NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

HORIZONTAL STEERING CONTROL IN DOCKING THE ARIES AUV

by

Tan Wee Kiat

December 2003

Thesis Co-Advisors: Fotis Papoulias
Anthony J. Healey

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 2003	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE: Horizontal Steering Control in Docking the ARIES AUV

5. FUNDING NUMBERS

6. AUTHOR(S) Tan Wee Kiat

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)
N/A

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 (maximum 200 words)

To keep the operational cost down and increase the mission time with minimum human intervention, autonomous recovery or docking operation of an Autonomous Underwater Vehicle (AUV) is required. Central to the successful autonomous docking process of the AUV is the capability of the AUV being able to track and steer itself accurately towards the dock which is constantly perturbed by wave motion effects. In addition, for accurate acoustic homing during the final stages of the docking, the AUV requires acoustic systems with high update rates. Equipped with acoustic modem, ARIES had experimentally been tested to have an update rate of only about 0.3 Hz. These delayed data can potentially cause a false commanded reference input to the tracking system in between the updates and cause ARIES to miss the moving cage's entrance. This thesis attempts to investigate the effectiveness on the use of cross track error and line of sight error sliding mode controller coupled with dynamic waypoints allocation in horizontal steering of ARIES in docking operations. In the absence of cage heading updates, a predictive method based on angular rate and direction of motion was used to estimate the dynamics of the moving cage. Further analysis was performed in order to understand the limitations of such an implementation.

14. SUBJECT TERMS
Sliding Mode Control, Waypoint Navigation, Docking of AUV

15. NUMBER OF PAGES 101

16. PRICE CODE

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

HORIZONTAL STEERING CONTROL IN DOCKING THE ARIES AUV

Wee Kiat Tan
Civilian, Singapore Ministry of Defense
B.Eng., Nanyang Technological University, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2003

Author: Wee Kiat Tan

Approved by: Papoulias Fotis
Thesis Co-Advisor

Anthony J. Healey
Thesis Co-Advisor

Anthony J. Healey
Chairman, Department of Mechanical and Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

To keep the operational cost down and increase the mission time with minimum human intervention, autonomous recovery or docking operation of an Autonomous Underwater Vehicle (AUV) is required. Central to the successful autonomous docking process of the AUV is the capability of the AUV being able to track and steer itself accurately towards the dock which is constantly perturbed by wave motion effects. In addition, for accurate acoustic homing during the final stages of the docking, the AUV requires acoustic systems with high update rates. Equipped with acoustic modem, ARIES had experimentally been tested to have an update rate of only about 0.3 Hz. These delayed data can potentially cause a false commanded reference input to the tracking system in between the updates and cause ARIES to miss the moving cage's entrance.

This thesis attempts to investigate the effectiveness on the use of cross track error and line of sight error sliding mode controller coupled with dynamic waypoints allocation in horizontal steering of ARIES in docking operations. In the absence of cage heading updates, a predictive method based on angular rate and direction of motion was used to estimate the dynamics of the moving cage. Further analysis was performed in order to understand the limitations of such an implementation.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge Professor Fotis Papoulias and Professor Tony Healey for their guidance, patience and motivation throughout the thesis process. Their high level of technical competence and uncanny ability to teach provided me with a great understanding of the subject area. Finally and most importantly I would like to thank my wife, Eileen Chong, for her love and support.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

A docking system is a crucial component in an Autonomous Underwater Vehicle (AUV) system since it greatly enhances AUV mission time and reduces its operation cost. It's primary purpose is to perform the tasks of deploying and recovering, downloading mission data, uploading new instructions to the vehicle, real-time vehicle diagnostics and recharging of vehicle batteries. With these capabilities, the docking system can greatly increase the range of potential applications possible with an AUV and further enhance its autonomy and operation time in its missions. It is also known that a large part of operational cost of AUV is attributed to the ship and personnel time required to get the AUV on station to begin a deployment, and the AUV recovery time at the end of a mission. By automating the docking process, these overheads can be greatly reduced making the AUV operationally more viable.

Central to the successful docking process of the AUV is the capability of the vehicle to track and steer itself accurately towards the dock that is constantly perturbed by wave motion effects. The type of docking system employed also dictates the complexity of the docking process. These can be broadly classified into two categories [1]:

- Those that equipped the AUV with special mounting hardware to grab a vertical pole or line
- Tethered cage with protective housing and it requires the AUV to swim in

The latter approach is the area of interest in this thesis. It presents a more challenging navigation problem than the former, as the AUV must orient itself to be aligned with the cage entrance. The cage platform that is tethered to a ