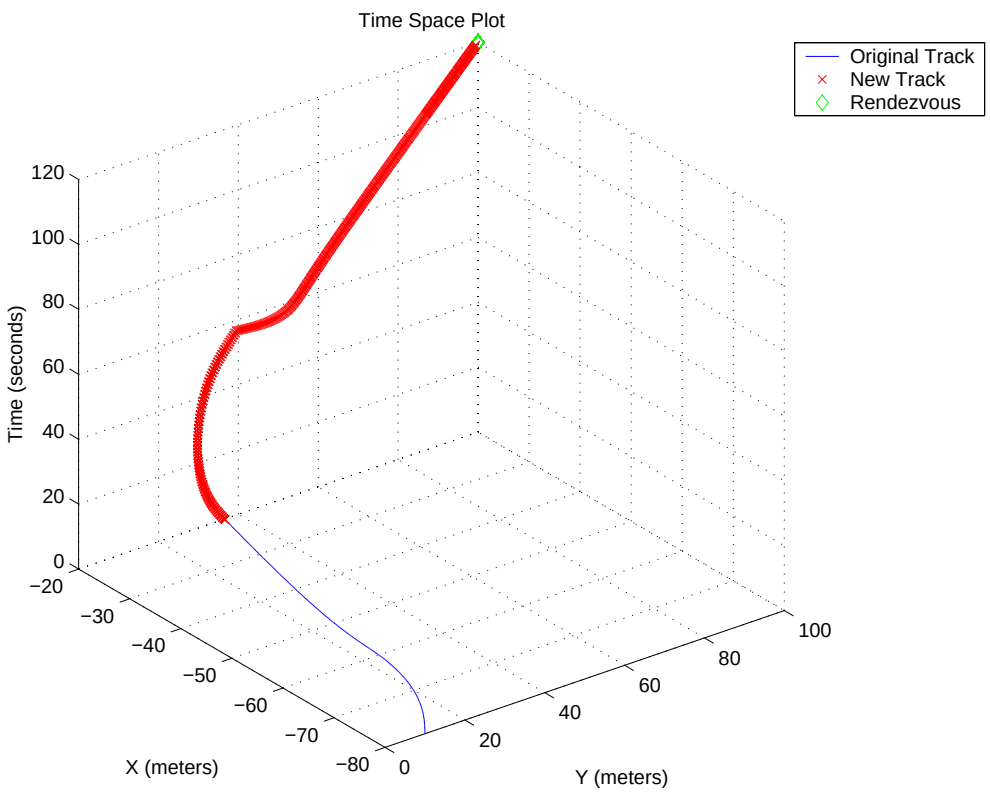

Figure 6: Three Dimensional Time-Space Plot

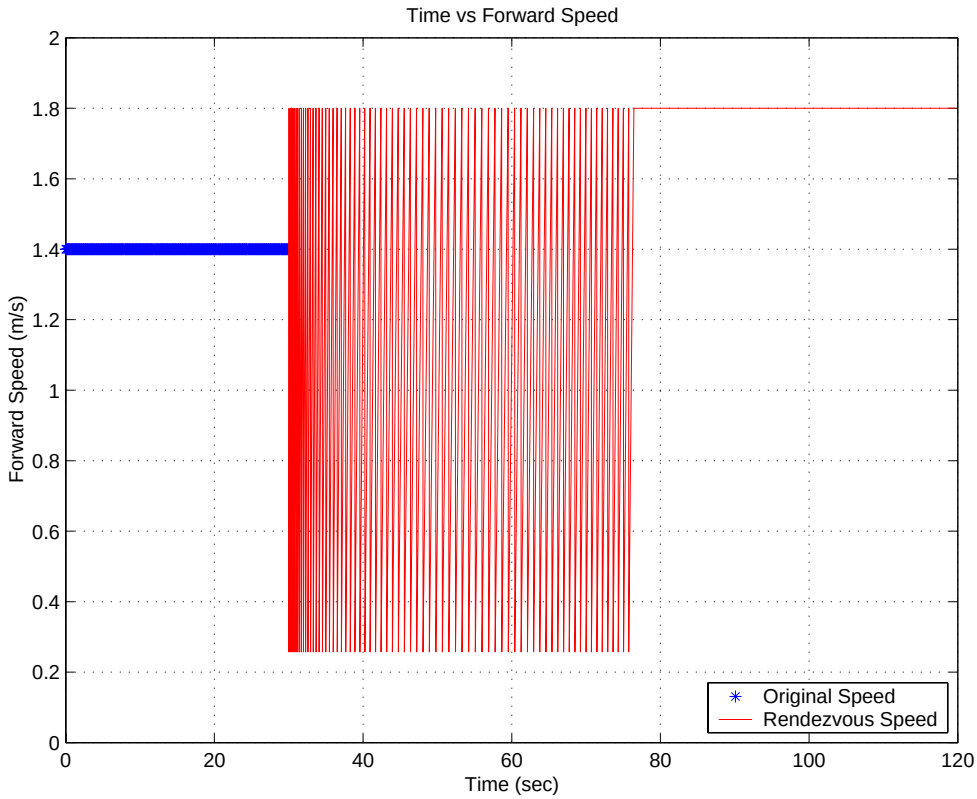

Figure 7: Time-Speed Plot Showing Step Behavior, No Acceleration/Deceleration

This simple speed controller, that assumes uniform motion, solves the problem of having ARIES arrive at a defined time. The assumption of no acceleration, however, is flawed and is now addressed. For that reason, the code utilized to produce Figures 6 and 7 is not included in this work.

D. ARRIVAL AT A DEFINED PLACE IN TIME WITH ACCELERATION

Using data files from three separate missions conducted in the Azores in August 2001, the longitudinal ground speed was plotted against the time in order to determine ARIES' acceleration and deceleration. Figure 8 shows the data from mission file Navd_081102_02.d.

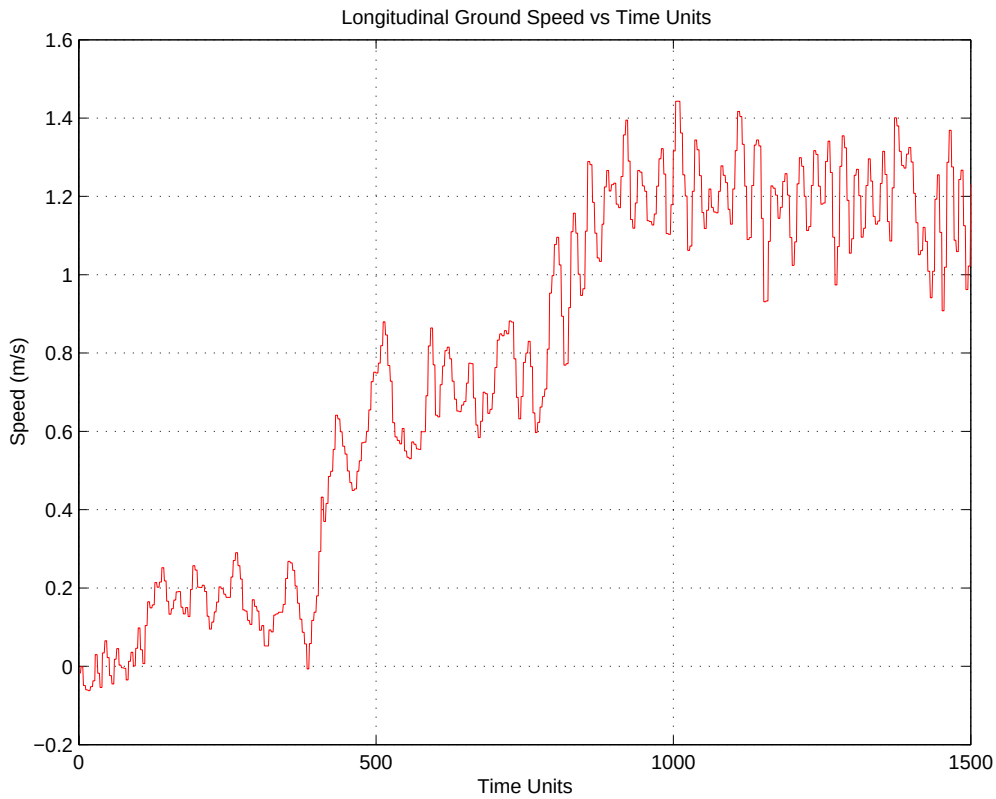

Figure 8: Longitudinal Ground Speed vs Time Units for Azores Mission 081101_02

For this mission, it was estimated that ARIES went from 0 m/s to 1.2 m/s in approximately 856 time units where each time unit is 0.125 seconds. This is an acceleration rate of approximately 0.0112 m/s^2. The other two mission files (Navd_081101_04.d and Navd_081001_03.d) showed approximate accelerations of 0.0152 m/s^2 and 0.0044 m/s^2 respectively. These results led to an estimate of ±0.01 m/s^2 as ARIES' acceleration/deceleration constant for the sole purpose of adding acceleration into the previously designed speed controller. These calculations do not take into consideration that ARIES starts on the surface and comes to its ordered depth.

Now that acceleration is being used, acceleration can be treated as a dynamic constraint. The mission cannot be feasible if the distance to the rendezvous minus the acceleration/deceleration distance is less than the min/max speed multiplied by the time it takes to accelerate/decelerate from initial speed (U) to final speed (U_o). The acceleration/deceleration distance is given by:

$$d = \frac{U^2 - U_o^2}{2a} \tag{38}$$

where a is the acceleration constant.

Once the mission is determined to be feasible, if ARIES needs to increase her speed based on the distance remaining to the rendezvous point and time remaining until the rendezvous time, speed is adjusted as:

$$U_i = U_{i-1} + a_i t_{used,i} \tag{39}$$

where U_i is the new speed, U_{i-1} is the previous speed, a_i is the acceleration constant and $t_{used,i}$ is the time used between (i-1) and i. The maximum U_i is held at 3.5 knots. If ARIES needs to decrease her speed, the following equation is used:

$$U_i = U_{i-1} - a_i t_{used,i} \tag{40}$$

The minimum speed of ARIES is held to 0.5 knots.

Figure 9 shows the effect of deceleration on the time-speed plot for the mission shown in Figure 6.

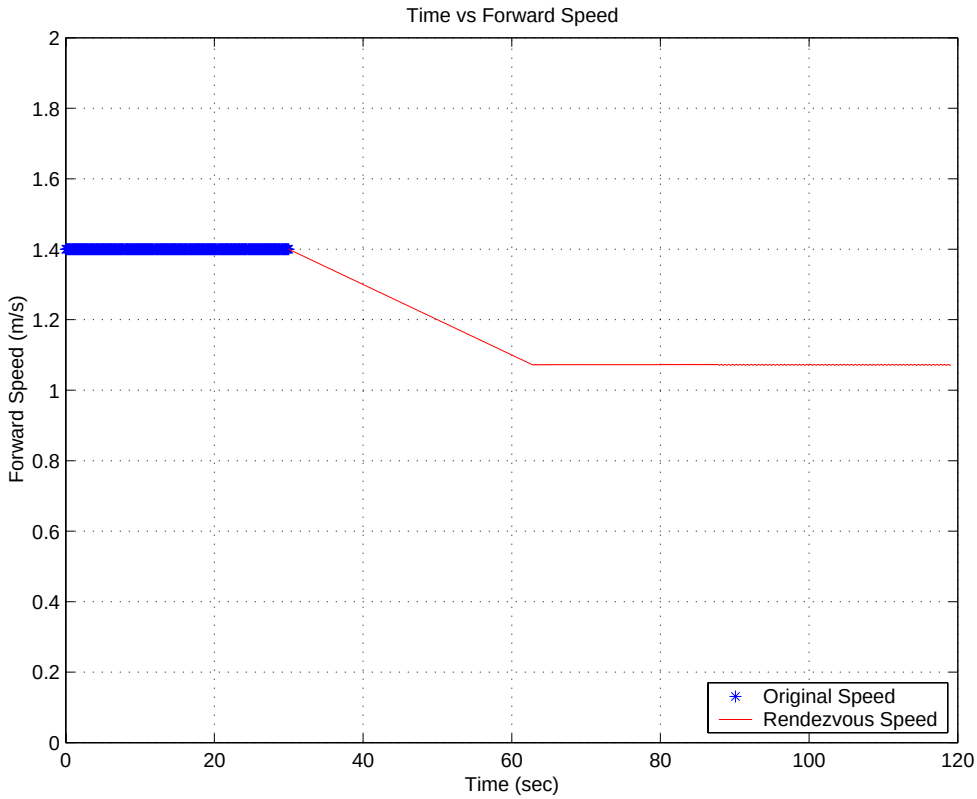

Figure 9: Time-Speed Plot With Deceleration

IV. BUILDING AN ALONG TRACK SPEED (ATS) CONTROLLER

A. THEORY BEHIND THE LONGITUDINAL EQUATION OF MOTION

The use of the basic uniform motion equations: $s(t) = s_o + vt$ and $v(t) = v$ with the addition of acceleration does not hold true for an underwater vehicle. For that reason, the code utilized to produce Figure 9 is not included in this work. Chapter II discussed the equations of motion for the ARIES vehicle. Healey (1995) further derives the surge equation of motion in order to model the longitudinal dynamics of the vehicle. He assumes that the hydrodynamic forces in the surge direction are not generated by lift and are dominated by drag and added mass effects. He also points out that the primary thrust forces for ARIES comes from the propellers. This gives rise to the equation of motion for longitudinal motion:

$$(m - X_{\dot{u}_r})\dot{u}_r = mv_r r + X_{u_r|u_r|}u_r|u_r| + X_{prop} \tag{41}$$

X_{prop} represents the net propulsive force on the vehicle from propeller action. It consists of two parts: the bollard pull force and the loss of thrust caused by the subsequent forward motion of the vehicle. The bollard pull force is equal to $\alpha n|n|$ where α equals $(K_t \rho D^4)$. K_t is the thrust coefficient and is a function of the propeller speed of advance. The loss of thrust caused by the subsequent forward motion of the vehicle is equal to $\gamma n|u|$ where γ equals $\gamma_o \rho D^3$. The coefficient γ_o is the slope of the K_t curve at the particular operating condition of interest. Assuming the lateral, sideslip velocity is negligible simplifies equation 41 yielding the equation of motion for longitudinal motion in a straight line as:

$$(m - X_{\dot{u}_r})\dot{u}_r = X_{u_r|u_r|}u_r|u_r| + \alpha n|n| - \gamma|n|u_r \tag{42}$$

From Johnson (2001), the value of m for ARIES is 222.26 kg, the length (a) is 64 inches and the diameter (b) is 7.53 inches. Based on work by Lamb (1945), for a neutrally buoyant mass and equating the total volume of ARIES to a prolate ellipsoid model while maintaining the overall length results in an a/b ratio of 8.499. Interpolating

Lamb's table results in a k_x value of 0.0266 which results in $X_{\dot{u}_r}$ = 6.79 kg and (m - $X_{\dot{u}_r}$) = 215.47 kg. Solving for $\dot{u}_r$ results in:

$$\dot{u}_r = 0.004641(X_{u_r|u_r|}u_r|u_r| + \alpha n|n| - \gamma|n|u_r) \tag{43}$$

B. PARAMETER IDENTIFICATION

The unknown values in equation 43 are $X_{u_r|u_r|}$, α, and γ. In order to successfully identify the unknown coefficients it is common practice to accurately model the ARIES control and response. By manipulating the equation of motion, we can let system parameters become unknowns and the variables, as measured, to be known. Data gathered by ARIES from her onboard sensors is sampled at eight Hertz and is therefore not continuous. This calls for the use of a discrete time model. In this case, $\dot{u}_r(t) = \frac{u_r(t+1) - u_r(t)}{\Delta t}$ where Δt equals 0.125 seconds. This simplifies equation 43 to:

$$\dot{u}_r(t+1) - u_r(t) = 0.004641\Delta t(X_{u_r|u_r|}u_r(t)|u_r(t)| + \alpha n(t)|n(t)| - \gamma|n(t)|u_r(t)) \tag{44}$$

Equation 44 can be cast into matrix form: $\boldsymbol{y(t)} = \boldsymbol{H(t)\Theta(t)}$ where $\boldsymbol{y(t)}$ = $\dot{u}_r(t+1) - u_r(t)$, $\boldsymbol{H(t)}$ is the n x 3 matrix of measurements relating to the output (i.e. $n(t)$ and $u_r(t)$) and $\boldsymbol{\Theta(t)}$ is the 3 x 1 matrix of coefficients ($X_{u_r|u_r|}$, α, and γ).

As Johnson (2001) pointed out, using a least squares method to estimate the parameters $\boldsymbol{\Theta(t)}$, which are never exactly known, results in minimizing the difference between the actual parameter and its estimate. The difference is known as the equation error and it is defined as:

$$e(t) = (y(t) - H(t)\hat{\Theta}(t)) \tag{45}$$

where $\hat{\Theta}(t)$ is the estimate of $\Theta(t)$. For this particular case, it involves using the data obtained from the vehicle sensors combined with the equation 44. Johnson (2001) further points out that in order to minimize the error, define the scalar positive squared error measure, $J(n) = \sum_{t=1}^{n} e'(t)e(t)$ then the minimization of J is given by:

$$\frac{dJ}{d\hat{\Theta}} = 0 = -\sum_{t=1}^{n} H^{'}(t)e(t) \tag{46}$$

and substituting ***y(t)*** = ***H(t)Θ(t)*** produces:

$$0 = -\sum_{t=1}^{n} H^{'}(t)(y(t) - H(t)\hat{\Theta}(t)) \tag{47}$$

Rearranging this into matrix form and solving for $\hat{\Theta}(t)$ gives:

$$\hat{\Theta} = [H^{'}H]^{-1}H^{'}y \tag{48}$$

Using the matrix divide function in MATLAB to evaluate equation 48 will produce a result that is the least squares fit for $\hat{\Theta}(t)$.

Gelb (1974) points out that the same result may be found using the gaussian random assumptions in which the solution, $\hat{\Theta}(t)$, in equation (47) is the most likely solution in which its probability is maximum. It should be noted that the regression matrix, $[\sum_{t=1}^{n} H^{'}(t)H(t)]$, must be positive and strong (with no singularity) otherwise its inverse does not exist. This means that the system must be perpetually excited by its input.

C. DEFINING THE COEFFICIENTS FOR THE LONGITUDINAL EQUATION OF MOTION

In order to gather the required output data (*n(t)* and $u_r(t)$) to solve equation 44 by the method of least squares, an experiment was planned. ARIES was programmed with the track.out file listed in Table 2.

Table 2. track.out file for 17 April 2002 Experiment

1	2	3	4	5	6	7	8	9	10	11
800.00	300.00	3.25	3.25	0	8.0	2.0	1	25.0	10.00	40.0
900.00	300.00	3.25	3.25	0	8.0	2.0	0	25.0	10.00	150.0
1000.00	300.00	2.50	2.50	0	8.0	2.0	0	25.0	10.00	150.0
1100.00	300.00	3.25	3.25	0	8.0	2.0	0	25.0	10.00	150.0
1200.00	300.00	2.50	2.50	0	8.0	2.0	0	25.0	10.00	150.0
1300.00	300.00	3.25	3.25	0	8.0	2.0	0	25.0	10.00	150.0
1300.00	350.00	3.25	3.25	0	8.0	2.0	0	25.0	10.00	75.0
1200.00	350.00	3.25	3.25	0	8.0	2.0	0	25.0	10.00	150.0

1100.00	350.00	2.50	2.50	0	8.0	2.0	0	25.0	10.00	150.0
1000.00	350.00	3.25	3.25	0	8.0	2.0	0	25.0	10.00	150.0
900.00	350.00	2.50	2.50	0	8.0	2.0	0	25.0	10.00	150.0
800.00	350.00	3.25	3.25	0	8.0	2.0	0	25.0	10.00	150.0

The values in the columns are described as follows:

Column #	Description
1	*X* position way point (meters)
2	*Y* position way point (meters)
3	Left screw command speed (volts)
4	Right screw command speed (volts)
5	Control mode flag, 0 = Depth Control, 1 = Altitude Control
6	Commanded Altitude (feet, if applicable)
7	Commanded Depth (feet, if applicable)
8	Perform GPS popup on this track? 1 = Yes, 0 = No
9	Duration of GPS popup (seconds)
10	Watch Radius, $R_{w(i)}$ (meters)
11	Way point timeout. (seconds)

The mission described in Table 2 starts ARIES out at a speed of approximately 1.8 m/s. After conducting a GPS popup, the vehicle runs for approximately 100 meters at this speed and then the speed is reduced to approximately 1.4 m/s for the next 100 meters. ARIES is then accelerated to 1.8 m/s for the next 100 meters. This pattern is continued until reaching the turn around point. Upon turning around, the vehicle continues the speed increase/decrease pattern until the end point. In this configuration, four sets of acceleration/deceleration data can be obtained.

The experiment was successfully conducted in Monterey Bay on 17 April 2002. In total, three data runs were conducted. In the third run, the right and left screw command voltages were slightly modified due to the first two runs showing that the voltages were not matched. The track.out file used in run three is shown in Table 3.

Table 3. track.out file for Run 3

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
800.00	300.00	2.8	2.8	0	8.0	2.0	0	25.0	10.00	40.0
900.00	300.00	2.8	2.8	0	8.0	2.0	0	25.0	10.00	150.0
1000.00	300.00	2.40	2.40	0	8.0	2.0	0	25.0	10.00	150.0
1100.00	300.00	2.8	2.8	0	8.0	2.0	0	25.0	10.00	150.0
1200.00	300.00	2.40	2.40	0	8.0	2.0	0	25.0	10.00	150.0
1300.00	300.00	2.8	2.8	0	8.0	2.0	0	25.0	10.00	150.0
1300.00	350.00	2.8	2.8	0	8.0	2.0	0	25.0	10.00	75.0

1200.00	350.00	2.8	2.8	0	8.0	2.0	0	25.0	10.00	150.0
1100.00	350.00	2.50	2.50	0	8.0	2.0	0	25.0	10.00	150.0
1000.00	350.00	2.8	2.8	0	8.0	2.0	0	25.0	10.00	150.0
900.00	350.00	2.50	2.50	0	8.0	2.0	0	25.0	10.00	150.0
800.00	350.00	2.8	2.8	0	8.0	2.0	0	25.0	10.00	150.0

Figure 10 shows the planned and actual tracks and Figure 11 shows the changes in longitudinal speed during the three runs.

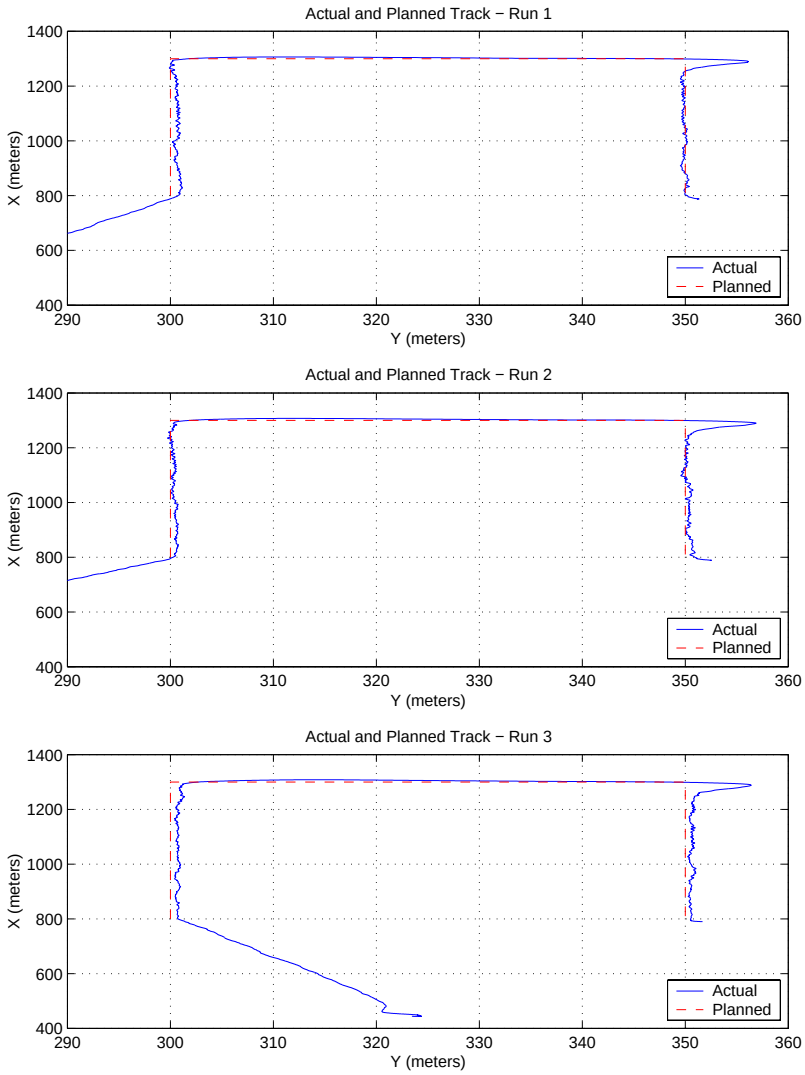

Figure 10: Actual and Planned Tracks of 17 April 2002 Experiment

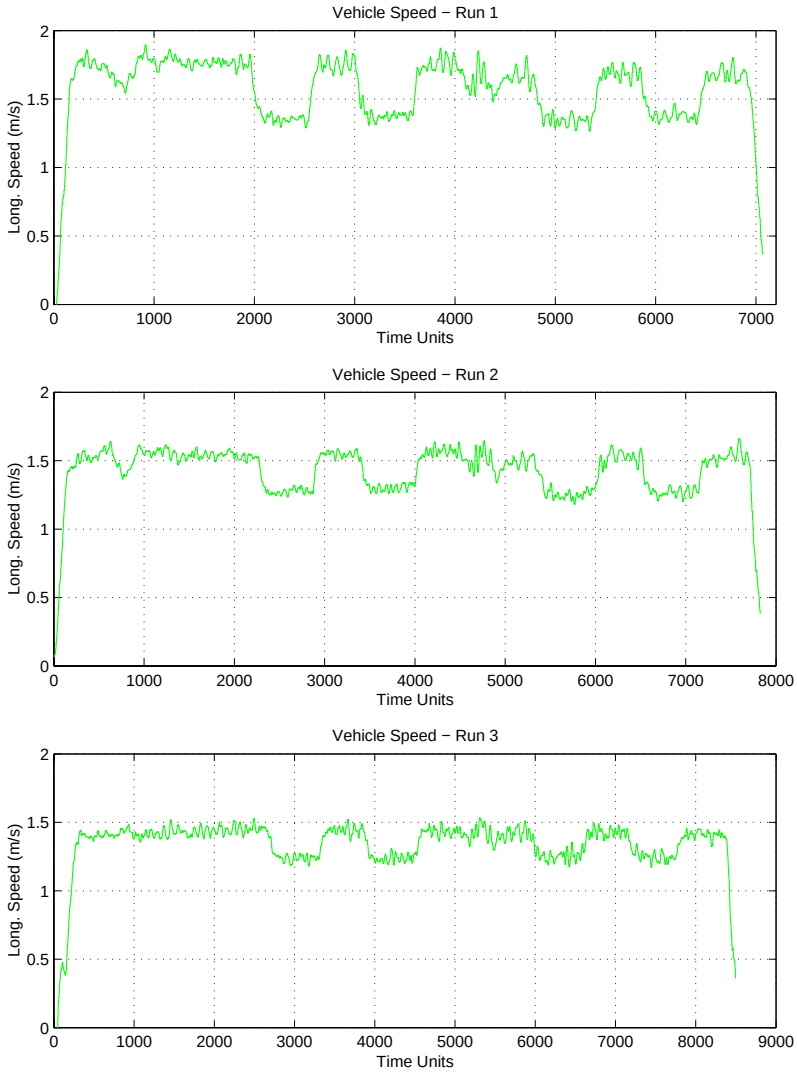

Figure 11: Speed Plots for 17 April 2002 Experiment

With the data gathered from this experiment, the MATLAB file shown in Appendix C, coefficients.m, was utilized to determine the unknown coefficients, $X_{u_r|u_r|}$, α, and γ. It should be noted that during this process it was discovered that previous work had identified the propeller rotation rate in rotations per minute (rpm) was calculated by multiplying the input voltage by a factor. The factor for the port propeller is 133.8047 and for the starboard propeller the factor is 124.4615. Looking at the technical

specifications provided by the manufactor, Tecnadyne Corporation, a factor of 266 should be used to determine the number of rpm for the propellers. Using the 266 factor for determining propeller speed based on input voltage, coefficients.m determined that $X_{u_r|u_r|}$ = -15.681, α = 0.079, and γ = 1.247. Plugging these values into equation 44 and using the propeller speed data obtained from the experiment the accuracy of the coefficients could be determined. Figure 12 shows the results of the model in relation to the actual longitudinal speed.

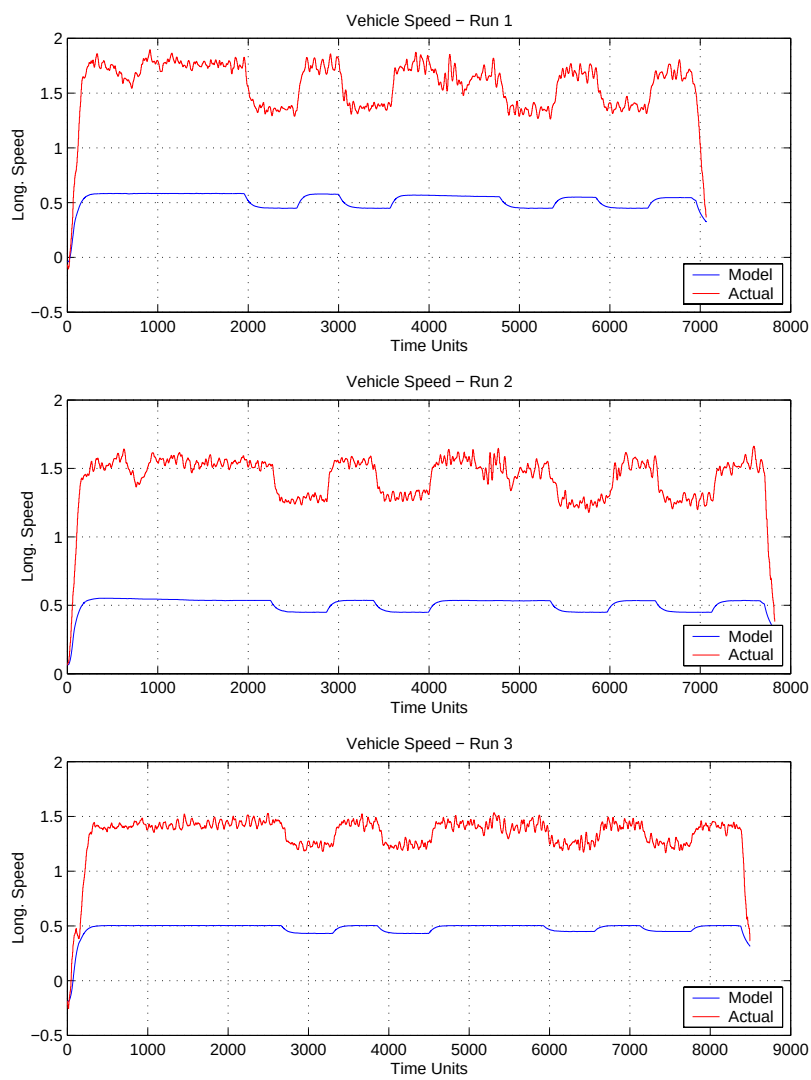

Figure 12: Model and Actual Vehicle Speed

D. COEFFICIENT IDENTIFICATION FROM THE RATIONAL APPROACH

As can be seen from Figure 12, the coefficients generated by the MATLAB file, coefficients.m, do not accurately model the data. Assuming ARIES has a drag coefficient (C_D) of 0.1, an area of 0.103 m^2 and the density of seawater (ρ) is 1025 $\frac{kg}{m^3}$, then $X_{u_r|u_r|}$ can be calculated using the formula:

$$X_{u_r|u_r|} = \frac{1}{2}\rho C_D A \tag{49}$$

The result is $X_{u_r|u_r|} \approx$ -5.25. This value of $X_{u_r|u_r|}$ translates into a force of approximately 16.35 N (3.68 lbf) when the vehicle is traveling at 1.76 m/s. This results in a typical K_T value for AUV/ROV propellers of about 0.4. This value of K_T requires the value of α = 0.0793. Lastly by determining that the thrust reduction factor was the determining coefficient in the model, γ= 0.05 was chosen. Plugging these new values ($X_{u_r|u_r|}$ = -5.25, α = 0.0793, and γ= 0.05) into equation 44 and using the propeller speed data obtained from the experiment the accuracy of these new coefficients could be determined. Figure 13 shows the results of the model in relation to the actual longitudinal speed.

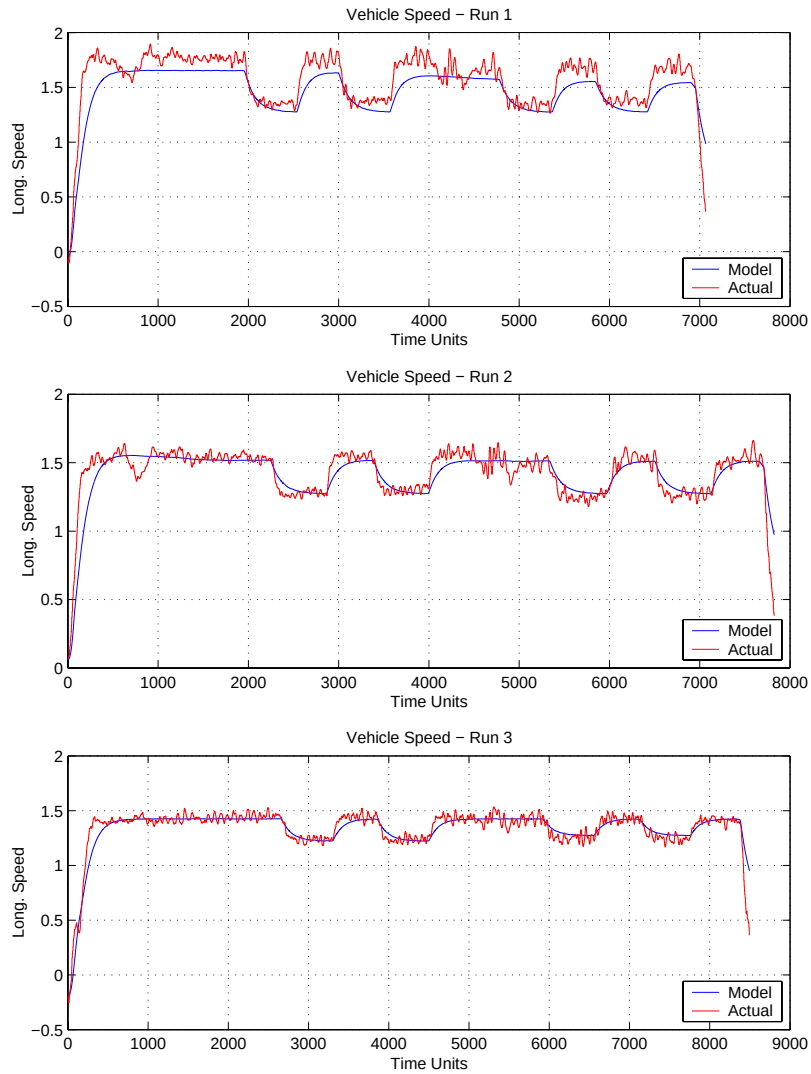

Figure 13. New Model and Actual Vehicle Speed

Although the results shown in Figure 13 look accurate for predicting the speed of ARIES, giving further thought to the value chosen for the drag coefficient, C_D, it was determined that with the various fins on the body of ARIES, a $C_D = 0.1$ was too small and a $C_D = 0.2$ was more appropriate. This change results in $X_{u_r|u_r|} \approx -10.5$. This value of $X_{u_r|u_r|}$ translates into a force of approximately 32.7 N (7.36 lbf) when the vehicle is traveling at 1.76 m/s. This changes the value for K_T to 0.86, which is closer to the value provided by Tecnadye of about 1.16. This value of K_T requires the value of $\alpha = 0.155$.

The γ= 0.05 was not changed. Plugging these new values ($X_{u_r|u_r|}$ = -10.5, α = 0.155, and γ = 0.05) into equation 44 and using the propeller speed data obtained from the experiment the accuracy of these new coefficients could be determined. Figure 14 shows the results of the model in relation to the actual longitudinal speed.

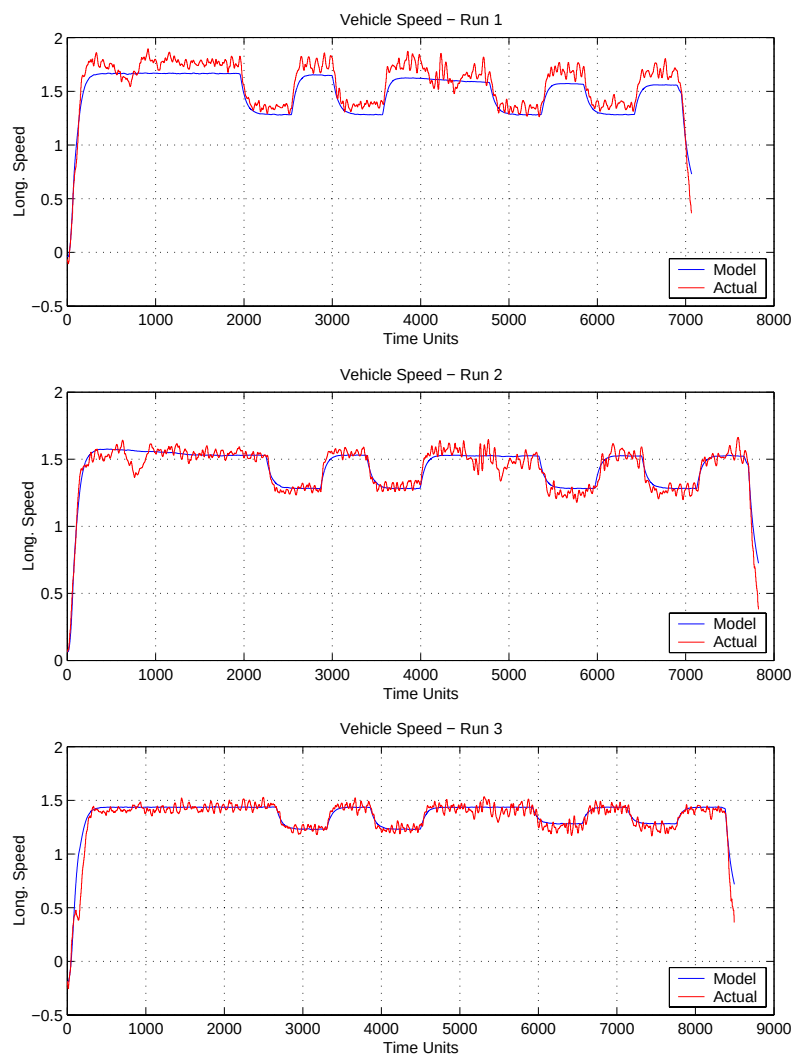

Figure 14: Best Model and Actual Vehicle Speed

It can be seen from Figure 14, that this model is more accurate than the previous model in predicting the actual speed measured by ARIES for these three data runs. The difficulty in obtaining the coefficients for the longitudinal equation of motion can best be explained by the very small operating range of the experiment. The speed range was at

the most, approximately 0.5 m/s. This translates into a very small region on the standard Thrust and Torque Coefficients versus Angle of Attack chart (Lewis, 1988). Final coefficients are summarized in Table 4 shown below:

Table 4. Summary of Coefficient Values

$X_{u_r\|u_r\|}$	-10.5
α	0.155
γ	0.05

E. THE SPEED CONTROLLER

With the coefficients for the longitudinal equation of motion identified, a sliding mode controller was developed based on the work of Healey and Lienard (1993). Rewriting equation 44 with the coefficients identified yields:

$$\dot{u}_r(t) = 0.004641\Delta t(-10.5u_r(t)|u_r(t)| + 0.155n(t)|n(t)| - 0.05|n(t)|u_r(t)) \tag{50}$$

Considering the $-0.05|n(t)|u_r(t)$ as a disturbance, equation 50 can be rewritten as:

$$\dot{u}_r(t) = 0.004641\Delta t(-10.5u_r(t)|u_r(t)| + 0.155n(t)|n(t)|) \tag{51}$$

Choosing $\sigma(t) = u_r(t) - u_{com}(t)$ as the sliding surface, where u_{com} is the desired vehicle speed, the control law in terms of the command for *n(t)*, propeller speed, is found from:

$$\dot{\sigma}(t) = -\eta\,\mathbf{tanh}(\sigma(t)/\phi) \tag{52}$$

This results in the control law being defined as:

$$n(t) = \sqrt{1.39(\dot{u}_{com}(t) - \eta\,\mathbf{tanh}(\sigma/\phi)) + 67.74u_r(t)|u_r(t)|} \tag{53}$$

The propeller-speed command thus arises from one term to accelerate the vehicle, one term to stabilize the motion and one term to overcome the vehicle forward drag. If a nonzero acceleration is required, $\dot{u}_{com}(t)$ is used, otherwise for conditions where it is required to 'regulate' speed, $\dot{u}_{com}(t)$ is held to zero.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RESULTS

A. DIFFERENT TIME COMBINATIONS AT SAME [X,Y] POSITION

The control law, equation 53, and the longitudinal equation of motion, equation 50, were inserted in the MATLAB file called finalrendezvous.m (Appendix D) in place of the pre-existing, rudimentary speed controller. Choosing a $\phi = 0.1$, a $\eta = 800$ and a rendezvous position of [-20, 100] at a time of 70 seconds produced Figures 15, 16 and 17.

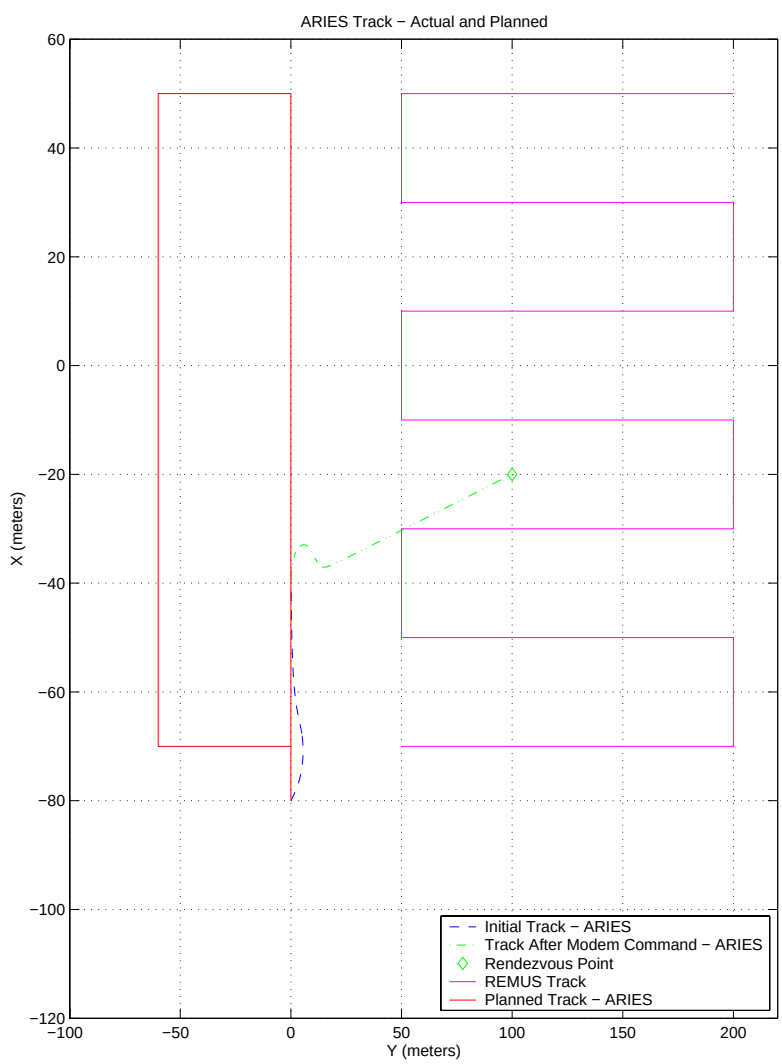

Figure 15: X-Y Plot for [-20, 100] Rendezvous Location

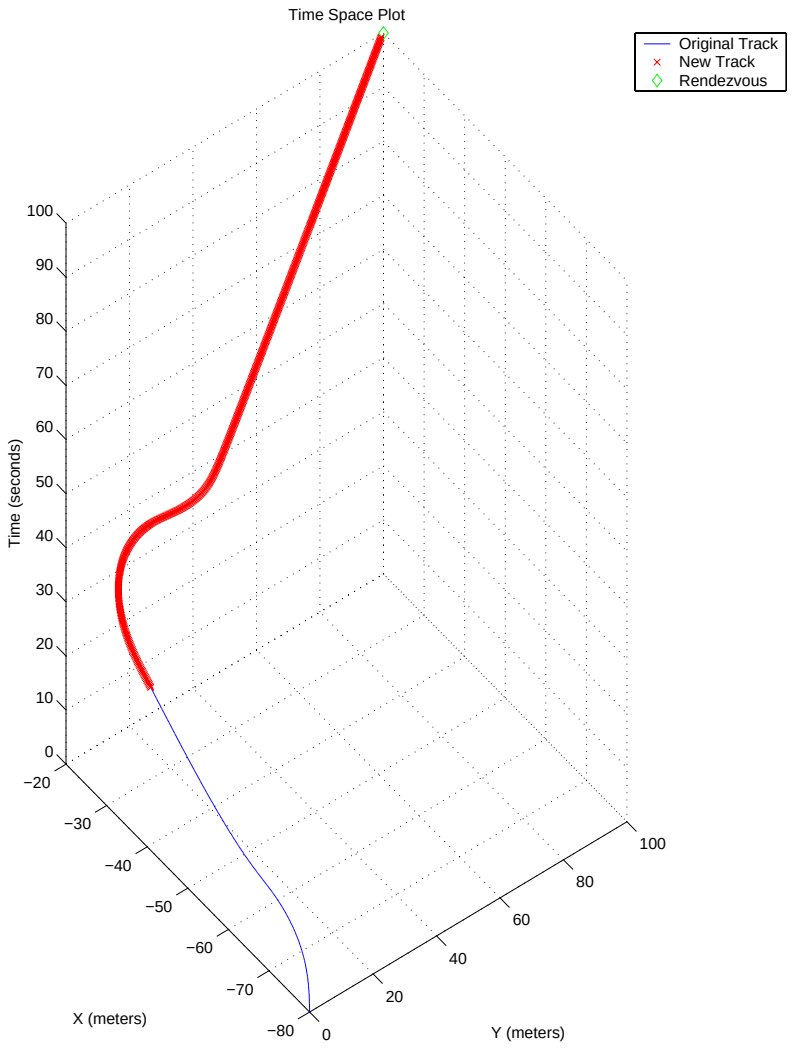

Figure 16: 3-D Plot for 70 Second Rendezvous Time

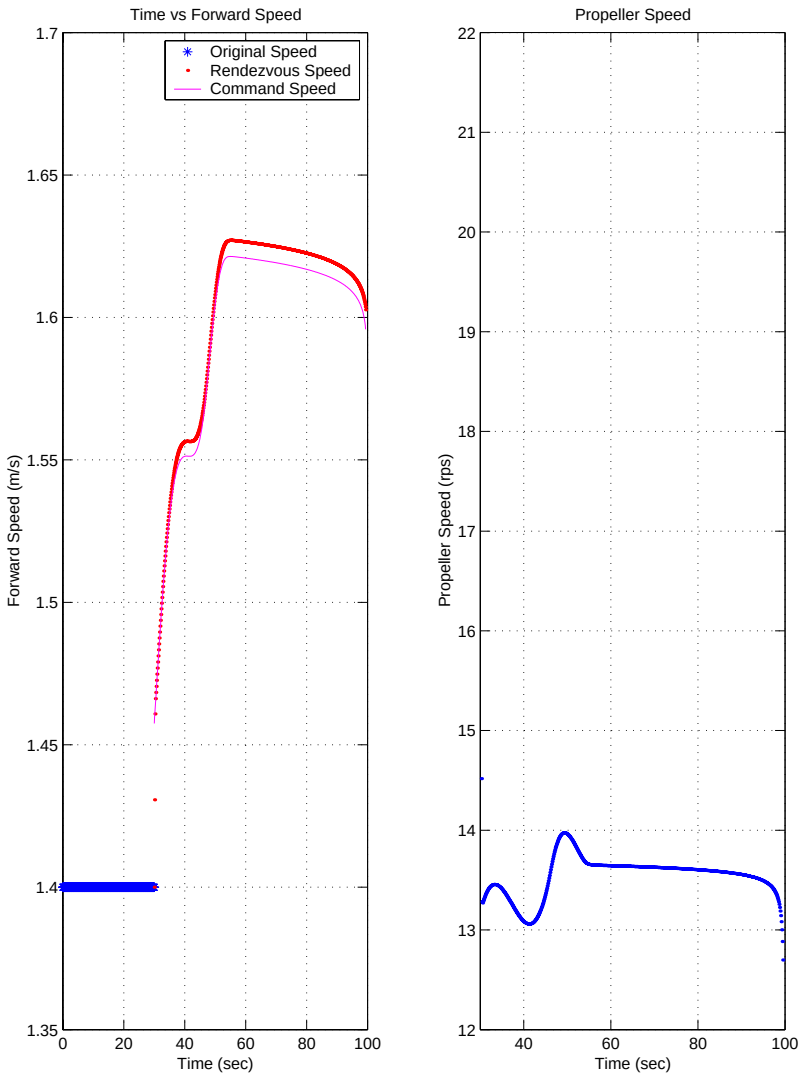

Figure 17: Speed vs Time for 70 Second Rendezvous Time and Propeller Speed vs Time

The above figures show that thirty seconds into ARIES' mission, the REMUS requests a rendezvous at location [-20, 100] in 70 seconds and ARIES makes the rendezvous. The speed controller commands ARIES to speed up from her initial cruising speed of 1.4 m/s in order to achieve the rendezvous at the designated time. The propeller speed (shown in the plot to the right) adjusts as necessary.

Keeping the same rendezvous location of [-20, 100] but changing the rendezvous time to 90 seconds and 120 seconds results in Figures 18 through 21.

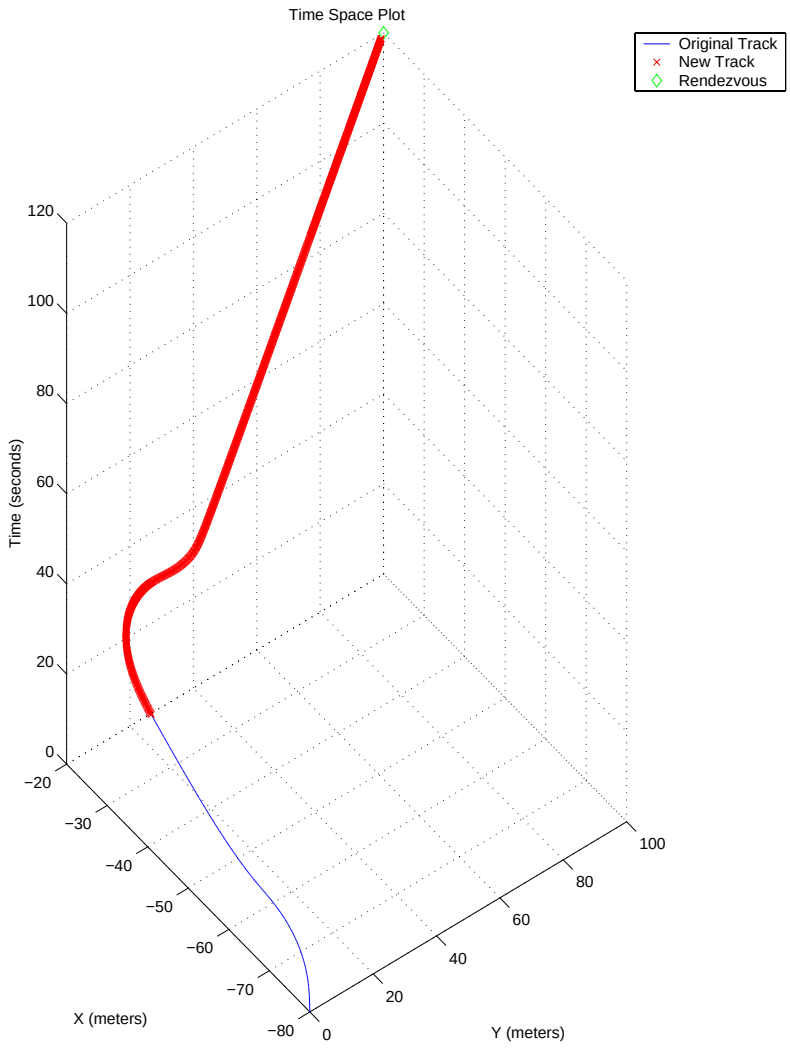

Figure 18: 3-D Plot for 90 Second Rendezvous Time

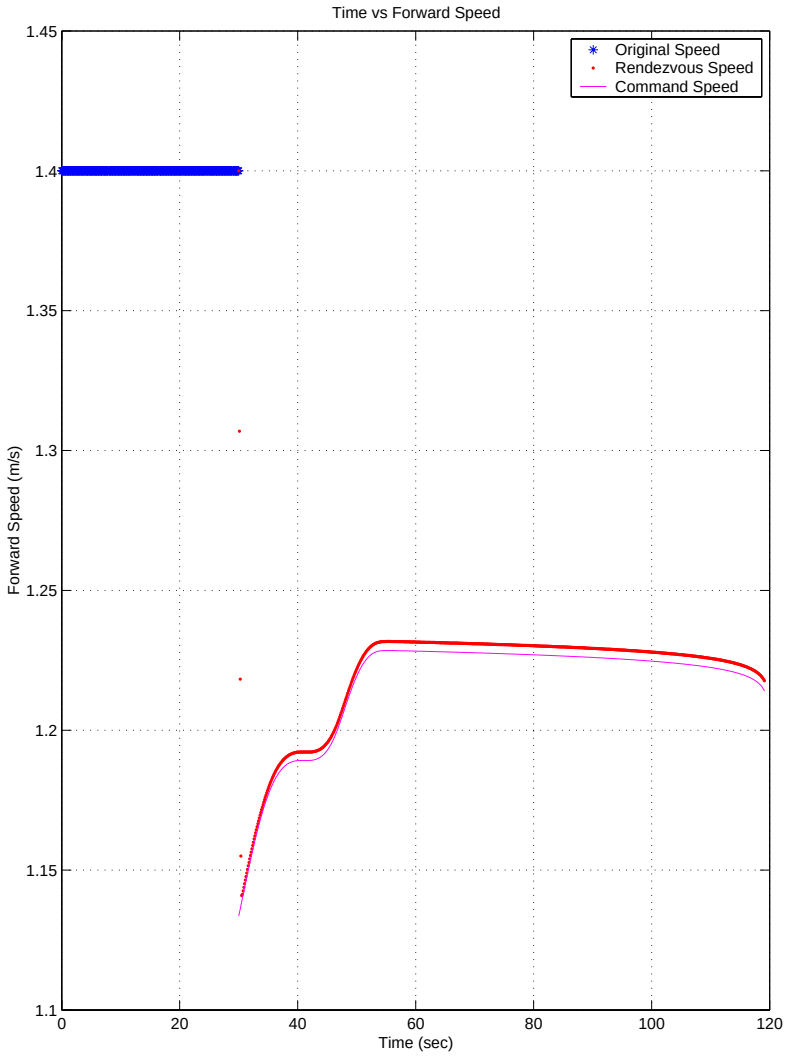

Figure 19: Speed vs Time for 90 Second Rendezvous Time

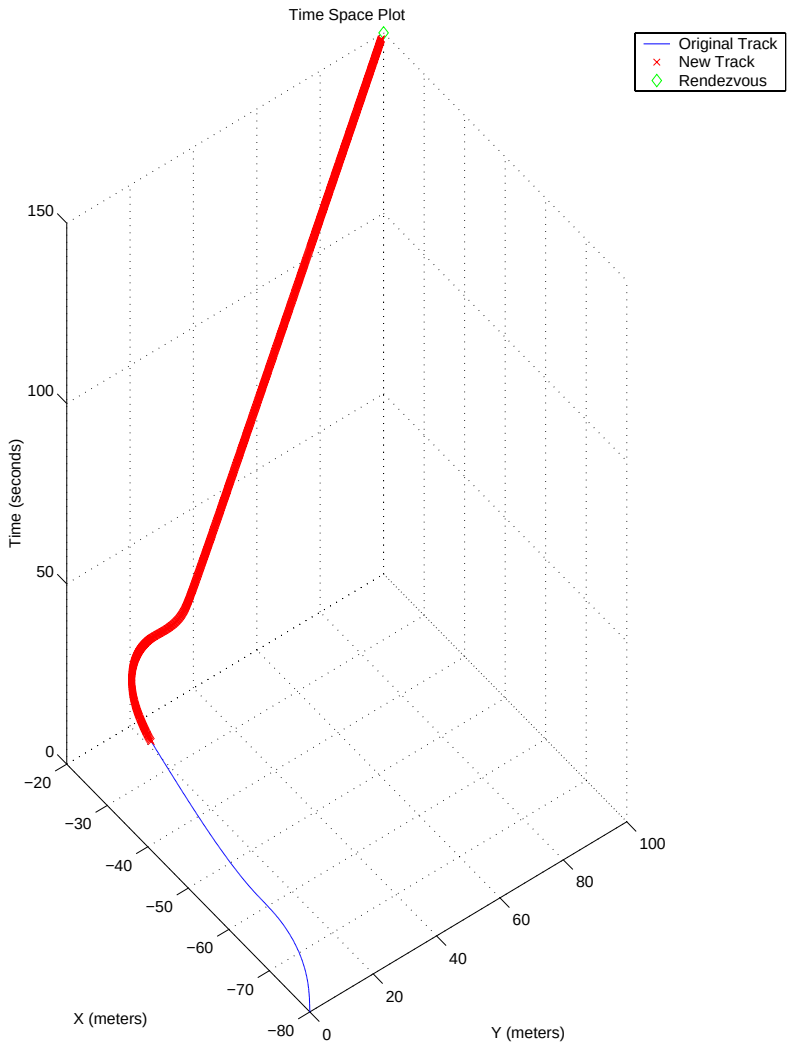

Figure 20: 3-D Plot for 120 Second Rendezvous Time

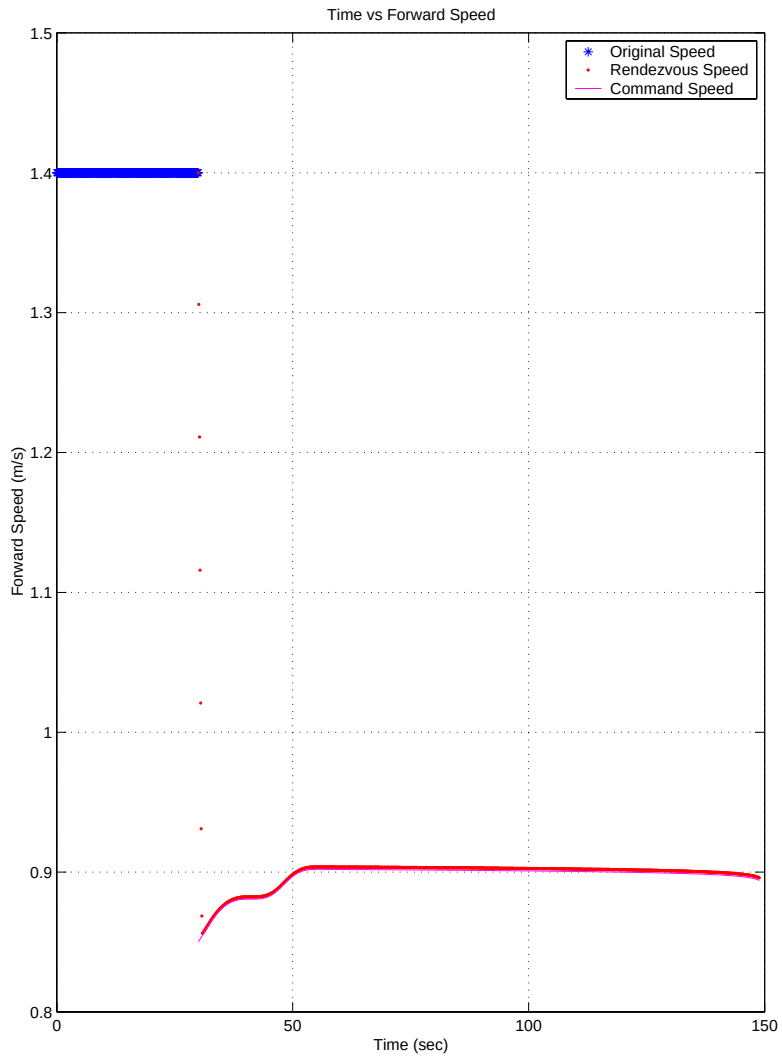

Figure 21: Speed vs Time for 120 Second Rendezvous Time

The above four figures show that by increasing the rendezvous time by twenty seconds, the speed controller responds by commanding ARIES to slow down in order to reach the rendezvous location at the designated time.

B. SAME TIME COMBINATIONS AT DIFFERENT [X,Y] POSITION

By holding the time of rendezvous constant at 120 seconds and changing the rendezvous position from [-20, 100] to [-100, 150] and [50, 50], Figures 22 through 27 were produced.

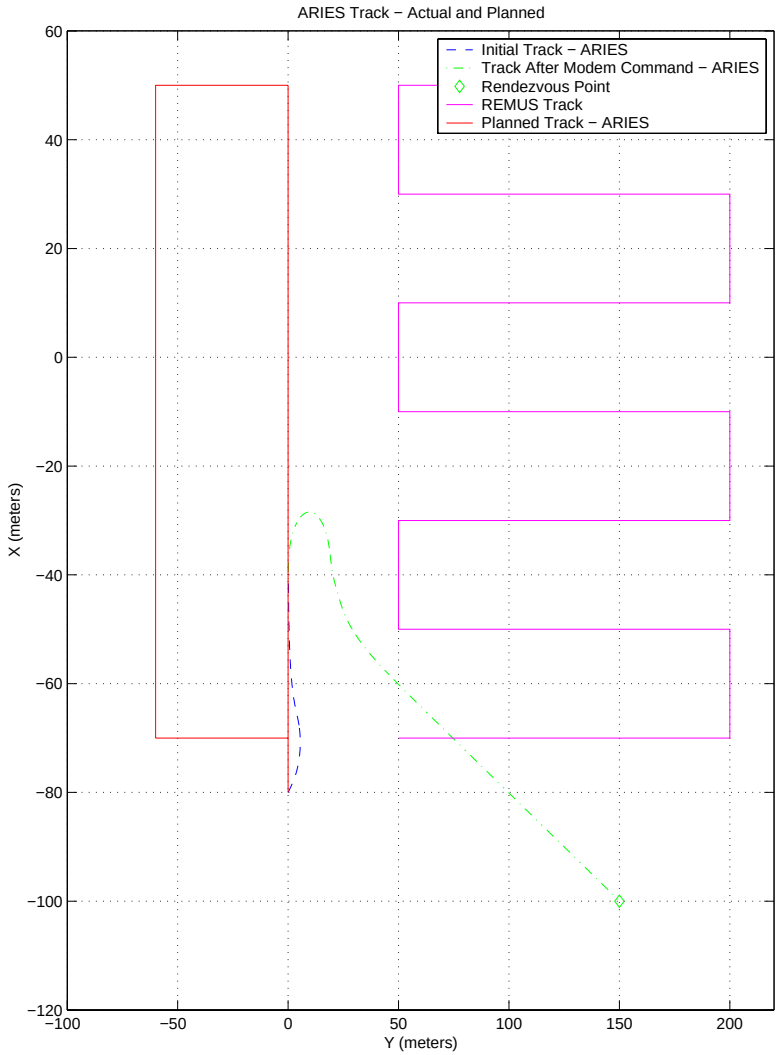

Figure 22: X-Y Plot for [-100, 150] Rendezvous Location

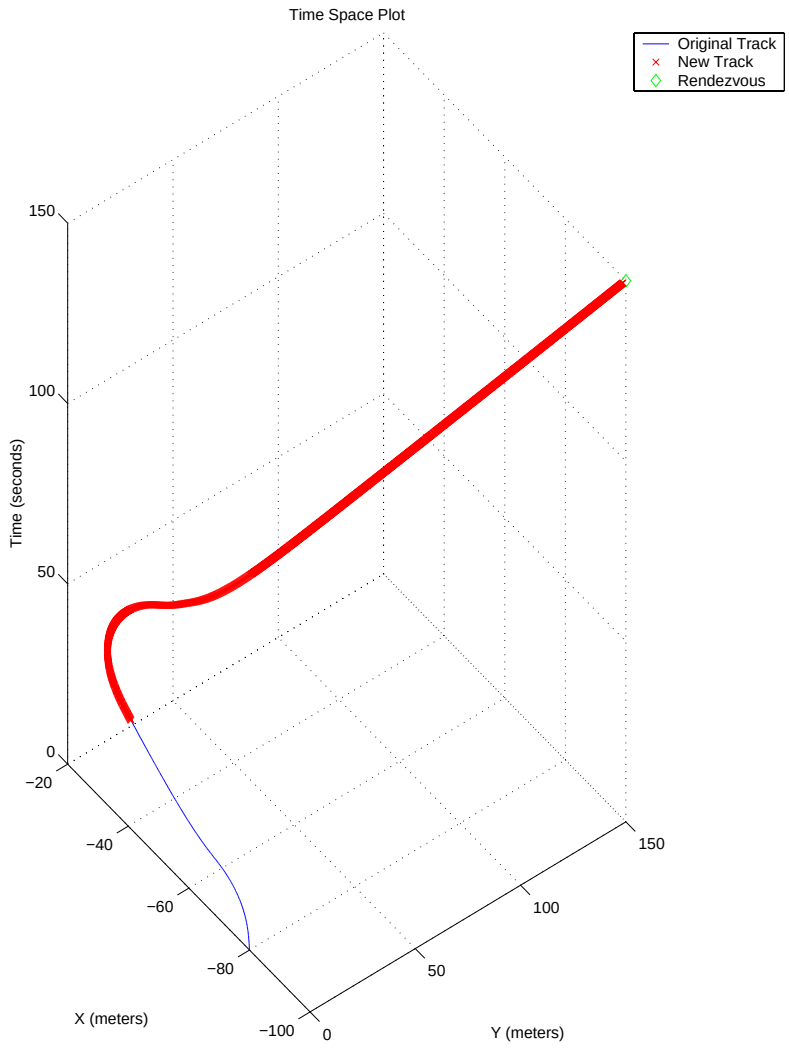

Figure 23: 3-D Plot for [-100, 150] Rendezvous Location

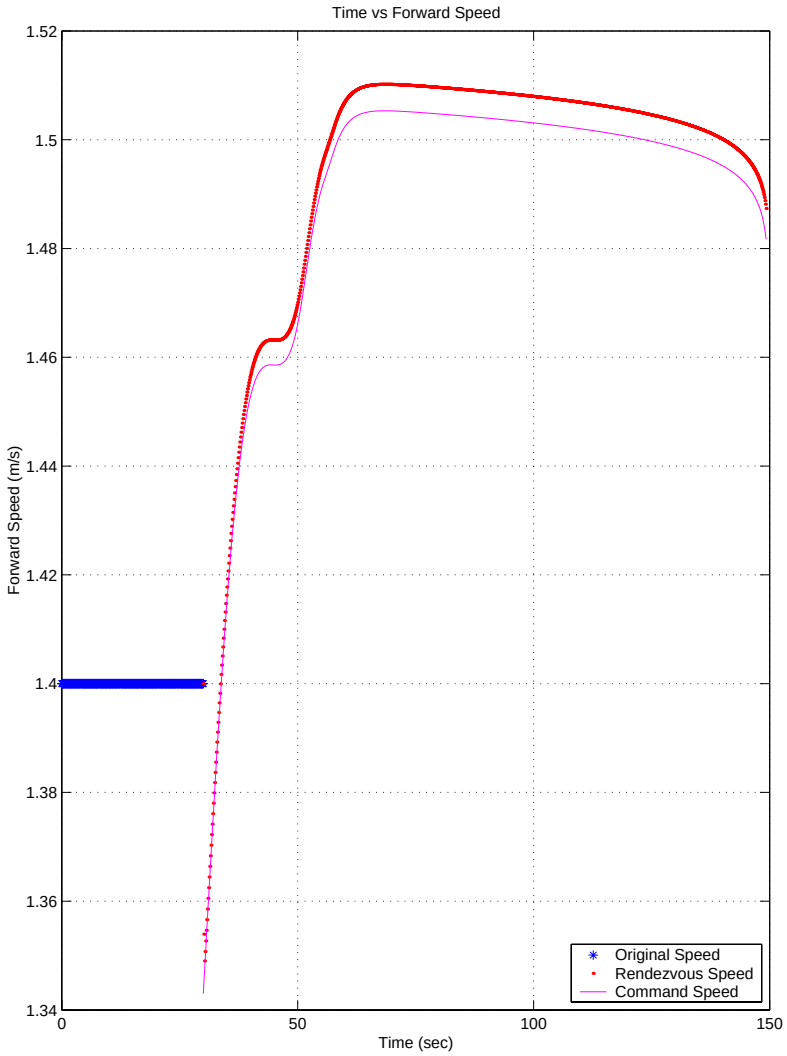

Figure 24: Speed vs Time for [-100, 150] Rendezvous Location

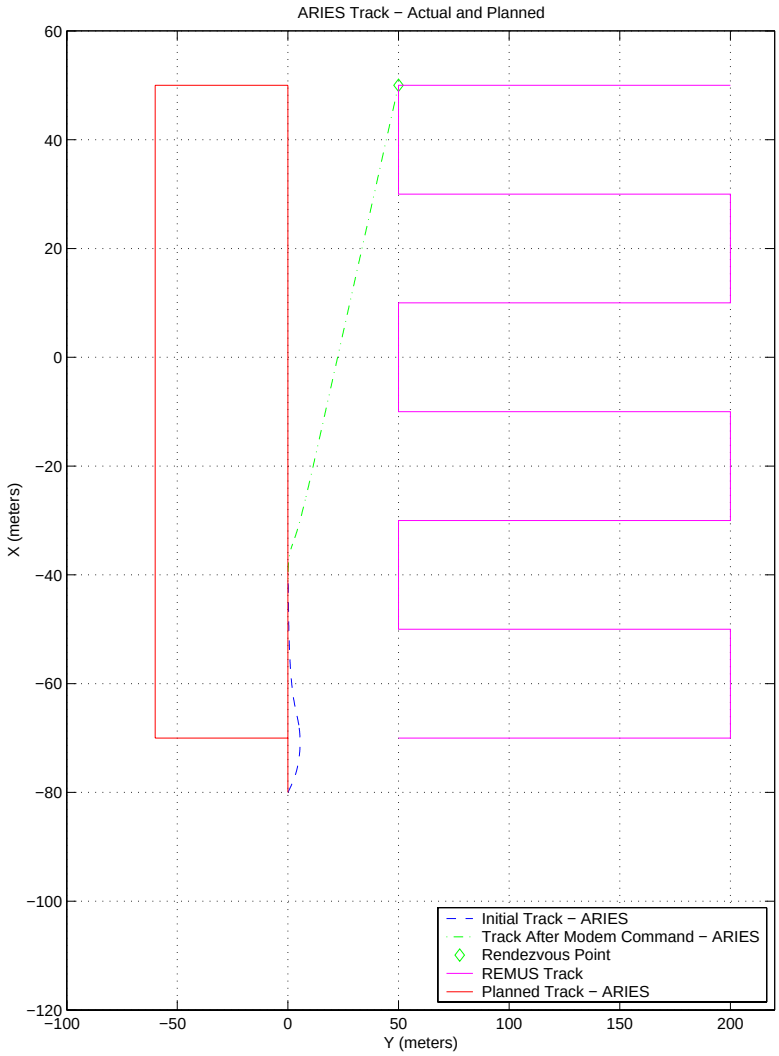

Figure 25: X-Y Plot for [50, 50] Rendezvous Location

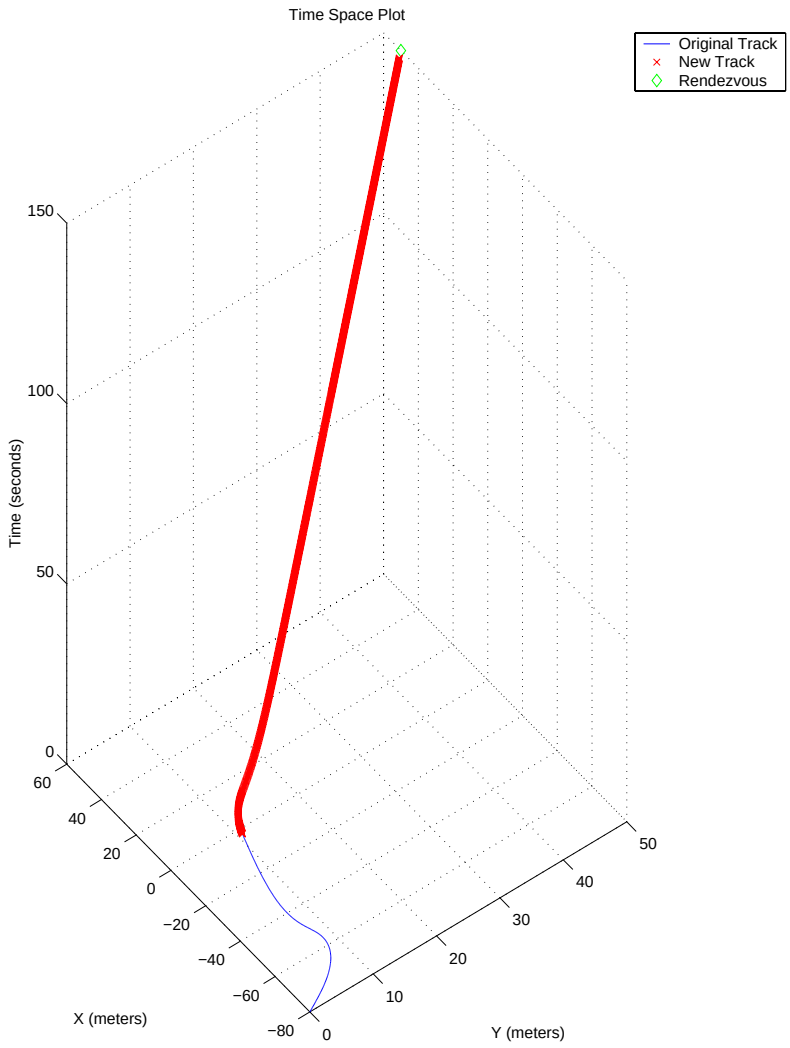

Figure 26: 3-D Plot for [50, 50] Rendezvous Location

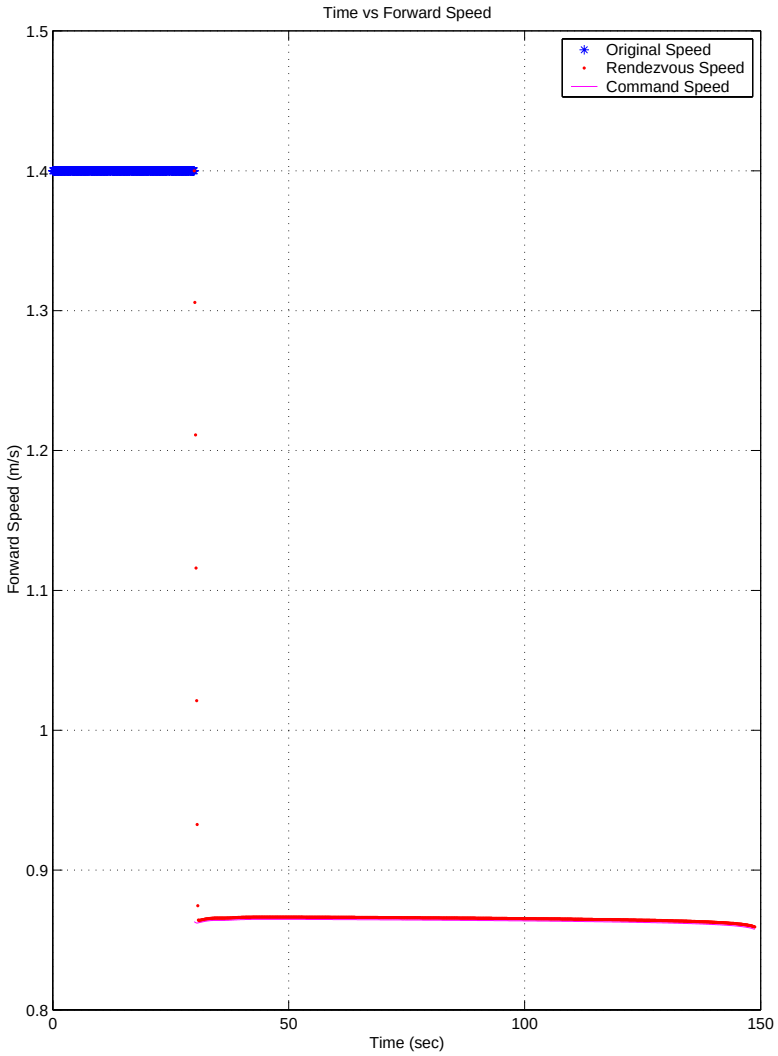

Figure 27: Speed vs Time for [50, 50] Rendezvous Location

In both cases, the speed controller reacts to the change in location and commands the vehicle to the proper speed in order to arrive at the designated time.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This work has demonstrated that the ARIES AUV can be outfitted with a new behavior of arriving at a defined place in space and time and that the pre-existing cross-track error guidance algorithms can be extended to close along track as well as cross track errors resulting in a trajectory controller.

While only a simulation, this work can be extremely useful to the mission planner. The ability to determine ARIES' response over various distances will allow the MIW Commander to effectively plan the use of his assets when setting up search patterns for the REMUS vehicles and waiting patterns for ARIES.

B. RECOMMENDATIONS

The difficulty encountered in determining the hydrodynamic coefficients of the longitudinal equation of motion can be directly attributed to the size and number of open ocean missions that were conducted. In order to fully realize these values, more runs would need to be conducted. It is further recommended that each run start at ARIES' minimum speed and then incrementally increase her speed in small intervals until reaching maximum speed. Stepping ARIES speed back down to minimum in the same interval would enhance the data for her deceleration. Overall this data would be a vast improvement over the data that was collected during the 17 April 2002 experiment and would increase confidence in the model for all speed scenarios.

The simulation presented involved ARIES receiving a simulated command from REMUS to rendezvous at a particular place and time. It was shown that ARIES is constrained by her velocity and acceleration and therefore will not always be able to conform to REMUS' command. There is no method in place for ARIES to make this known to REMUS and propose an alternate location and/or time. Additionally, the simulation is hard coded to ask the user to input a simulated modem command thirty seconds into ARIES' mission. In practice, ARIES should be able to receive a command to rendezvous at any time during her mission. It is obvious that further study involving the communication between vehicles is thus warranted.

Finally, while this simulation shows that ARIES can achieve a rendezvous in space and time, it is only a simulation. Transforming the code contained in finalrendezvous.m (Appendix D) into ARIES' C programming language would be a worthy endeavor, as the Center for AUV Research stands ready to receive their fourth generation vehicle in the coming months. With two vehicles in operation, the work presented here can someday be validated on the open ocean.

APPENDIX A. MATLAB FILE WAYPOINT.M

This appendix contains a MATLAB file written by Marco (2001) that simulates ARIES using heading control and cross track error control for a planned path.

```
%File CTE_Loiter
whitebg('k');
% State = [v r psi]
clear

TRUE  = 1;
FALSE = 0;

DegRad = pi/180;
RadDeg = 180/pi;
%State Model PArameters
W   = 600.0;
U = 1.4*3.28;
g = 32.174;
Boy = 500.0;
xg  = 0.125/12.0;
m = W/g;

rho = 1.9903;
L = 10;

Iz = (1/12)*m*(1.33^2 + 10^2); % Approx. Using I = 1/12*m*(a^2 + b^2)
Iz = Iz*5.0;
Yv_dot = -0.03430*(rho/2)*L^3;
Yr_dot = -0.00178*(rho/2)*L^4;
Yv = -0.10700*(rho/2)*L^2;
Yr = 0.01187*(rho/2)*L^3;
Ydrs = (0.01241*(rho/2)*L^2)/2.0; % Since Bow & Stern Lower Rudders Removed
Ydrb = (0.01241*(rho/2)*L^2)/2.0;
Nv_dot = -0.00178*(rho/2)*L^4;
%Nr_dot = -0.00047*(rho/2)*L^5;
Nr_dot = -Iz;
Nv = -0.00769*(rho/2)*L^3;
Nr = -0.00390*(rho/2)*L^4;
%Ndrs = -2.6496/2.0; % Since Bow & Stern Lower Rudders Removed
%Ndrb = 1.989/2.0;

% Below Modified on 7/12/00 The 3.5 and 3.4167 is the Moment Arm Length in Feet
Ndrs = -0.01241*(rho/2)*(L^2)*(3.5)/2.0; % Since Stern Lower Rudder Removed
Ndrb = 0.01241*(rho/2)*(L^2)*(3.4167)/2.0; % Since Bow Lower Rudder Removed

% Combining Stern & Bow Rudder Effectivness
Ndr = Ndrs - Ndrb;
Ydr = Ydrs - Ydrb; % Cancel Out
m1 = m - Yv_dot;
m2 = m*xg - Yr_dot;
m3 = m*xg - Nv_dot;
m4 = Iz - Nr_dot;
Y1 = Yv;
Y2 = Yr;
```

```
Y3 = U^2*Ydr;
N1 = Nv;
N2 = Nr;
N3 = U^2*Ndr;

A = [Y1*U Y2*U;N1*U N2*U];
B = [Y3 N3]';
M = [m1 m2;m3 m4];
A1 = inv(M)*A;
B1 = inv(M)*B;
AO = [A1(1,1) A1(1,2) 0;
    A1(2,1) A1(2,2) 0;
     0 1 0];
BO = [B1;0];

dt = 0.125;
t = [0:dt:1000]';

size(t)
% set initial conditions
start=10;
v(1)   = 0.0;
r(1)   = 0.0;
rRM(1) = r(1);
% This is the Initial Heading of the Vehicle
psi(1) = 50.0*DegRad;

% This is the Initial Position of the Vehicle
X(1) = -80.0; % Meters
Y(1) = 10.0;

% Convert to Feet
% this data from track.out file
No_tracks=5;
Track=[ 10.0 10.0   2.75 2.75  0  1.25  1.00 0 25.00 8.00 40.00
       10.0 210.0  2.75 2.75  0  1.25  1.00 0 25.00 8.00 200.00
       40.0 210.0  2.75 2.75  0  1.25  1.00 0 25.00 2.00 30.00
       40.0 10.0   2.75 2.75  0  1.25  1.00 0 25.00 2.00 200.00
      -20.0 -60.0  2.75 2.75  0  1.25  1.00 0 25.00 2.00 100.00];
track=Track(:,1:2);
SurfaceTime = Track(:,9);
SurfPhase   = Track(:,8);

% readin wayopoints from track data assumes track is loaded
for j=1:No_tracks,
  X_Way_c(j)     = track(j,1);
  Y_Way_c(j)     = track(j,2);
end;

%Set start position
PrevX_Way_c(1) = -80.0;
PrevY_Way_c(1) =  10.0;
r_com = 0.0;
W_R = 10.0;
a = -.3;
b = (9/24)*a;
```

```
x(:,1) = [v(1);r(1);psi(1)];

% Below are in British Units for CTE Sliding Mode
%Lam1 = 0.75;
%Lam2 = 0.5;
Lam1 = 2.0;
Lam2 = 1.0;
Eta_FlightHeading = 1.0;
Phi_FlightHeading = 0.5;

% Below for tanh
Eta_CTE = 0.1;
Eta_CTE_Min = 1.0;
Phi_CTE = 0.5;
   Uc = [];
   Vc = [];
PLOT_PART = 0;
SegLen(1) = sqrt((X_Way_c(1)-PrevX_Way_c(1))^2+(Y_Way_c(1)-PrevY_Way_c(1))^2);
psi_track(1) = atan2(Y_Way_c(1)-PrevY_Way_c(1),X_Way_c(1)-PrevX_Way_c(1));

for j=2:No_tracks,
   SegLen(j) = sqrt((X_Way_c(j)-X_Way_c(j-1))^2+(Y_Way_c(j)-Y_Way_c(j-1))^2);
   psi_track(j) = atan2(Y_Way_c(j)-Y_Way_c(j-1),X_Way_c(j)-X_Way_c(j-1));
end;

  j=1;
  Sigma = [];
  Depth_com = [];
  dr=[];
  drl = [];
  drl(1) = 0.0;
  Depth_com(1) = 5.0;
  WayPointVertDist_com = [5.0 5.0 5.0 5.0 5.0];

SURFACE_TIMER_ACTIVE = FALSE;
for i=1:length(t)-1,
   Depth_com(i) = WayPointVertDist_com(j);
   X_Way_Error(i) = X_Way_c(j) - X(i);
   Y_Way_Error(i) = Y_Way_c(j) - Y(i);
   % DeWrap psi to within +/- 2.0*pi;
   psi_cont(i) = psi(i);
   while(abs(psi_cont(i)) > 2.0*pi)
      psi_cont(i) = psi_cont(i) - sign(psi_cont(i))*2.0*pi;
   end;
   psi_errorCTE(i) = psi_cont(i) - psi_track(j);
   % DeWrap psi_error to within +/- pi;
   while(abs(psi_errorCTE(i)) > pi)
      psi_errorCTE(i) = psi_errorCTE(i) - sign(psi_errorCTE(i))*2.0*pi;
   end;

% **  Always Calculate this
      Beta = v(i)/U;
%      Beta = 0.0;
      cpsi_e = cos(psi_errorCTE(i)+Beta);
      spsi_e = sin(psi_errorCTE(i)+Beta);
```

```
    s(i) = [X_Way_Error(i),Y_Way_Error(i)]*...
          [(X_Way_c(j)-PrevX_Way_c(j)),(Y_Way_c(j)-PrevY_Way_c(j))]';
    % s is distance to go projected to track line(goes from 0-100%L)

    s(i) = s(i)/SegLen(j);
    Ratio=(1.0-s(i)/SegLen(j))*100.0;
% **
    % radial distance to go to next WP
    ss(i) = sqrt(X_Way_Error(i)^2 + Y_Way_Error(i)^2);
    % dp is angle between line of sight and current track line
    dp(i) = ...
          atan2( (Y_Way_c(j)-PrevY_Way_c(j)),(X_Way_c(j)-PrevX_Way_c(j)) )...
        - atan2( Y_Way_Error(i),X_Way_Error(i) );
    if(dp(i) > pi),
      dp(i) = dp(i) - 2.0*pi;
    end;
    cte(i) = s(i)*sin(dp(i));
    if( abs(psi_errorCTE(i)) >= 40.0*pi/180.0 | s(i) < 0.0 ),
      % Use LOS Control
      LOS(i) = 1;
      psi_comLOS = atan2(Y_Way_Error(i),X_Way_Error(i));
      psi_errorLOS(i) = psi_comLOS - psi_cont(i);
     if(abs(psi_errorLOS(i)) > pi),
       psi_errorLOS(i) = ...
       psi_errorLOS(i) - 2.0*pi*psi_errorLOS(i)/abs(psi_errorLOS(i));
     end;
     Sigma_FlightHeading = 0.9499*(r_com - r(i)) + 0.1701*psi_errorLOS(i);
     dr(i) = -1.5435*( 2.5394*r(i) ...
           + Eta_FlightHeading*tanh(Sigma_FlightHeading/Phi_FlightHeading));
  else
    % Use CTE Controller
    LOS(i) = 0;
    if(cpsi_e ~= 0.0), % Trap Div. by Zero !
%     SMC Soln
    Sigma(i) = U*rRM(i)*cpsi_e + Lam1*U*spsi_e + 3.28*Lam2*cte(i);
    dr(i) = (1.0/(U*b*cpsi_e))*(-U*a*rRM(i)*cpsi_e + U*rRM(i)^2*spsi_e ...
        - Lam1*U*rRM(i)*cpsi_e - Lam2*U*spsi_e - Eta_CTE*(Sigma(i)/Phi_CTE));
    else
          dr(i) = dr(i-1);
    end;

  end; % End of CTE Controller

  % use LOS if near to loiter point

                    %         if (loiter==1)& s(i)<10;  dr(i)=drlos(i);end;

  % Surface Phase Logic (Independent of LOS or CTE)

    if(SurfPhase(j) == TRUE)
      if(SURFACE_TIMER_ACTIVE == FALSE)
        if(Ratio > 40.0)
          % Start a Timer
          SURFACE_TIMER_ACTIVE = TRUE;
          Depth_com(i) = 0.0;
          SurfaceWait = SurfaceTime(j) + t(i);
```

```
            SurfaceWait
          end;
        end;
      end;

      if(SURFACE_TIMER_ACTIVE == TRUE)
        if(t(i) >= SurfaceWait)
          SURFACE_TIMER_ACTIVE = FALSE;
          Depth_com(i) = WayPointVertDist_com(j);
          SurfPhase(j) = 0;
        else
          Depth_com(i) = 0.0;
        end;
      end;

    if(abs(dr(i)) > 0.4)
      dr(i) = 0.4*sign(dr(i));
    end;

  % Model drl is the actual lagged rudder, dr is the rudder command.
  %   taudr = 0.255;

  %  drl(i+1) = drl(i) + dt*(dr(i)-drl(i))/taudr;
  %          if(abs(drl(i)) > 0.4)
  %                                  drl(i) = 0.4*sign(drl(i));
  %          end;


    %Jay Johnson Model;
    Yv  = -68.16;
    Yr  = 406.3;
    Ydr = 70.0;
    Nv  = -10.89;
    Nr  = -88.34;
    Ndr = -35.47;
    MY = 456.76;
    IN = 215;

    M = diag([MY,IN,1]);
    AA = [Yv,Yr,0;Nv,Nr,0;0,1,0];
    BB = [Ydr;Ndr;0];
    A  = inv(M)*AA;
    B  = inv(M)*BB;

  % x_dot(:,i+1) = [ A(1,1)*v(i) + A(1,2)*r(i) + B(1)*drl(i);
  %                A(2,1)*v(i) + A(2,2)*r(i) + B(2)*drl(i);
  %                r(i)];
  x_dot(:,i+1) = [ A(1,1)*v(i) + A(1,2)*r(i) + B(1)*dr(i);
                 A(2,1)*v(i) + A(2,2)*r(i) + B(2)*dr(i);
                 r(i)];

    x(:,i+1) = x(:,i)+dt*x_dot(:,i);
    v(i+1)   = x(1,i+1)/12;
    r(i+1)   = x(2,i+1);
    psi(i+1) = x(3,i+1);
    rRM(i+1) = r(i+1);
```

```
%  Added
%   rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*dr(i));
%   psi(i+1) = psi(i) + dt*rRM(i);

%  Throw in some Waves
  %Uc(i) = -0.5*sin(2*pi*t(i)/5);
  %Vc(i) =  0.5*sin(2*pi*t(i)/5);

  %Model using system ID results from Bay tests

  %  rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*drl(i));
  %  psi(i+1) = psi(i) + dt*rRM(i);
  % side slip added proprtional to turn rate from AZORES data V in ft/sec
  % v(i+1) = 1.0*rRM(i+1)*3.28;

  Uc = 0.0;
  Vc = 0.0;

  %Kinematics
  X(i+1) = X(i) + (Uc + (U/3.28)*cos(psi(i)) - v(i)/3.28*sin(psi(i)) )*dt;
  Y(i+1) = Y(i) + (Vc + (U/3.28)*sin(psi(i)) + v(i)/3.28*cos(psi(i)) )*dt;

  %  Check to See if we are Within the Watch_Radius

  if(sqrt(X_Way_Error(i)^2.0 + Y_Way_Error(i)^2.0) <= W_R | s(i) < 0.0),
  disp(sprintf('WayPoint %d Reached',j));
    if(j==No_tracks),
      PLOT_PART = 1;
      break;
    end;
    PrevX_Way_c(j+1) = X_Way_c(j);
    PrevY_Way_c(j+1) = Y_Way_c(j);
    j=j+1;
  end;
end;

dr(i+1) = dr(i);
cte(i+1) = cte(i);
s(i+1) = s(i);
ss(i+1) = ss(i);

if(PLOT_PART),

  figure(1);
  plot(t([1:i+1]),psi*180/pi);
  hold;
  plot(t([1:i+1]),dr*180/pi,'r');grid;
  hold;zoom on;
  figure(2);
  plot(t([1:i+1]),cte);
  hold;
  plot(t([1:i+1]),s,'r');
  plot(t([1:i+1]),ss,'g');grid;
  hold;zoom on;
```

```
else
   figure(1);
   plot(t,psi*180/pi);
   hold;
   plot(t,dr*180/pi,'r');
   hold;grid;
   figure(2);
   plot(t,cte);
   hold;
   plot(t,s,'r');
   plot(t,ss,'g');grid;
   hold;zoom on;
end;

   figure(3);
   plot(Y,X);grid;
   hold
   plot([Y_Way_c(1) PrevY_Way_c(1)],[X_Way_c(1) PrevX_Way_c(1)],'r');
   for ii=2:No_tracks,
      plot([Y_Way_c(ii) Y_Way_c(ii-1)],[X_Way_c(ii) X_Way_c(ii-1)],'r');
   end;
   hold;zoom on;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MATLAB FILE RENDEZVOUS.M

This appendix contains a MATLAB file based on the file in appendix A that simulates ARIES receiving a modem command thirty seconds into her planned mission for her to rendezvous at a specific location.

```
% This file has a pre-defined track.out file in it.  30 seconds into the
% simulation, the user is asked if ARIES should be diverted to a
% rendezvous point.  If the answer is 'yes', the user is prompted to enter
% a new track.out file that consists of one row (This simulates an acoustic
% modem command).  The simulation will show ARIES aborts the planned track
% and goes to the directed rendezvous point.  If the answer is 'no' to the
% query concerning the rendezvous point, the mission continues as planned.

% All equation numbers (i.e, Eq (#)) refer to equations presented in Marco
% Healey's "Command, Control and Navigation Experimental Results
% With the NPS ARIES AUV"

% 06 Feb 2002

whitebg('k');
% State = [v r psi]
clear
TRUE  = 1;
FALSE = 0;

% Converts Degrees to Radians & Radians to Degrees

DegRad = pi/180;
RadDeg = 180/pi;

% State Model Parameters

W   = 600.0;                % Weight in LB
U = 1.4*3.28;               % Forward Speed ?
g = 32.174;                 % Gravity in ft/sec^2
Boy = 500.0;                % Bouyancy ?
xg  = 0.125/12.0;           % ??
m = W/g;                    % Mass
rho = 1.9903;               % Density of Seawater in slugs/ft^3
L = 10;                     % Length in ft of ARIES
Iz = (1/12)*m*(1.33^2 + 10^2);  % Approx. Using I = 1/12*m*(a^2 + b^2)
                            % where a is width & b is length
Iz = Iz*5.0;

% Coefficients

Yv_dot = -0.03430*(rho/2)*L^3;  % Added Mass in Sway Coefficient.
Yr_dot = -0.00178*(rho/2)*L^4;  % Added Mass in Yaw Coefficient.
Yv = -0.10700*(rho/2)*L^2;      % Coeff. of Sway Force induced by Side Slip
Yr = 0.01187*(rho/2)*L^3;       % Coeff. of Sway Force induced by Yaw
Ydrs = (0.01241*(rho/2)*L^2)/2.0; % Since Bow & Stern Lower Rudders Removed
Ydrb = (0.01241*(rho/2)*L^2)/2.0; % So don't use these equations

Nv_dot = -0.00178*(rho/2)*L^4; % Added Mass Moment of Inertia in Sway Coeff
```

```
%Nr_dot = -0.00047*(rho/2)*L^5;
Nr_dot = -Iz;                % Added Mass Moment of Inertia in Yaw Coeff
Nv = -0.00769*(rho/2)*L^3;     % Coeff. of Sway Moment from Side Slip
Nr = -0.00390*(rho/2)*L^4;     % Coeff. of Sway Moment from Yaw
%Ndrs = -2.6496/2.0;          % Since Bow & Stern Lower Rudders Removed
%Ndrb = 1.989/2.0;

% Below Modified on 7/12/00 The 3.5 and 3.4167 is the Moment Arm Length
% in Feet - Since Bow & Stern Lower Rudders Removed

Ndrs = -0.01241*(rho/2)*(L^2)*(3.5)/2.0;
Ndrb = 0.01241*(rho/2)*(L^2)*(3.4167)/2.0;

% Combining Stern & Bow Rudder Effectivness

Ndr = Ndrs - Ndrb;
Ydr = Ydrs - Ydrb;                        % Cancel Out

% Matrices

m1 = m - Yv_dot;
m2 = m*xg - Yr_dot;
m3 = m*xg - Nv_dot;
m4 = Iz - Nr_dot;
Y1 = Yv;
Y2 = Yr;
Y3 = U^2*Ydr;
N1 = Nv;
N2 = Nr;
N3 = U^2*Ndr;
A = [Y1*U Y2*U;N1*U N2*U];
B = [Y3 N3]';
M = [m1 m2;m3 m4];
A1 = inv(M)*A;
B1 = inv(M)*B;
AO = [A1(1,1) A1(1,2) 0;
    A1(2,1) A1(2,2) 0;
     0 1 0];
BO = [B1;0];
dt = 0.125;
t = [0:dt:1000]';
size(t);

% Set initial conditions

start=10;
v(1)   = 0.0;                    % Initial Side Slip Velocity
r(1)   = 0.0;                    % Initial Yaw
rRM(1) = r(1);
psi(1) = 50.0*DegRad;            % Initial Heading of ARIES
X(1) = -80.0;                    % Initial Position in Feet
Y(1) = 10.0;

% Convert to Feet ?

% This data from track.out file
```

```
No_tracks=5;                    % Sets # of Tracks = # of Rows

Track=[ 10.0 10.0   2.75 2.75  0  1.25  1.00 0 25.00 8.00 40.00
        10.0 210.0  2.75 2.75  0  1.25  1.00 0 25.00 8.00 200.00
        40.0 210.0  2.75 2.75  0  1.25  1.00 0 25.00 2.00 30.00
        40.0 10.0   2.75 2.75  0  1.25  1.00 0 25.00 2.00 200.00
       -20.0 -60.0  2.75 2.75  0  1.25  1.00 0 25.00 2.00 100.00];

track=Track(:,1:2);             % Defines track as Track(X,Y)
SurfaceTime = Track(:,9);       % Col 9 of Track is Surface Time for Pop-up
SurfPhase   = Track(:,8);       % Col 8 of Track designates if Pop-up

% Read in wayopoints from track data assumes track is loaded

for j=1:No_tracks,
   X_Way_c(j)     = track(j,1);
   Y_Way_c(j)     = track(j,2);
end;

% Set start position

PrevX_Way_c(1) = -80.0;
PrevY_Way_c(1) =  10.0;
r_com = 0.0;
W_R = 10.0;                     % Sets initial Watch Radius
a = -.3;
b = (9/24)*a;
x(:,1) = [v(1);r(1);psi(1)];

% Below are in British Units for CTE Sliding Mode
%Lam1 = 0.75;
%Lam2 = 0.5;
Lam1 = 2.0;
Lam2 = 1.0;
Eta_FlightHeading = 1.0;
Phi_FlightHeading = 0.5;

% Below for tanh

Eta_CTE = 0.1;
Eta_CTE_Min = 1.0;
Phi_CTE = 0.5;
Uc = [];
Vc = [];
PLOT_PART = 0; disp(sprintf('PLOT_PART = 0'));

% Total Track Length between initial waypoint and waypoint (1)

SegLen(1) = sqrt((X_Way_c(1)-PrevX_Way_c(1))^2+(Y_Way_c(1)...
   -PrevY_Way_c(1))^2);

% Track Angle of first track

psi_track(1) = atan2(Y_Way_c(1)-PrevY_Way_c(1),X_Way_c(1)-PrevX_Way_c(1));
```

```
% Computes track lengths and track angles for each track

for j=2:No_tracks,
  SegLen(j) = sqrt((X_Way_c(j)-X_Way_c(j-1))^2+(Y_Way_c(j)-...
     Y_Way_c(j-1))^2);
  psi_track(j) = atan2(Y_Way_c(j)-Y_Way_c(j-1),X_Way_c(j)-X_Way_c(j-1));
end;
j=1;
Sigma = [];
Depth_com = [];
dr=[];
drl = [];
drl(1) = 0.0;
Depth_com(1) = 5.0;
WayPointVertDist_com = [5.0 5.0 5.0 5.0 5.0];
SURFACE_TIMER_ACTIVE = FALSE;

% Starts a loop that computes values for each data point corresponding to
% a time value (in this case, every 0.125 seconds from 1 to 1000 seconds)

for i=1:length(t)-1,
  Depth_com(i) = WayPointVertDist_com(j);

% Difference between current vehicle position & the next waypoint Eq(13)

  X_Way_Error(i) = X_Way_c(j) - X(i);
  Y_Way_Error(i) = Y_Way_c(j) - Y(i);

% DeWrap psi to within +/- 2.0*pi; Makes Heading Angle to lie between
% 0-360 degrees

  psi_cont(i) = psi(i);
  while(abs(psi_cont(i)) > 2.0*pi)
    psi_cont(i) = psi_cont(i) - sign(psi_cont(i))*2.0*pi;
  end;

% Cross Track Heading Error Eq(12)

  psi_errorCTE(i) = psi_cont(i) - psi_track(j);

% DeWrap psi_error to within +/- pi;  Normalized to Lie between +/- 180
% degrees

  while(abs(psi_errorCTE(i)) > pi)
    psi_errorCTE(i) = psi_errorCTE(i) - sign(psi_errorCTE(i))*2.0*pi;
  end;

% **  Always Calculate this (What is This?)
  Beta = v(i)/U;
%  Beta = 0.0;
  cpsi_e = cos(psi_errorCTE(i)+Beta);
  spsi_e = sin(psi_errorCTE(i)+Beta);

% Distance to the ith way point projected to the track line S(t)i -
% Eq (14)
```

```
  s(i) = [X_Way_Error(i),Y_Way_Error(i)]*[(X_Way_c(j)-...
        PrevX_Way_c(j)),(Y_Way_c(j)-PrevY_Way_c(j))]';

% s is distance to go projected to track line(goes from 0-100%L) - Eq (14)

  s(i) = s(i)/SegLen(j);
  Ratio=(1.0-s(i)/SegLen(j))*100.0;     % Ranges from 0-100% of SegLen

% Radial distance to go to next WP

  ss(i) = sqrt(X_Way_Error(i)^2 + Y_Way_Error(i)^2);

% dp is angle between line of sight and current track line - Eq (16)

  dp(i) = atan2( (Y_Way_c(j)-PrevY_Way_c(j)),(X_Way_c(j)-...
        PrevX_Way_c(j)) )- atan2( Y_Way_Error(i),X_Way_Error(i) );
  if(dp(i) > pi),
     dp(i) = dp(i) - 2.0*pi;
  end;

% Cross Track Error Definition - Eq (15)

  cte(i) = s(i)*sin(dp(i));

% If the magnitude of the CTE Heading exceeds 40 degrees, a LOS Controller
% is used.

  if( abs(psi_errorCTE(i)) >= 40.0*pi/180.0 | s(i) < 0.0 ),
    LOS(i) = 1;
    psi_comLOS = atan2(Y_Way_Error(i),X_Way_Error(i));    % Eq (22)
    psi_errorLOS(i) = psi_comLOS - psi_cont(i);           % Eq (23)
                                          % LOS Error
    if(abs(psi_errorLOS(i)) > pi),
      psi_errorLOS(i) = psi_errorLOS(i) - 2.0*pi*psi_errorLOS(i)...
         /abs(psi_errorLOS(i));
    end;

% Eq (8)

    Sigma_FlightHeading = 0.9499*(r_com - r(i)) + 0.1701*psi_errorLOS(i);

% Eq (9)

    dr(i) = -1.5435*( 2.5394*r(i)+ Eta_FlightHeading*tanh...
       (Sigma_FlightHeading/Phi_FlightHeading));

    else

% Use CTE Controller if CTE Heading is less than 40 degrees

    LOS(i) = 0;
    if(cpsi_e ~= 0.0),              % Trap Div. by Zero !

% SMC Soln

% Sliding Surface - Eq (20)
```

```
        Sigma(i) = U*rRM(i)*cpsi_e + Lam1*U*spsi_e + 3.28*Lam2*cte(i);

% Rudder Input - Eq (21)

        dr(i) = (1.0/(U*b*cpsi_e))*(-U*a*rRM(i)*cpsi_e + U*rRM(i)^2*...
           spsi_e - Lam1*U*rRM(i)*cpsi_e - Lam2*U*spsi_e - Eta_CTE* ...
           (Sigma(i)/Phi_CTE));
     else
            dr(i) = dr(i-1);
     end;
  end;                          % End of CTE Controller

  % Use LOS if near to loiter point
  % if (loiter==1)& s(i)<10;  dr(i)=drlos(i);end;

  % Surface Phase Logic (Independent of LOS or CTE)

  if(SurfPhase(j) == TRUE)
     if(SURFACE_TIMER_ACTIVE == FALSE)
        if(Ratio > 40.0)
           % Start a Timer
           SURFACE_TIMER_ACTIVE = TRUE;
           Depth_com(i) = 0.0;
           SurfaceWait = SurfaceTime(j) + t(i);
           SurfaceWait
        end;
     end;
  end;
  if(SURFACE_TIMER_ACTIVE == TRUE)
    if(t(i) >= SurfaceWait)
       SURFACE_TIMER_ACTIVE = FALSE;
       Depth_com(i) = WayPointVertDist_com(j);
       SurfPhase(j) = 0;
       else
           Depth_com(i) = 0.0;
    end;
  end;
  if(abs(dr(i)) > 0.4)
     dr(i) = 0.4*sign(dr(i));
  end;

  % Model drl is the actual lagged rudder, dr is the rudder command.
  % taudr = 0.255;

  %  drl(i+1) = drl(i) + dt*(dr(i)-drl(i))/taudr;
  %         if(abs(drl(i)) > 0.4)
  %                     drl(i) = 0.4*sign(drl(i));
  %         end;

  % Jay Johnson Model

   Yv  = -68.16;
   Yr  = 406.3;
   Ydr = 70.0;
   Nv  = -10.89;
```

```
  Nr  = -88.34;
  Ndr = -35.47;

  MY = 456.76;
  IN = 215;

  M = diag([MY,IN,1]);
  AA = [Yv,Yr,0;Nv,Nr,0;0,1,0];
  BB = [Ydr;Ndr;0];
  A  = inv(M)*AA;
  B  = inv(M)*BB;

 % x_dot(:,i+1) = [ A(1,1)*v(i) + A(1,2)*r(i) + B(1)*drl(i);
 %                 A(2,1)*v(i) + A(2,2)*r(i) + B(2)*drl(i);
 %                 r(i)];
  x_dot(:,i+1) = [ A(1,1)*v(i) + A(1,2)*r(i) + B(1)*dr(i);
                 A(2,1)*v(i) + A(2,2)*r(i) + B(2)*dr(i);
                 r(i)];

  x(:,i+1) = x(:,i)+dt*x_dot(:,i);
  v(i+1)   = x(1,i+1)/12;
  r(i+1)   = x(2,i+1);
  psi(i+1) = x(3,i+1);
  rRM(i+1) = r(i+1);

 % Added
 % rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*dr(i));
 % psi(i+1) = psi(i) + dt*rRM(i);

 % Throw in some Waves
 % Uc(i) = -0.5*sin(2*pi*t(i)/5);
 % Vc(i) =  0.5*sin(2*pi*t(i)/5);

 % Model using system ID results from Bay tests

 % rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*drl(i));
 % psi(i+1) = psi(i) + dt*rRM(i);
 % side slip added proprtional to turn rate from AZORES data V in ft/sec
 % v(i+1) = 1.0*rRM(i+1)*3.28;

  Uc = 0.0;
  Vc = 0.0;

  % Kinematics

  X(i+1) = X(i) + (Uc + (U/3.28)*cos(psi(i)) - v(i)/3.28*sin(psi(i)) )*dt;
  Y(i+1) = Y(i) + (Vc + (U/3.28)*sin(psi(i)) + v(i)/3.28*cos(psi(i)) )*dt;

%*************************************************************************

% This should abort @ 30 seconds if input is empty or 'Y'
% This modification done on 05 Feb 2002 - original file is waypoint1.m

  if i == 240,
    K = input('Is Rendezvous Required? (Enter 1 for Yes, 0 for No)-->>');
    if isempty(K)==1; K = 1; break; end;
```

```
    if K == 1; break;
    else i = i; end;
  end

%*************************************************************************

  % Check to See if we are Within the Watch_Radius

  if(sqrt(X_Way_Error(i)^2.0 + Y_Way_Error(i)^2.0) <= W_R | s(i) < 0.0),
  disp(sprintf('WayPoint %d Reached',j));
    if(j==No_tracks),
      PLOT_PART = 1;
      disp(sprintf('PLOT_PART = 1'));
      break;
    end;
    PrevX_Way_c(j+1) = X_Way_c(j);
    PrevY_Way_c(j+1) = Y_Way_c(j);
    j=j+1;
 end;
end;

%*************************************************************************

% Requests Rendezvous Point Information
% This modification done on 05 Feb 2002 - original file waypoint2.m

if j == No_tracks,
  disp(sprintf('Mission Complete'));
else
new_r_com = 0.0;
new_v(1) = v(i+1);
new_r(1) = r(i+1);
new_rRM(1) = new_r(1);
new_psi(1) = psi(i+1);
New_X(1) = X(i+1);
New_Y(1) = Y(i+1);
New_No_Tracks = 1;
New_Track = input('Enter 11 Column Track, i.e., [1 1 ...]-->>');
new_track = New_Track(:,1:2);
new_SurfaceTime = New_Track(:,9);
new_Surfphase = New_Track(:,8);
for jj = 1:New_No_Tracks,
 New_X_Way_c(jj) = new_track(jj,1);
 New_Y_Way_c(jj) = new_track(jj,2);
end;
New_PrevX_Way_c(1) = X(i+1);        % Sets Abort Posit as start of new
New_PrevY_Way_c(1) = Y(i+1);        % track.
% Below for tanh
new_Eta_CTE = 0.1;
new_Eta_CTE_Min = 1.0;
new_Phi_CTE = 0.5;
PLOT_PART = 0; disp(sprintf('PLOT_PART = 0'));
new_x(:,1) = [new_v(1); new_r(1); new_psi(1)];

% Total Track Length between abort point and rendezvous point
```

```
New_SegLen(1) = sqrt((New_X_Way_c(1)-New_PrevX_Way_c(1))^2+...
   (New_Y_Way_c(1) - New_PrevY_Way_c(1))^2);

% Track Angle of track between abort point and rendezvous point

new_psi_track(1) = atan2(New_Y_Way_c(1)-New_PrevY_Way_c(1),...
   New_X_Way_c(1)-New_PrevX_Way_c(1));

% Starts loop that computes values for each data point corresponding to a
% time value along new track

jj=1;
new_Sigma = [];
new_Depth_com = [];
new_dr=[];
new_drl = [];
new_drl(1) = 0.0;
new_Depth_com(1) = 5.0;
new_WayPointVertDist_com = [5.0 5.0 5.0 5.0 5.0];
new_SURFACE_TIMER_ACTIVE = FALSE;
tt = [t(i+1):dt:500]';
size(tt);
for ii = 1:length(tt)-1,
   new_Depth_com(ii) = new_WayPointVertDist_com(jj);
   New_X_Way_Error(ii) = New_X_Way_c(jj) - New_X(ii);
   New_Y_Way_Error(ii) = New_Y_Way_c(jj) - New_Y(ii);
   new_psi_cont(ii) = new_psi(ii);
   while(abs(new_psi_cont(ii)) > 2.0*pi)
     new_psi_cont(ii) = new_psi_cont(ii) - sign(new_psi_cont(ii))*...
         2.0*pi;
   end;

% Cross Track Heading Error Eq(12)

   new_psi_errorCTE(ii) = new_psi_cont(ii) - new_psi_track(jj);

% DeWrap psi_error to within +/- pi;  Normalized to Lie between +/- 180
% degrees

   while(abs(new_psi_errorCTE(ii)) > pi)
      new_psi_errorCTE(ii) = new_psi_errorCTE(ii) - sign(...
         new_psi_errorCTE(ii))*2.0*pi;
   end;

% **  Always Calculate this (What is This?)
   new_Beta = new_v(ii)/U;
%   Beta = 0.0;
   new_cpsi_e = cos(new_psi_errorCTE(ii)+new_Beta);
   new_spsi_e = sin(new_psi_errorCTE(ii)+new_Beta);

% Distance to the ith way point projected to the track line S(t)i -
% Eq (14)

   new_s(ii) = [New_X_Way_Error(ii),New_Y_Way_Error(ii)]*[(...
         New_X_Way_c(jj)-New_PrevX_Way_c(jj)),(New_Y_Way_c(jj)...
         -New_PrevY_Way_c(jj))]';
```

```
% s is distance to go projected to track line(goes from 0-100%L) - Eq (14)

   new_s(ii) = new_s(ii)/New_SegLen(jj);
   % Ranges from 0-100% of SegLen
   Ratio=(1.0-new_s(ii)/New_SegLen(jj))*100.0;

% Radial distance to go to next WP

   new_ss(ii) = sqrt(New_X_Way_Error(ii)^2 + New_Y_Way_Error(ii)^2);

% dp is angle between line of sight and current track line - Eq (16)

   new_dp(ii) = atan2( (New_Y_Way_c(jj)-New_PrevY_Way_c(jj)),(...
      New_X_Way_c(jj)-New_PrevX_Way_c(jj)) )- atan2...
      (New_Y_Way_Error(ii),New_X_Way_Error(ii) );
   if(new_dp(ii) > pi),
      new_dp(ii) = new_dp(ii) - 2.0*pi;
   end;

% Cross Track Error Definition - Eq (15)

   new_cte(ii) = new_s(ii)*sin(new_dp(ii));

% If the magnitude of the CTE Heading exceeds 40 degrees, a LOS Controller
% is used.

   if( abs(new_psi_errorCTE(ii)) >= 40.0*pi/180.0 | new_s(ii) < 0.0 ),
     new_LOS(ii) = 1;
     new_psi_comLOS = atan2(New_Y_Way_Error(ii),New_X_Way_Error(ii));
     new_psi_errorLOS(ii) = new_psi_comLOS - new_psi_cont(ii);
     if(abs(new_psi_errorLOS(ii)) > pi),
       new_psi_errorLOS(ii) = new_psi_errorLOS(ii) - 2.0*pi*...
       new_psi_errorLOS(ii)/abs(new_psi_errorLOS(ii));
     end;

% Eq (8)

     new_Sigma_FlightHeading = 0.9499*(new_r_com - new_r(ii)) +...
        0.1701*new_psi_errorLOS(ii);

% Eq (9)

     new_dr(ii) = -1.5435*( 2.5394*new_r(ii)+ Eta_FlightHeading*tanh...
        (new_Sigma_FlightHeading/Phi_FlightHeading));

     else

% Use CTE Controller if CTE Heading is less than 40 degrees

     new_LOS(ii) = 0;
     if(new_cpsi_e ~= 0.0),              % Trap Div. by Zero !

% SMC Soln

% Sliding Surface - Eq (20)
```

```
      new_Sigma(ii) = U*new_rRM(ii)*new_cpsi_e + Lam1*U*new_spsi_e...
         + 3.28*Lam2*new_cte(ii);

% Rudder Input - Eq (21)

      new_dr(ii) = (1.0/(U*b*new_cpsi_e))*(-U*a*new_rRM(ii)*...
         new_cpsi_e + U*new_rRM(ii)^2*new_spsi_e - Lam1*U*...
         new_rRM(ii)*new_cpsi_e - Lam2*U*new_spsi_e - new_Eta_CTE* ...
         (new_Sigma(ii)/new_Phi_CTE));
      else
            new_dr(ii) = new_dr(ii-1);
      end;
  end;                          % End of CTE Controller

  % Use LOS if near to loiter point
  % if (loiter==1)& new_s(ii)<10;  new_dr(ii)=new_drlos(ii);end;

  % Surface Phase Logic (Independent of LOS or CTE)

  if(SurfPhase == TRUE)
     if(new_SURFACE_TIMER_ACTIVE == FALSE)
        if(Ratio > 40.0)
          % Start a Timer
          new_SURFACE_TIMER_ACTIVE = TRUE;
          new_Depth_com(ii) = 0.0;
          new_SurfaceWait = new_SurfaceTime(1) + tt(ii);
          new_SurfaceWait
        end;
     end;
  end;
  if(new_SURFACE_TIMER_ACTIVE == TRUE)
    if(tt(ii) >= new_SurfaceWait)
      new_SURFACE_TIMER_ACTIVE = FALSE;
      new_Depth_com(ii) = new_WayPointVertDist_com(1);
      new_SurfPhase(1) = 0;
      else
         new_Depth_com(ii) = 0.0;
    end;
  end;
  if(abs(new_dr(ii)) > 0.4)
     new_dr(ii) = 0.4*sign(new_dr(ii));
  end;

  % Model drl is the actual lagged rudder, dr is the rudder command.
  % taudr = 0.255;

  %  drl(i+1) = drl(i) + dt*(dr(i)-drl(i))/taudr;
  %         if(abs(drl(i)) > 0.4)
  %                     drl(i) = 0.4*sign(drl(i));
  %         end;
   new_x_dot(:,ii+1) = [ A(1,1)*new_v(ii) + A(1,2)*new_r(ii) + B(1)*...
         new_dr(ii); A(2,1)*new_v(ii) + A(2,2)*new_r(ii) + B(2)*...
         new_dr(ii); new_r(ii)];
   new_x(:,ii+1) = new_x(:,ii)+dt*new_x_dot(:,ii);
   new_v(ii+1)   = new_x(1,ii+1)/12;
```

```
  new_r(ii+1)   = new_x(2,ii+1);
  new_psi(ii+1) = new_x(3,ii+1);
  new_rRM(ii+1) = new_r(ii+1);

 % Added
 % rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*dr(i));
 % psi(i+1) = psi(i) + dt*rRM(i);

 % Throw in some Waves
 % Uc(i) = -0.5*sin(2*pi*t(i)/5);
 % Vc(i) =  0.5*sin(2*pi*t(i)/5);

 % Model using system ID results from Bay tests

 % rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*drl(i));
 % psi(i+1) = psi(i) + dt*rRM(i);
 % side slip added proprtional to turn rate from AZORES data V in ft/sec
 % v(i+1) = 1.0*rRM(i+1)*3.28;

  Uc = 0.0;
  Vc = 0.0;

  % Kinematics

  New_X(ii+1) = New_X(ii) + (Uc + (U/3.28)*cos(new_psi(ii)) - new_v(ii)...
     /3.28*sin(new_psi(ii)) )*dt;
  New_Y(ii+1) = New_Y(ii) + (Vc + (U/3.28)*sin(new_psi(ii)) + new_v(ii)...
     /3.28*cos(new_psi(ii)) )*dt;

  % Check to See if we are Within the Watch_Radius (set to 1 foot here)

  if(sqrt(New_X_Way_Error(ii)^2.0 + New_Y_Way_Error(ii)^2.0)...
        <= 1 | new_s(ii) < 0.0),

  % Next Line ends mission if within Watch Radius.

  disp(sprintf('WayPoint %d Reached',jj)); break;
    if(jj==No_tracks),
      PLOT_PART = 2;
      disp(sprintf('PLOT_PART = 2'));
      break;
    end;
    New_PrevX_Way_c(jj+1) = New_X_Way_c(jj);
    New_PrevY_Way_c(jj+1) = New_Y_Way_c(jj);
 end;
end
end
%*************************************************************************
dr(i+1) = dr(i);
cte(i+1) = cte(i);
s(i+1) = s(i);
ss(i+1) = ss(i);

%new_dr(ii+1) = new_dr(ii);
%new_cte(ii+1) = new_cte(ii);
%new_s(ii+1) = new_s(ii);
```

```
%new_ss(ii+1) = new_ss(ii);

% Plotting

if PLOT_PART == 1,

   % Plot of Time vs Rudder Angle & Vehicle Heading

   figure(1);
   plot(t([1:i+1]),psi*180/pi);
   hold;
   plot(t([1:i+1]),dr*180/pi,'r:');grid;
   title('Time vs Rudder Angle and Vehicle Heading');
   xlabel('Time (sec)'); ylabel('Rudder Angle/Vehicle Heading (degrees)');
   legend('Vehicle Heading', 'Rudder Angle');
   print -tiff -depsc figure1b
   hold;zoom on;

   % Plot of Time vs CTE, Distance to Go to Projected Track, Radial
   % Distance to Next Waypoint

   figure(2);
   plot(t([1:i+1]),cte);
   hold;
   plot(t([1:i+1]),s,'r:');
   plot(t([1:i+1]),ss,'g--');grid;
   title('Time vs Cross Track Error')
   xlabel('Time (sec)');ylabel('Distance (feet)');
   legend('Cross Track Error', 'Distance to Go Projected to Track', ...
   'Radial Distance to Go to Next Way Point');
   print -tiff -depsc figure2b
   hold;zoom on;

elseif PLOT_PART == 0,

   % Plot of Time vs Rudder Angle & Vehicle Heading for Rendezvous Mission

   figure(3);
   plot(t([1:i+1]),psi*180/pi, 'y');
   hold;
   plot(tt([1:ii+1]),new_psi*180/pi, 'y:');
   plot(t([1:i+1]),dr*180/pi,'r');
   plot(tt([1:ii]),new_dr*180/pi, 'r:');
   title('Time vs Rudder Angle and Vehicle Heading - 0');
   xlabel('Time (sec)'); ylabel('Rudder Angle/Vehicle Heading (degrees)');
   legend('Vehicle Heading Before Mission Abort',...
      'Vehicle Heading After Mission Abort',...
      'Rudder Angle Before Mission Abort',...
      'Rudder Angle After Mission Abort');
   print -tiff -depsc figure3b
   hold;grid;

   % Plot of Time vs CTE, Distance to Go to Projected Track, Radial
   % Distance to Next Waypoint for Rendezvous Mission

   figure(4);
```

```
  plot(t([1:i+1]), cte, 'y');
  hold;
  plot(tt([1:ii]), new_cte, 'y:');
  plot(t([1:i+1]), s,'r');
  plot(tt([1:ii]), new_s, 'r:');
  plot(t([1:i+1]), ss,'g');grid;
  plot(tt([1:ii]), new_ss, 'g:');
  title('Time vs Cross Track Error - 0')
  xlabel('Time (sec)'); ylabel('Distance (feet)');
  legend('Cross Track Error Before Abort',...
     'Cross Track Error After Abort',...
     'Distance to Go Projected to Track Before Abort', ...
     'Distance to Go Projected to Track After Abort',...
     'Radial Distance to Go to Next Way Point Before Abort',...
     'Radial Distance to Go to Next Way Point After Abort');
  print -tiff -depsc figure4b
  hold;zoom on;

end;

  % Plot of Actual Track and Planned Track
  % Modified on 05 Feb 2002 - To include waypoint2.m modifications.

  if PLOT_PART == 1,
    figure(5);
    plot(Y,X,'b--');grid;                    % Actual Track
    title('ARIES Track - Actual and Planned');
    xlabel('Y (feet)');ylabel('X (feet)');
    hold;

  % Planned Track
    plot([Y_Way_c(1) PrevY_Way_c(1)],[X_Way_c(1) PrevX_Way_c(1)],'r');
    for ii=2:No_tracks,
      plot([Y_Way_c(ii) Y_Way_c(ii-1)],[X_Way_c(ii) X_Way_c(ii-1)],'r');
    end;
    legend('Actual Track', 'Planned Track',4);
    print -tiff -depsc figure5b
    hold; zoom on

  elseif PLOT_PART == 0 | PLOT_PART == 2,

    % Plot of Planned Track, Track after Mission Change, Rendezvous
    % Point and Initial Track

    figure(6);
    plot(Y,X,'b--');grid;                    % Actual Track
    title('ARIES Track - Actual and Planned');
    xlabel('Y (feet)');ylabel('X (feet)');
    hold;
    plot(New_Y, New_X,'g-.');
    plot(New_Y_Way_c, New_X_Way_c,'d');

  % Planned Track
  plot([Y_Way_c(1) PrevY_Way_c(1)],[X_Way_c(1) PrevX_Way_c(1)],'r');
  for ii=2:No_tracks,
    plot([Y_Way_c(ii) Y_Way_c(ii-1)],[X_Way_c(ii) X_Way_c(ii-1)],'r');
```

```
  end;
  legend('Initial Track', 'Track After Mission Change'...
     , 'Rendezvous Point', 'Planned Track',4);
  AXIS([-100 250 -80 60])
  print -tiff -depsc figure6b
  hold; zoom on
%     plot([Y_Way_c(2) Y_Way_c(1)],[X_Way_c(2) X_Way_c(1)],'r');
%     plot([Y_Way_c(3) Y_Way_c(2)],[X_Way_c(3) X_Way_c(2)],'r');
%     plot([Y_Way_c(4) Y_Way_c(3)],[X_Way_c(4) X_Way_c(3)],'r');
%     plot([Y_Way_c(5) Y_Way_c(4)],[X_Way_c(5) X_Way_c(4)],'r');
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. MATLAB FILE COEFFICIENTS.M

This appendix contains a MATLAB file that finds the coefficients of the longitudinal equation of motion using the method of least squares. It then disregards those values and uses the hand-manipulated values in order to produce a model for speed.

```
% This version doubles all the RPS values based on the Tecnadyne Data.

% Loads the 3 .d files from the experiment conducted on 17APR02.

load d041702_03.d;
load d041702_04.d;
load d041702_05.d;
a = d041702_03;          % 1st run
b = d041702_04;          % 2nd run
c = d041702_05;          % 3rd run

% a(:,31) is left screw voltage for 1st run
% a(:,32) is right screw voltage for 1st run
% a(:,17) is u in m/s

% First Calculate Average Thruster RPS

a(:,39) = ((a(:,31)*133.8047)+(a(:,32)*124.4615))/60;  %Avg rps, Run 1
b(:,39) = ((b(:,31)*133.8047)+(b(:,32)*124.4615))/60;  %Avg rps, Run 2
c(:,39) = ((c(:,31)*133.8047)+(c(:,32)*124.4615))/60;  %Avg rps, Run 3

[j,k] = size(a);                % Figures size of data matrices
[jj, kk] = size(b);
[jjj, kkk] = size(c);

u1 = []; u2 = []; u3 =[]; uu1 =[]; uu2 = []; uu3 =[];
y1 = []; y2 = []; y3 = [];

%.....................................................................

% EOM for Longitudinal Motion:

% (m-X_udot_r)udot_r = [X_u * u_r * abs(u_r)] + [alpha * n * abs(n)] -
%     [gamma * abs(n) * u_r]

% (m-X_udot_r) = 215.47 kg   ==>>  z = 1/(m-X_udot_r) = 0.004641

% udot_r = (u_r(t+1) - u_r(t))/dt  therefore,

% u_r(t+1) - u_r(t) =  (z*dt)[(X_u * f(u_r)) + (alpha * n * abs(n)) -
%     (gamma * n * abs(u_r))

% dt = 0.125  ==>>  (z*dt) = 0.004641 * 0.125.

% In matrix form:  y = H * Theta

% where H = (z*dt)[f(u_r) n*abs(n) n*abs(u_r)] and Theta = [X alpha gamma]'
% and y = u_r(t+1) - u_r(t)
```

```
z = 0.004641;
dt = 0.125;

% Calculations for Run 1

for n1 = 2:j;
   u1(n1-1) = a(n1,17);                % u_r(t+1)
   n1 = n1+1;
end

u1 = u1';                          % Makes u1 a j-1 x 1 matrix

H11 = []; H12 = []; H13 = []; H1 = [];

for nn1 = 1:j-1;
   H11(nn1) = (z*dt) * (a(nn1,17)*abs(a(nn1,17)));
   H12(nn1) = (z*dt) * (a(nn1,39)* abs(a(nn1,39)));
   H13(nn1) = (z*dt) * (a(nn1,17)*abs(a(nn1,39)));
   uu1(nn1) = a(nn1,17);               % u_r(t)
   nn1 = nn1+1;
end

uu1 = uu1';                        % Makes uu1 a j-1 x 1 matrix
y1 = u1-uu1;                        % y = u_r(t+1) - u_r(t)
H1 = [H11' H12' H13'];                 % Makes H a j-1 x 3 matrix

theta_hat1 = inv(H1' * H1) * H1' * y1;    % 3 x 1 matrix
error1 = y1 - (H1 * theta_hat1);
%.....................................................................

% Calculations for Run 2

for n2 = 2:jj;
   u2(n2-1) = b(n2,17);                % u_r(t+1)
   n2 = n2+1;
end

u2 = u2';                          % Makes u2 a jj-1 x 1 matrix

H21 = []; H22 = []; H23 = []; H2 = [];

for nn2 = 1:jj-1;
   H21(nn2) = (z*dt) * (b(nn2,17)*abs(b(nn2,17)));
   H22(nn2) = (z*dt) * (b(nn2,39)*abs(b(nn2,39)));
   H23(nn2) = (z*dt) * (b(nn2,17)*abs(b(nn2,39)));
   uu2(nn2) = b(nn2,17);               % u_r(t)
   nn2 = nn2+1;
end

uu2 = uu2';                        % Makes uu2 a jj-1 x 1 matrix
y2 = u2-uu2;                        % y = u_r(t+1) - u_r(t)
H2 = [H21' H22' H23'];                 % Makes H a jj-1 x 3 matrix

theta_hat2 = inv(H2' * H2) * H2' * y2;    % 3 x 1 matrix
error2 = y2 - (H2 * theta_hat2);
%.....................................................................
```

```
% Calculations for Run 3

for n3 = 2:jjj;
   u3(n3-1) = c(n3,17);                % u_r(t+1)
   n3 = n3+1;
end

u3 = u3';                              % Makes u3 a jjj-1 x 1 matrix

H31 = []; H32 = []; H33 = []; H3 = [];

for nn3 = 1:jjj-1;
   H31(nn3) = (z*dt) * (c(nn3,17)*abs(c(nn3,17)));
   H32(nn3) = (z*dt) * (c(nn3,39)*abs(c(nn3,39)));
   H33(nn3) = (z*dt) * (c(nn3,17)*abs(c(nn3,39)));
   uu3(nn3) = c(nn3,17);               % u_r(t)
   nn3 = nn3+1;
end

uu3 = uu3';                            % Makes uu3 a jjj-1 x 1 matrix
y3 = u3-uu3;                           % y = u_r(t+1) - u_r(t)
H3 = [H31' H32' H33'];                 % Makes H a jjj-1 x 3 matrix


theta_hat3 = inv(H3' * H3) * H3' * y3;    % 3 x 1 matrix
error3 = y3 - (H3 * theta_hat3);

%.....................................................................

% Summary of coefficients for all 3 runs and average value.

avg =[];
format
avg_X = (theta_hat1(1,1) + theta_hat2(1,1) + theta_hat3(1,1))/3;
avg_a = (theta_hat1(2,1) + theta_hat2(2,1) + theta_hat3(2,1))/3;
avg_g = (theta_hat1(3,1) + theta_hat2(3,1) + theta_hat3(3,1))/3;
avgs = [avg_X avg_a avg_g];
avgs = avgs';
all_theta_hats = [theta_hat1 theta_hat2 theta_hat3 avgs]

% all_theta_hats =
%      Run 1     Run 2     Run 3      Avg
%   -13.4933  -16.7915  -16.7587  -15.6812  X_u
%     0.4445    0.3115    0.1957    0.0793  alpha
%     1.6506    2.6863    3.1466    1.2473  gamma

%a(39) is data for revs
%a(17) is the speed
%Prediction

% Manipulated Values
avg_X = -10.5;                         % Based on C_D of 0.2
avg_g = 0.05;
avg_a=0.155;
model_u1 = []; model_u2 = []; model_u3 = [];
```

```
% Run 1 Model
model_u1(1) = a(1,17);
for m = 1:j-1;
   model_u1(m+1) = (z*dt)*((avg_X * model_u1(m) * abs(model_u1(m))) +...
      (avg_a * a(m,39) * abs(a(m,39))) - (avg_g * a(m,39) *...
      abs(model_u1(m)))) + model_u1(m);
   m = m +1;
end
model_u1 = model_u1';

% Run 2 Model
model_u2(1) = b(1,17);
for mm = 1:jj-1;
   model_u2(mm+1) = (z*dt)*((avg_X * model_u2(mm) * abs(model_u2(mm)))+...
      (avg_a * b(mm,39) * abs(b(mm,39))) - (avg_g * b(mm,39) * ...
      abs(model_u2(mm)))) + model_u2(mm);
   mm = mm +1;
end
model_u2 = model_u2';

% Run 3 Model
model_u3(1) = c(1,17);
for mmm = 1:jjj-1;
   model_u3(mmm+1) = (z*dt)*((avg_X * model_u3(mmm) * ...
      abs(model_u3(mmm))) + (avg_a * c(mmm,39) * abs(c(mmm,39))) -...
      (avg_g * c(mmm,39) * abs(model_u3(mmm)))) + model_u3(mmm);
   mmm = mmm +1;
end
model_u3 = model_u3';

figure(1)
orient tall
subplot(3,1,1)
plot(model_u1(:,1))
hold
plot(a(2:j,17), 'r'); grid
legend('Model', 'Actual', 4)
title('Vehicle Speed - Run 1'); xlabel('Time Units');
ylabel('Long. Speed')
hold

subplot(3,1,2)
plot(model_u2(:,1))
hold
plot(b(2:jj,17), 'r'); grid
legend('Model', 'Actual', 4)
title('Vehicle Speed - Run 2'); xlabel('Time Units');
ylabel('Long. Speed')
hold

subplot(3,1,3)
plot(model_u3(:,1))
hold
plot(c(2:jjj,17), 'r'); grid
legend('Model', 'Actual', 4)
```

```
title('Vehicle Speed - Run 3'); xlabel('Time Units');
ylabel('Long. Speed')
print -tiff -depsc modelplot
hold
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. MATLAB FILE FINALRENDEZVOUS.M

This appendix contains the MATLAB code that produces a simulation of a rendezvous in space and time using a sliding mode control speed controller.

```
% This is the final program.  It adds an updated speed controller based on
% the control law and the longitudinal equation of motion.

% 15 May 2002

whitebg('k');
% State = [v r psi]
clear all
TRUE  = 1;
FALSE = 0;

% Converts Degrees to Radians & Radians to Degrees

DegRad = pi/180;
RadDeg = 180/pi;

% State Model Parameters

W   = 600.0;                % Weight in LB
U = 1.4*3.28;               % Forward Speed in ft/s (1.4 m/s)
g = 32.174;                 % Gravity in ft/sec^2
Boy = 500.0;                % Bouyancy ?
xg  = 0.125/12.0;           % ??
m = W/g;                    % Mass
rho = 1.9903;               % Density of Seawater in slugs/ft^3
L = 10;                     % Length in ft of ARIES
Iz = (1/12)*m*(1.33^2 + 10^2);  % Approx. Using I = 1/12*m*(a^2 + b^2)
% where a is width & b is length
Iz = Iz*5.0;

% Coefficients

Yv_dot = -0.03430*(rho/2)*L^3;  % Added Mass in Sway Coefficient.
Yr_dot = -0.00178*(rho/2)*L^4;  % Added Mass in Yaw Coefficient.
Yv = -0.10700*(rho/2)*L^2;      % Coeff. of Sway Force induced by Side Slip
Yr = 0.01187*(rho/2)*L^3;       % Coeff. of Sway Force induced by Yaw
Ydrs = (0.01241*(rho/2)*L^2)/2.0; % Since Bow & Stern Lower Rudders Removed
Ydrb = (0.01241*(rho/2)*L^2)/2.0; % So don't use these equations

Nv_dot = -0.00178*(rho/2)*L^4; % Added Mass Moment of Inertia in Sway Coeff
%Nr_dot = -0.00047*(rho/2)*L^5;
Nr_dot = -Iz;                   % Added Mass Moment of Inertia in Yaw Coeff
Nv = -0.00769*(rho/2)*L^3;      % Coeff. of Sway Moment from Side Slip
Nr = -0.00390*(rho/2)*L^4;      % Coeff. of Sway Moment from Yaw
%Ndrs = -2.6496/2.0;            % Since Bow & Stern Lower Rudders Removed
%Ndrb = 1.989/2.0;

% Below Modified on 7/12/00 The 3.5 and 3.4167 is the Moment Arm Length
% in Feet - Since Bow & Stern Lower Rudders Removed
```

```
Ndrs = -0.01241*(rho/2)*(L^2)*(3.5)/2.0;
Ndrb = 0.01241*(rho/2)*(L^2)*(3.4167)/2.0;

% Combining Stern & Bow Rudder Effectivness

Ndr = Ndrs - Ndrb;
Ydr = Ydrs - Ydrb;                    % Cancel Out

% Matrices

m1 = m - Yv_dot;
m2 = m*xg - Yr_dot;
m3 = m*xg - Nv_dot;
m4 = Iz - Nr_dot;
Y1 = Yv;
Y2 = Yr;
Y3 = U^2*Ydr;
N1 = Nv;
N2 = Nr;
N3 = U^2*Ndr;
A = [Y1*U Y2*U;N1*U N2*U];
B = [Y3 N3]';
M = [m1 m2;m3 m4];
A1 = inv(M)*A;
B1 = inv(M)*B;
AO = [A1(1,1) A1(1,2) 0;
   A1(2,1) A1(2,2) 0;
   0 1 0];
BO = [B1;0];
dt = 0.125;
t = [0:dt:1000]';
size(t);

% Set initial conditions

start=10;
v(1)   = 0.0;                 % Initial Side Slip Velocity
r(1)   = 0.0;                 % Initial Yaw
U(1)   = 1.4*3.28;            % Initial Forward Speed
rRM(1) = r(1);
psi(1) = 50.0*DegRad;         % Initial Heading of ARIES
X(1) = -80.0;                 % Initial Position in meters
Y(1) = 0.0;
ucom=[];
% Convert to Feet ?

% This data from track.out file for ARIES in Waiting Pattern
% (12 Mar 02)

No_tracks=4;                  % Sets # of Tracks = # of Rows

Track=[ 50.0  0.0   2.75 2.75  0  1.25  1.00 0 25.00 8.00 40.00
   50.0 -60.0  2.75 2.75  0  1.25  1.00 0 25.00 8.00 200.00
   -70.0 -60.0  2.75 2.75  0  1.25  1.00 0 25.00 2.00 200.00
   -70.0  0.0  2.75 2.75  0  1.25  1.00 0 25.00 2.00 40.00];
```

```
track=Track(:,1:2);            % Defines track as Track(X,Y)
SurfaceTime = Track(:,9);      % Col 9 of Track is Surface Time for Pop-up
SurfPhase   = Track(:,8);      % Col 8 of Track designates if Pop-up

% This is the REMUS Search Pattern
% 12 Mar 02

Y_REMUS = [200 50 50 200 200 50 50 200 200 50 50 200 200 50];
X_REMUS = [50 50 30 30 10 10 -10 -10 -30 -30 -50 -50 -70 -70];

% Read in way points from track data assumes track is loaded

for j=1:No_tracks,
   X_Way_c(j)     = track(j,1);
   Y_Way_c(j)     = track(j,2);
end;

% Set start position

PrevX_Way_c(1) = -80.0;        % meters
PrevY_Way_c(1) =  00.0;        % meters
r_com = 0.0;
W_R = 10.0;                    % Sets initial Watch Radius (meters)
a = -.3;
b = (9/24)*a;
x(:,1) = [v(1);r(1);psi(1)];

% Below are in British Units for CTE Sliding Mode
%Lam1 = 0.75;
%Lam2 = 0.5;
Lam1 = 2.0;
Lam2 = 1.0;
Eta_FlightHeading = 1.0;
Phi_FlightHeading = 0.5;

% Below for tanh

Eta_CTE = 0.1;
Eta_CTE_Min = 1.0;
Phi_CTE = 0.5;
Uc = [];
Vc = [];
PLOT_PART = 0; disp(sprintf('PLOT_PART = 0'));

% Total Track Length between initial waypoint and waypoint (1) - Eq (10)

SegLen(1) = sqrt((X_Way_c(1)-PrevX_Way_c(1))^2+(Y_Way_c(1)...
   -PrevY_Way_c(1))^2);

% Track Angle of first track - Eq (11)

psi_track(1) = atan2(Y_Way_c(1)-PrevY_Way_c(1),X_Way_c(1)-PrevX_Way_c(1));

% Computes track lengths and track angles for each track

for j=2:No_tracks,
```

```
   SegLen(j) = sqrt((X_Way_c(j)-X_Way_c(j-1))^2+(Y_Way_c(j)-...
      Y_Way_c(j-1))^2);
   psi_track(j) = atan2(Y_Way_c(j)-Y_Way_c(j-1),X_Way_c(j)-X_Way_c(j-1));
end;
j=1;
Sigma = [];
Depth_com = [];
dr=[];
drl = [];
drl(1) = 0.0;
Depth_com(1) = 5.0;
WayPointVertDist_com = [5.0 5.0 5.0 5.0 5.0];
SURFACE_TIMER_ACTIVE = FALSE;

% Starts a loop that computes values for each data point corresponding to
% a time value (in this case, every 0.125 seconds from 1 to 1000 seconds)

for i=1:length(t)-1,
   Depth_com(i) = WayPointVertDist_com(j);

   % Difference between current vehicle position & the next
   % waypoint Eq(13)

   X_Way_Error(i) = X_Way_c(j) - X(i);
   Y_Way_Error(i) = Y_Way_c(j) - Y(i);

   % DeWrap psi to within +/- 2.0*pi; Makes Heading Angle to lie between
   % 0-360 degrees

   psi_cont(i) = psi(i);
   while(abs(psi_cont(i)) > 2.0*pi)
      psi_cont(i) = psi_cont(i) - sign(psi_cont(i))*2.0*pi;
   end;

   % Cross Track Heading Error Eq(12)

   psi_errorCTE(i) = psi_cont(i) - psi_track(j);

   % DeWrap psi_error to within +/- pi;  Normalized to Lie between +/- 180
   % degrees

   while(abs(psi_errorCTE(i)) > pi)
      psi_errorCTE(i) = psi_errorCTE(i) - sign(psi_errorCTE(i))*2.0*pi;
   end;

   % **  Always Calculate this (What is This?)
   Beta = v(i)/U(i);
   %  Beta = 0.0;
   cpsi_e = cos(psi_errorCTE(i)+Beta);
   spsi_e = sin(psi_errorCTE(i)+Beta);

   % Distance to the ith way point projected to the track line S(t)i -
   % Eq (14)

   s(i) = [X_Way_Error(i),Y_Way_Error(i)]*[(X_Way_c(j)-...
         PrevX_Way_c(j)),(Y_Way_c(j)-PrevY_Way_c(j))]';
```

```
% s is distance to go projected to track line
% (goes from 0-100%L) - Eq (14)

s(i) = s(i)/SegLen(j);
Ratio=(1.0-s(i)/SegLen(j))*100.0;     % Ranges from 0-100% of SegLen

% Radial distance to go to next WP

ss(i) = sqrt(X_Way_Error(i)^2 + Y_Way_Error(i)^2);

% dp is angle between line of sight and current track line - Eq (16)

dp(i) = atan2( (Y_Way_c(j)-PrevY_Way_c(j)),(X_Way_c(j)-...
   PrevX_Way_c(j)) )- atan2( Y_Way_Error(i),X_Way_Error(i) );
if(dp(i) > pi),
   dp(i) = dp(i) - 2.0*pi;
end;

% Cross Track Error Definition - Eq (15)

cte(i) = s(i)*sin(dp(i));

% If the magnitude of the CTE Heading exceeds 40 degrees, a LOS
% Controller is used.

if( abs(psi_errorCTE(i)) >= 40.0*pi/180.0 | s(i) < 0.0 ),
   LOS(i) = 1;
   psi_comLOS = atan2(Y_Way_Error(i),X_Way_Error(i));    % Eq (22)
   psi_errorLOS(i) = psi_comLOS - psi_cont(i);           % Eq (23)
   % LOS Error
   if(abs(psi_errorLOS(i)) > pi),
      psi_errorLOS(i) = psi_errorLOS(i) - 2.0*pi*psi_errorLOS(i)...
         /abs(psi_errorLOS(i));
   end;

   % Eq (8)

   Sigma_FlightHeading = 0.9499*(r_com - r(i)) + 0.1701*...
      psi_errorLOS(i);

   % Eq (9)

   dr(i) = -1.5435*( 2.5394*r(i)+ Eta_FlightHeading*tanh...
      (Sigma_FlightHeading/Phi_FlightHeading));

else

   % Use CTE Controller if CTE Heading is less than 40 degrees

   LOS(i) = 0;
   if(cpsi_e ~= 0.0),              % Trap Div. by Zero !

      % SMC Soln

      % Sliding Surface - Eq (20)
```

```
        Sigma(i) = U(i)*rRM(i)*cpsi_e + Lam1*U(i)*spsi_e + 3.28*Lam2...
            *cte(i);

        % Rudder Input - Eq (21)

        dr(i) = (1.0/(U(i)*b*cpsi_e))*(-U(i)*a*rRM(i)*cpsi_e + U(i)*...
            rRM(i)^2*spsi_e - Lam1*U(i)*rRM(i)*cpsi_e - Lam2*U(i)*...
            spsi_e - Eta_CTE*(Sigma(i)/Phi_CTE));
    else
        dr(i) = dr(i-1);
    end;
end;                            % End of CTE Controller

% Use LOS if near to loiter point
% if (loiter==1)& s(i)<10;  dr(i)=drlos(i);end;

% Surface Phase Logic (Independent of LOS or CTE)

if(SurfPhase(j) == TRUE)
    if(SURFACE_TIMER_ACTIVE == FALSE)
        if(Ratio > 40.0)
            % Start a Timer
            SURFACE_TIMER_ACTIVE = TRUE;
            Depth_com(i) = 0.0;
            SurfaceWait = SurfaceTime(j) + t(i);
            SurfaceWait
        end;
    end;
end;
if(SURFACE_TIMER_ACTIVE == TRUE)
    if(t(i) >= SurfaceWait)
        SURFACE_TIMER_ACTIVE = FALSE;
        Depth_com(i) = WayPointVertDist_com(j);
        SurfPhase(j) = 0;
    else
        Depth_com(i) = 0.0;
    end;
end;
if(abs(dr(i)) > 0.4)
    dr(i) = 0.4*sign(dr(i));
end;

% Model drl is the actual lagged rudder, dr is the rudder command.
% taudr = 0.255;

%  drl(i+1) = drl(i) + dt*(dr(i)-drl(i))/taudr;
%       if(abs(drl(i)) > 0.4)
%                   drl(i) = 0.4*sign(drl(i));
%       end;

% Jay Johnson Model

Yv  = -68.16;
Yr  = 406.3;
Ydr = 70.0;
```

```
Nv  = -10.89;
Nr  = -88.34;
Ndr = -35.47;

MY = 456.76;
IN = 215;

M = diag([MY,IN,1]);
AA = [Yv,Yr,0;Nv,Nr,0;0,1,0];
BB = [Ydr;Ndr;0];
A  = inv(M)*AA;
B  = inv(M)*BB;

% x_dot(:,i+1) = [ A(1,1)*v(i) + A(1,2)*r(i) + B(1)*drl(i);
%                  A(2,1)*v(i) + A(2,2)*r(i) + B(2)*drl(i);
%                  r(i)];
x_dot(:,i+1) = [ A(1,1)*v(i) + A(1,2)*r(i) + B(1)*dr(i);
   A(2,1)*v(i) + A(2,2)*r(i) + B(2)*dr(i);
   r(i)];

x(:,i+1) = x(:,i)+dt*x_dot(:,i);
v(i+1)   = x(1,i+1)/12;
r(i+1)   = x(2,i+1);
U(i+1) = 1.4*3.28;        % Constant speed of 2.72 knots
psi(i+1) = x(3,i+1);
rRM(i+1) = r(i+1);

% Added
% rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*dr(i));
% psi(i+1) = psi(i) + dt*rRM(i);

% Throw in some Waves
% Uc(i) = -0.5*sin(2*pi*t(i)/5);
% Vc(i) =  0.5*sin(2*pi*t(i)/5);

% Model using system ID results from Bay tests

% rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*drl(i));
% psi(i+1) = psi(i) + dt*rRM(i);
% side slip added proprtional to turn rate from AZORES data V in ft/sec
% v(i+1) = 1.0*rRM(i+1)*3.28;

Uc = 0.0;
Vc = 0.0;

% Kinematics

X(i+1) = X(i) + (Uc + (U(i)/3.28)*cos(psi(i)) - v(i)/3.28*sin(psi(i))...
   )*dt;
Y(i+1) = Y(i) + (Vc + (U(i)/3.28)*sin(psi(i)) + v(i)/3.28*cos(psi(i))...
   )*dt;

%*********************************************************************

% This should abort @ 30 seconds if input is empty or 'Y'
% This modification done on 05 Feb 2002 - original file is waypoint1.m
```

```
   if i == 240,
      K = input(...
         'Is Rendezvous Required? (Enter 1 for Yes, 0 for No)-->>');
      if isempty(K)==1; K = 1; break; end;
      if K == 1; break;
      else i = i; end;
   end

   %*********************************************************************

   % Check to See if we are Within the Watch_Radius

   if(sqrt(X_Way_Error(i)^2.0 + Y_Way_Error(i)^2.0) <= W_R | s(i) < 0.0),
      disp(sprintf('WayPoint %d Reached',j));
      if(j==No_tracks),
         PLOT_PART = 1;
         disp(sprintf('PLOT_PART = 1'));
         break;
      end;
      PrevX_Way_c(j+1) = X_Way_c(j);
      PrevY_Way_c(j+1) = Y_Way_c(j);
      j=j+1;
   end;

end; %end of i loop

%**************************************************************************

% Requests Rendezvous Point Information
% This modification done on 05 Feb 2002 - original file waypoint2.m

if j == No_tracks,
   disp(sprintf('Mission Complete'));
else
   new_r_com = 0.0;
   new_v(1) = v(i+1);
   new_r(1) = r(i+1);
   new_rRM(1) = new_r(1);
   new_psi(1) = psi(i+1);
   New_X(1) = X(i+1);
   New_Y(1) = Y(i+1);
   New_No_Tracks = 1;
   New_Track = input('Enter 12 Column Track, i.e., [1 1 ...]-->>');
   new_track = New_Track(:,1:2);
   new_SurfaceTime = New_Track(:,9);
   new_Surfphase = New_Track(:,8);

   %*********************************************************************
   new_U(1) = U(i+1)/3.28;      % Sets Initial Rendezvous Speed to 1.4 m/s
   % new_U in meters/sec.
   new_time(1) = New_Track(:,12);  % Desired Time to Rendezvous in seconds
   overall_distance_travelled(1) = 0;
   distance_travelled(1) = 0;
   time_used(1) = 0;
   time_remaining(1) = new_time(1);
```

```
real_time(1) = 0;
accel(1) = 0.03;              % Sets Acceleration to 0.03 m/s^2
decel(1) = 0.0249;
ncom(1) = 12;
% Sets Deceleration to 0.0249 m/s^2
%*********************************************************************

for jj = 1:New_No_Tracks,
   New_X_Way_c(jj) = new_track(jj,1);
   New_Y_Way_c(jj) = new_track(jj,2);
end;
New_PrevX_Way_c(1) = X(i+1);       % Sets Abort Posit as start of new
New_PrevY_Way_c(1) = Y(i+1);       % track.
% Below for tanh
new_Eta_CTE = 0.1;
new_Eta_CTE_Min = 1.0;
new_Phi_CTE = 0.5;
PLOT_PART = 0; disp(sprintf('PLOT_PART = 0'));
new_x(:,1) = [new_v(1); new_r(1); new_psi(1)];

% Total Track Length between abort point and rendezvous point

New_SegLen(1) = sqrt((New_X_Way_c(1)-New_PrevX_Way_c(1))^2+...
   (New_Y_Way_c(1) - New_PrevY_Way_c(1))^2)

% Track Angle of track between abort point and rendezvous point

new_psi_track(1) = atan2(New_Y_Way_c(1)-New_PrevY_Way_c(1),...
   New_X_Way_c(1)-New_PrevX_Way_c(1));

%*********************************************************************
% Determines if the Mission is Feasible - Can Distance be covered in
% time required traveling at maximum speed of 3.5 knots or Is time
% required to be at the rendezvous too much for vehicle travelling at
% minimum speed of 0.5 knots? (28 Feb 02)
% Added 10 extra meters to account for curvature of path (14 May 02)

if new_time > ((New_SegLen(1)+10)/(1.8)) &...
      new_time < ((New_SegLen(1)+10)/(0.2571)),
   disp('Mission Feasible');
else disp('Mission Not Feasible');
   break;                     % Ends Simulation if Not Feasible
end

%*********************************************************************
% Determines if there is enough length to achieve deceleration or
% acceleration - (12 Mar 02)

decel_Len = abs(((0.2571)^2 - (1.4)^2)/(2*0.0249));
accel_Len = ((1.8)^2 - (1.4)^2)/(2*0.03);
time_of_decel = 114;        %Time to decel from 1.4 m/s to 0.2571 m/s
time_left = New_Track(:,12)-time_of_decel;
distance_remaining = New_SegLen(1)-decel_Len;
if distance_remaining > 0.2571 * time_left,
   disp('Mission Feasible for Deceleration');
else disp('Mission Not Feasible for Deceleration');
```

```
   break;
end

%*****************************************************************

% Starts loop that computes values for each data point corresponding to
% a time value along new track

jj=1;
new_Sigma = [];
new_Depth_com = [];
new_dr=[];
new_drl = []; n = [];
new_drl(1) = 0.0;
new_Depth_com(1) = 5.0;
new_WayPointVertDist_com = [5.0 5.0 5.0 5.0 5.0];
new_SURFACE_TIMER_ACTIVE = FALSE;
tt = [t(i+1):dt:3*new_time]';
size(tt);

% Start of Loop for Rendezvous Point

for ii = 1:length(tt)-1,

   new_Depth_com(ii) = new_WayPointVertDist_com(jj);
   New_X_Way_Error(ii) = New_X_Way_c(jj) - New_X(ii);
   New_Y_Way_Error(ii) = New_Y_Way_c(jj) - New_Y(ii);
   new_psi_cont(ii) = new_psi(ii);
   while(abs(new_psi_cont(ii)) > 2.0*pi)
      new_psi_cont(ii) = new_psi_cont(ii) - sign(new_psi_cont(ii))*...
         2.0*pi;
   end;

   % Cross Track Heading Error Eq(12)

   new_psi_errorCTE(ii) = new_psi_cont(ii) - new_psi_track(jj);

   % DeWrap psi_error to within +/- pi;  Normalized to Lie between +/-
   % 180 degrees

   while(abs(new_psi_errorCTE(ii)) > pi)
      new_psi_errorCTE(ii) = new_psi_errorCTE(ii) - sign(...
         new_psi_errorCTE(ii))*2.0*pi;
   end;

   % **  Always Calculate this (What is This?)
   new_Beta = new_v(ii)/new_U(ii);
   %   Beta = 0.0;
   new_cpsi_e = cos(new_psi_errorCTE(ii)+new_Beta);
   new_spsi_e = sin(new_psi_errorCTE(ii)+new_Beta);

   % Distance to the ith way point projected to the track line S(t)i
   % - Eq (14)

   new_s(ii) = [New_X_Way_Error(ii),New_Y_Way_Error(ii)]*[(...
         New_X_Way_c(jj)-New_PrevX_Way_c(jj)),(New_Y_Way_c(jj)...
```

```
      -New_PrevY_Way_c(jj))]';
%*****************************************************************
% Calculates the Overall Distance Travelled, Distance Travelled,
% Time Used Overall, Time Remaining, Switches velocity from 0.5
% knots to 3.5 knots depending on Time Remaining and Distance
% Remaining while taking into account acceleration/deceleration.
% (28 Feb 02)

if ii == 1,
   overall_distance_travelled(ii) = 0;
   distance_travelled(ii) = 0;
   time_used(ii) = 0;
   time_remaining(ii) = new_time(ii);
   real_time(ii) = 0;
elseif ii == 2,
   overall_distance_travelled(ii) = New_SegLen(ii) - new_s(ii-1);
   distance_travelled(ii) = overall_distance_travelled(ii);
   %time_used(ii) = distance_travelled(ii)/(new_U(ii)/3.28);
   time_remaining(ii) = time_remaining(ii-1)-dt;%time_used(ii);
   real_time(ii) = new_time(ii) - time_remaining(ii);
else
   overall_distance_travelled(ii) = New_SegLen(ii) - new_s(ii-1);
   distance_travelled(ii) = overall_distance_travelled(ii) -...
      overall_distance_travelled(ii-1);
   time_used(ii) = distance_travelled(ii)/(new_U(ii)/3.28);
   time_remaining(ii) = time_remaining(ii-1) - dt;%time_used(ii);
   real_time(ii) = new_time(ii) - time_remaining(ii);
end

% s is distance to go projected to track line(goes from 0-100%L)
% - Eq (14)

new_s(ii) = new_s(ii)/New_SegLen(jj);

if (time_remaining(ii)<0.125), disp('mission out of time'),...
      break,end;

% Determines what speed to set the vehicle at
%time_available(ii) = new_s(ii)/(new_U(ii)/3.28);
ucom(ii)=new_s(ii)/time_remaining(ii);

if ucom(ii) > 1.8 ;
   ucom(ii) = 1.8 ;        % Max velocity is 3.5 Knots
end
if ucom(ii) < 0.2571 ;
   ucom(ii) = 0.2571 ;     % Min velocity is 0.5 Knots
end



%*****************************************************************
%*****************************************************************
% New Control Law added for Longitudinal Equation of Motion -
% 14 May 2002

new_sigma(ii) = new_U(ii)-ucom(ii);
```

```
new_phi(ii) = 0.1;tau=0.1;ncommax=22;

ncom2(ii) = (1.39*(accel(ii)*0-800*tanh(new_sigma(ii)/...
   new_phi(ii))+67.74*(new_U(ii)*abs(new_U(ii)))));
ncom(ii)=sqrt(abs(ncom2(ii)))*sign(ncom2(ii));

% Limits propeller speed to 22 rps

if (abs(ncom(ii))>ncommax),
   ncom(ii)=ncommax*sign(ncom(ii));
end;

% always +ve in these simulations
% solve longitudinal dynamics

new_U(ii+1) = (0.004641*dt*(-10.5*(new_U(ii)*abs(new_U(ii)))+...
   0.155*(ncom(ii)*abs(ncom(ii))) - 0.05*ncom(ii)*new_U(ii)))...
   +new_U(ii);

New_SegLen(ii+1) = New_SegLen(ii);
new_time(ii+1) = new_time(ii);
accel(ii+1) = accel(ii);

%*****************************************************************
%*****************************************************************

% Ranges from 0-100% of SegLen

Ratio=(1.0-new_s(ii)/New_SegLen(jj))*100.0;

% Radial distance to go to next WP

new_ss(ii) = sqrt(New_X_Way_Error(ii)^2 + New_Y_Way_Error(ii)^2);

% dp is angle between line of sight and current track line
% - Eq (16)

new_dp(ii) = atan2( (New_Y_Way_c(jj)-New_PrevY_Way_c(jj)),(...
   New_X_Way_c(jj)-New_PrevX_Way_c(jj)) )- atan2...
   (New_Y_Way_Error(ii),New_X_Way_Error(ii) );
if(new_dp(ii) > pi),
   new_dp(ii) = new_dp(ii) - 2.0*pi;
end;

% Cross Track Error Definition - Eq (15)

new_cte(ii) = new_s(ii)*sin(new_dp(ii));

% If the magnitude of the CTE Heading exceeds 40 degrees, a
% LOS Controller is used.

if( abs(new_psi_errorCTE(ii)) >= 40.0*pi/180.0 | new_s(ii) < 0.0 ),
   new_LOS(ii) = 1;
   new_psi_comLOS = atan2(New_Y_Way_Error(ii),...
      New_X_Way_Error(ii));
   new_psi_errorLOS(ii) = new_psi_comLOS - new_psi_cont(ii);
```

```
    if(abs(new_psi_errorLOS(ii)) > pi),
       new_psi_errorLOS(ii) = new_psi_errorLOS(ii) - 2.0*pi*...
          new_psi_errorLOS(ii)/abs(new_psi_errorLOS(ii));
    end;

    % Eq (8)

    new_Sigma_FlightHeading = 0.9499*(new_r_com - new_r(ii)) +...
       0.1701*new_psi_errorLOS(ii);

    % Eq (9)

    new_dr(ii) = -1.5435*( 2.5394*new_r(ii)+ Eta_FlightHeading*...
       tanh(new_Sigma_FlightHeading/Phi_FlightHeading));

else

    % Use CTE Controller if CTE Heading is less than 40 degrees

    new_LOS(ii) = 0;
    if(new_cpsi_e ~= 0.0),              % Trap Div. by Zero !

       % SMC Soln

       % Sliding Surface - Eq (20)

       new_Sigma(ii) = new_U(ii)*3.28*new_rRM(ii)*new_cpsi_e...
          + Lam1*new_U(ii)*3.28*new_spsi_e + 3.28*Lam2*...
          new_cte(ii);

       % Rudder Input - Eq (21)

       % new_dr(ii) = (1.0/(new_U(ii)*b*new_cpsi_e))*...
       %     (-new_U(ii)*a*new_rRM(ii)*new_cpsi_e + new_U(ii)*...
       %     new_rRM(ii)^2*new_spsi_e - Lam1*new_U(ii)*...
       %     new_rRM(ii)*new_cpsi_e - Lam2*new_U(ii)*...
       %     new_spsi_e -new_Eta_CTE*(new_Sigma(ii)/new_Phi_CTE));
       % changes because new_U is now in m/sec.

       new_dr(ii) = (1.0/(new_U(ii)*3.28*b*new_cpsi_e))*...
          (-new_U(ii)*3.28*a*new_rRM(ii)*new_cpsi_e +...
          new_U(ii)*3.28*new_rRM(ii)^2*new_spsi_e - ...
          Lam1*new_U(ii)*3.28*new_rRM(ii)*new_cpsi_e - ...
          Lam2*new_U(ii)*3.28*new_spsi_e -new_Eta_CTE*...
          (new_Sigma(ii)/new_Phi_CTE));

    else
       new_dr(ii) = new_dr(ii-1);
    end;
end;                          % End of CTE Controller

% Use LOS if near to loiter point
% if (loiter==1)& new_s(ii)<10;  new_dr(ii)=new_drlos(ii);end;

% Surface Phase Logic (Independent of LOS or CTE)
```

```
if(SurfPhase == TRUE)
   if(new_SURFACE_TIMER_ACTIVE == FALSE)
      if(Ratio > 40.0)
         % Start a Timer
         new_SURFACE_TIMER_ACTIVE = TRUE;
         new_Depth_com(ii) = 0.0;
         new_SurfaceWait = new_SurfaceTime(1) + tt(ii);
         new_SurfaceWait
      end;
   end;
end;
if(new_SURFACE_TIMER_ACTIVE == TRUE)
   if(tt(ii) >= new_SurfaceWait)
      new_SURFACE_TIMER_ACTIVE = FALSE;
      new_Depth_com(ii) = new_WayPointVertDist_com(1);
      new_SurfPhase(1) = 0;
   else
      new_Depth_com(ii) = 0.0;
   end;
end;
if(abs(new_dr(ii)) > 0.4)
   new_dr(ii) = 0.4*sign(new_dr(ii));
end;

% Model drl is the actual lagged rudder, dr is the rudder command.
% taudr = 0.255;

%  drl(i+1) = drl(i) + dt*(dr(i)-drl(i))/taudr;
%   if(abs(drl(i)) > 0.4)
%                drl(i) = 0.4*sign(drl(i));
%   end;
new_x_dot(:,ii+1) = [ A(1,1)*new_v(ii) + A(1,2)*new_r(ii) + B(1)*...
      new_dr(ii); A(2,1)*new_v(ii) + A(2,2)*new_r(ii) + B(2)*...
      new_dr(ii); new_r(ii)];
new_x(:,ii+1) = new_x(:,ii)+dt*new_x_dot(:,ii);
new_v(ii+1)   = new_x(1,ii+1)/12;
new_r(ii+1)   = new_x(2,ii+1);
new_psi(ii+1) = new_x(3,ii+1);
new_rRM(ii+1) = new_r(ii+1);

% Added
% rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*dr(i));
% psi(i+1) = psi(i) + dt*rRM(i);

% Throw in some Waves
% Uc(i) = -0.5*sin(2*pi*t(i)/5);
% Vc(i) =  0.5*sin(2*pi*t(i)/5);

% Model using system ID results from Bay tests

% rRM(i+1) = rRM(i) + dt*(a*rRM(i) + b*drl(i));
% psi(i+1) = psi(i) + dt*rRM(i);
% side slip added proprtional to turn rate from AZORES data V in
% ft/sec
% v(i+1) = 1.0*rRM(i+1)*3.28;
```

```
      Uc = 0.0;
      Vc = 0.0;

      % Kinematics note new_U is in meters /sec, new_v is in ft/sec
      % hold over from long time ago

      New_X(ii+1) = New_X(ii) + (Uc + (new_U(ii))*cos(new_psi(ii)) -...
         new_v(ii)/3.28*sin(new_psi(ii)) )*dt;
      New_Y(ii+1) = New_Y(ii) + (Vc + (new_U(ii))*sin(new_psi(ii))...
         + new_v(ii)/3.28*cos(new_psi(ii)) )*dt;

      % Check to See if we are Within the Watch_Radius (set to 1
      % meter here)

      if(sqrt(New_X_Way_Error(ii)^2.0 + New_Y_Way_Error(ii)^2.0)...
            <= 1 | new_s(ii) < 0.0),

         % Next Line ends mission if within Watch Radius.

         disp(sprintf('Rendezvous Point Reached')); break;
         if(jj==No_tracks),
            PLOT_PART = 2;
            disp(sprintf('PLOT_PART = 2'));
            break;
         end;
         New_PrevX_Way_c(jj+1) = New_X_Way_c(jj);
         New_PrevY_Way_c(jj+1) = New_Y_Way_c(jj);
      end
   end  %end of ii loop
end  % if j=No_Tracks
%*************************************************************************

% Plotting

if PLOT_PART == 1,

   % Plot of Time vs Rudder Angle & Vehicle Heading

   figure(1); clf
   orient tall
   plot(t([1:i+1]),psi*180/pi);
   hold;
   plot(t([1:i+1]),dr*180/pi,'r:');grid;
   title('Time vs Rudder Angle and Vehicle Heading');
   xlabel('Time (sec)'); ylabel('Rudder Angle/Vehicle Heading (degrees)');
   legend('Vehicle Heading', 'Rudder Angle');
   print -tiff -depsc figure1_wp5
   hold;zoom on;

   % Plot of Time vs CTE, Distance to Go to Projected Track, Radial
   % Distance to Next Waypoint

   figure(2); clf
   orient tall
   plot(t([1:i+1]),cte);
   hold;
```

```
    plot(t([1:i+1]),s,'r:');
    plot(t([1:i+1]),ss,'g--');grid;
    title('Time vs Cross Track Error')
    xlabel('Time (sec)');ylabel('Distance (meters)');
    legend('Cross Track Error', 'Distance to Go Projected to Track', ...
       'Radial Distance to Go to Next Way Point');
    print -tiff -depsc figure2_wp5
    hold;zoom on;

    %***************
    figure(3); clf
    orient tall
    plot(t([1:i+1]),U); grid;
    title('Time vs Forward Speed')
    xlabel('Time (sec)'); ylabel('Forward Speed (m/s)');
    %***************

elseif PLOT_PART == 0,

    % Plot of Time vs Rudder Angle & Vehicle Heading for Rendezvous Mission

    figure(4); clf
    orient tall
    plot(t([1:i+1]),psi*180/pi, 'g');
    hold;
    plot(real_time([1:ii])+30,new_psi([1:ii])*180/pi, 'g:');
    plot(t([1:i]),dr*180/pi,'r');
    plot(real_time([1:ii-1])+30,new_dr([1:ii-1])*180/pi, 'r:');
    title('Time vs Rudder Angle and Vehicle Heading');
    xlabel('Time (sec)'); ylabel('Rudder Angle/Vehicle Heading (degrees)');
    legend('Vehicle Heading Before Mission Abort',...
       'Vehicle Heading After Mission Abort',...
       'Rudder Angle Before Mission Abort',...
       'Rudder Angle After Mission Abort');
    print -tiff -depsc figure4_wp5
    hold;grid;

    % Plot of Time vs CTE, Distance to Go to Projected Track, Radial
    % Distance to Next Waypoint for Rendezvous Mission

    figure(5); clf
    orient tall
    plot(t([1:i]), cte, 'g');
    hold;
    plot(real_time([1:ii-1])+30, new_cte([1:ii-1]), 'g:');
    plot(t([1:i]), s,'r');
    plot(real_time([1:ii-1])+30, new_s([1:ii-1]), 'r:');
    plot(t([1:i-1]), ss([1:i-1]),'b');grid;
    plot(real_time([1:ii-1])+30, new_ss([1:ii-1]), 'b:');
    title('Time vs Cross Track Error')
    xlabel('Time (sec)'); ylabel('Distance (meters)');
    legend('Cross Track Error Before Abort',...
       'Cross Track Error After Abort',...
       'Distance to Go Projected to Track Before Abort', ...
       'Distance to Go Projected to Track After Abort',...
       'Radial Distance to Go to Next Way Point Before Abort',...
```

```
      'Radial Distance to Go to Next Way Point After Abort');
   print -tiff -depsc figure5_wp5
   hold;zoom on;
   %***************
   figure(6); clf
   orient tall
%     subplot(1,2,1) % In order to produce Figure 17 in thesis
   plot(t([1:i+1]),U/3.28, 'b*');
   hold;
   plot(real_time([1:ii-1])+30,new_U([1:ii-1]),'r.',...
      real_time([1:ii-1])+30,ucom([1:ii-1]),'m'); grid;
   %    AXIS([0 30+new_time(1) 0 2]);
   title('Time vs Forward Speed')
   xlabel('Time (sec)'); ylabel('Forward Speed (m/s)');
   legend('Original Speed', 'Rendezvous Speed', 'Command Speed');
   hold;
%  In order to produce Figure 17 in thesis
%     subplot(1,2,2)
%     plot(real_time(1:ii)+30, ncom(1:ii),'b.'); grid
%     title('Propeller Speed')
%     xlabel('Time (sec)'); ylabel('Propeller Speed (rps)')
%     axis([30 100 12 22])
   print -tiff -depsc figure6_wp5

   % 3-D Plot

   figure(7); clf
   orient tall
   plot3(Y, X, t([1:i+1]), 'b'); grid;
   title('Time Space Plot')
   xlabel('Y (meters)');
   ylabel('X (meters)');
   zlabel('Time (seconds)');
   hold;
   plot3(New_Y([1:ii]), New_X([1:ii]), real_time([1:ii])+30, 'rx');
   plot3(New_Y_Way_c, New_X_Way_c, new_time(ii)+30, 'gd');
   legend('Original Track', 'New Track', 'Rendezvous')
   print -tiff -depsc figure7_wp5
   hold;

   %    figure(8); clf
   %    orient tall
   %    plot(t([1:i+1]),X, 'r');
   %    hold;
   %    plot(real_time([1:ii])+30, New_X([1:ii]),'r:');
   %    plot(t([1:i+1]), Y, 'g');
   %    plot(real_time([1:ii])+30, New_Y([1:ii]),'g:'); grid;
   %    title('Time vs Postition');
   %    xlabel('Time (sec)'); ylabel('Position (meters)');
   %    legend('Original X', 'Rendezvous X', 'Original Y',...
   %       'Rendezvous Y',2);
   %    hold;

   %***************
end;
```

```
% Plot of Actual Track and Planned Track
% Modified on 05 Feb 2002 - To include waypoint2.m modifications.

if PLOT_PART == 1,
   figure(9); clf
   orient tall
   plot(Y,X,'b--');grid;                          % Actual Track
   title('ARIES Track - Actual and Planned');
   xlabel('Y (meters)');ylabel('X (meters)');
   hold;

   % Planned Track
   plot([Y_Way_c(1) PrevY_Way_c(1)],[X_Way_c(1) PrevX_Way_c(1)],'r');
   plot(Y_REMUS, X_REMUS, 'g');
   axis([-100 220 -120 60]);
   for ik=2:No_tracks,
      plot([Y_Way_c(ik) Y_Way_c(ik-1)],[X_Way_c(ik)...
            X_Way_c(ik-1)],'r');
   end;
   legend('Actual Track - ARIES', 'Planned Track - ARIES',...
      'REMUS Path',4);
   print -tiff -depsc figure9_wp5
   hold; zoom on

elseif PLOT_PART == 0 | PLOT_PART == 2,

   % Plot of Planned Track, Track after Mission Change, Rendezvous
   % Point and Initial Track

   figure(10); clf
   orient tall
   plot(Y,X,'b--');grid;
   title('ARIES Track - Actual and Planned');
   xlabel('Y (meters)');ylabel('X (meters)');
   hold;
   plot(New_Y, New_X,'g-.');
   plot(New_Y_Way_c, New_X_Way_c,'gd');
   plot(Y_REMUS, X_REMUS, 'm');
   plot([Y_Way_c(1) PrevY_Way_c(1)],[X_Way_c(1) PrevX_Way_c(1)],'r');
   for ik=2:No_tracks,
      plot([Y_Way_c(ik) Y_Way_c(ik-1)],[X_Way_c(ik)...
            X_Way_c(ik-1)],'r');
   end;
   legend('Initial Track - ARIES',...
      'Track After Modem Command - ARIES','Rendezvous Point',...
      'REMUS Track','Planned Track - ARIES',4);
   axis([-100 220 -120 60]);

   %      AXIS([-100 250 -80 60])
   print -tiff -depsc figure10_wp5
   hold; zoom on
end
```

LIST OF REFERENCES

Cox, I.J. and G.T. Wilfong (ed.), *Autonomous Robot Vehicles*, Springer-Verlag, New York, 1990.

Faiz, Nadeem and Sunil Agrawal, "Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 2, pp. 219-227, Mar-Apr 2001.

Fourquet, Jean-Yves and Marc Renaud, "Time-Optimal Motions for a Torque Controlled Wheeled Mobile Robot Along Specified Paths," *Proceedings of the 35th Conference on Decision and Control*, pp. 3587-3592, Dec 1996.

Fraichard, Th., "Dynamic Trajectory Planning with Dynamic Constraints: a 'State-Time Space' Approach," *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1393-1400, Jul 1993.

Fraichard, Th and A. Scheuer, "Car-Like Robots and Moving Obstacles," *1994 IEEE International Conference on Robotics and Automation*, pp. 64-69, 1998.

Gelb, A., [and others], *Applied Optimal Estimation*, MIT Press, Cambridge, MA, 1974.

Hanselman, Duane and Bruce Littlefield, *Mastering MATLAB*, Prentice-Hall, Upper Saddle River, NJ, 1996.

Healey, A.J., *Dynamics of Marine Vehicles (ME-4823)*, Class Notes, Naval Postgraduate School, Monterey, CA, 1995.

Healey, A. J., "Command and Control Demonstrations with Cooperating Vehicles," *ONR Research Proposal in response to ONR BAA 01-012 "Demonstration of Undersea Autonomous Operation Capabilities and Related Technology Development"*, August 2001.

Healey, A.J. and David Lienard, "Multivariable Sliding Mode Control for Autonomous Diving and Steering of Unmanned Underwater Vehicles," *IEEE Journal of Oceanic Engineering*, vol. 18, pp 327-339, Jul 1993.

Johnson, Jay, "Parameter Identification of the ARIES AUV," M.S. Thesis Naval Postgraduate School, Monterey, CA, June 2001.

Kanayama, Yutaka and Bruce I. Hartman, "Smooth Local Path Planning for Autonomous Vehicles," *Autonomous Robot Vehicles*, Springer-Verlag, New York, 1990, pp. 62-67.

Kant, Kamal and Steven Zucker, "Toward Efficient Trajectory Planning: The Path-Velocity Decomposition," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72-89, Fall 1986.

Lamb, H., Sir, *Hydrodynamics by Sir Horace Lamb*, New York: Dover Publications, 1945.

Lewis, E.V. (ed.), *Principles of Naval Architecture*, vol. II, second revision, Society of Naval Architects and Marine Engineers (SNAME), Jersey City, NJ, 1988.

Lienard, David, "Autopilot Design for Autonomous Underwater Vehicles Based on Sliding Mode Control," M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1990.

Lopez, Ismael and Colin McInnes, "Autonomous Rendezvous Using Artificial Potential Function Guidance," *Journal of Guidance, Control and Dynamics*, vol. 18, no. 2, pp. 237-241, Mar-Apr 1995.

Marco, D.B., "Procedure to Run Missions with the ARIES," Personal Notes, 2001.

Marco, D.B. and A.J. Healey, "Current Developments in Underwater Vehicle Control and Navigation," *Proceedings of IEEE Oceans*, 2000.

Marco, D.B and A.J. Healey, "Command, Control and Navigation Experimental Results With the NPS ARIES AUV," *IEEE Journal of Oceanic Engineering – Special Issue*, 2001.

Munoz, V.F. [and others], "Speed Planning Method for Mobile Robots Under Motion Constraints," *IFAC Intelligent Autonomous Vehicles*, pp. 123-128, 1998.

Munoz, V.F. and A. Garcia-Cerezo, "Speed Planning and Generation Approach Based on the Path-Time Space for Mobile Robots," *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pp. 2199-2204, May 1998.

Nelson, W.L. and I.J. Cox, "Local Path Control for an Autonomous Vehicle," *Autonomous Robot Vehicles*, Springer-Verlag, New York, 1990, pp. 38-44.

Reidel, Jeffrey, "Seaway Learning and Motion Compensation in Shallow Waters for Small AUVs," Doctoral Dissertation, Naval Postgraduate School, Monterey, CA, June 1999.

Sabin, William A., *The Gregg Reference Manual*, McGraw-Hill, New York, 1992.

Sasiadek, Jerzy and Igancy Duleba, "3D Local Trajectory Planner," *AIAA Guidance, Navigation and Control Conference Collection of Technical Papers*, pp. 517-526, 1997.

Shiller, Z. and W. Serate, "Trajectory Planning of Tracked Vehicles," *Journal of Dynamic Systems, Measurement, and Control*, vol 117, pp 619-624, Dec 1995.

Tecnadyne Corporation, *Model 520 Dimensions, Configuration and Specifications*, www.tecnadyne.com/thrusters.htm.

Wang, Danwei and Feng Qi, "Trajectory Planning for a Four-Wheel-Steering Vehicle," *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, pp. 3320-3325, May 2001.

Weiguo, Wang, Chen Huitang and Woo Peng-Yung, "Optimal Motion Planning for a Wheeled Mobile Robot," *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pp. 41-46, May 1999.

White, F.M., 'Fluid Mechanics," McGraw-Hill, Boston, MA, 1999.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Mechanical Engineering Department Chairman, Code ME
   Naval Postgraduate School
   Monterey, CA

4. Naval/Mechanical Engineering Curriculum Code 34
   Naval Postgraduate School
   Monterey, CA

5. Professor Anthony J. Healey, Code ME/HY
   Department of Mechanical Engineering
   Naval Postgraduate School
   Monterey, CA

6. Dr. Donald Brutzman, Code UW/Br
   Undersea Warfare Group
   Naval Postgraduate School
   Monterey, CA

7. Dr. T. B. Curtin, Code 322OM
   Office of Naval Research
   Arlington, VA

8. Dr. T. Swean, Code 32OE
   Office of Naval Research
   Arlington, VA

9. LCDR John J. Keegan
   Supervisor of Shipbuilding
   Pascagoula, MS

10. Mr. John J. Keegan III
    Chesapeake, VA

11. LT Joe Keller
    Naval Postgraduate School
    Monterey, CA

12. LT Lynn Fodrea
Naval Postgraduate School
Monterey, CA

13. CDR A.B. Fuller, USN (Ret.)
Washington, DC

14. CDR T.M. Negus
Washington, DC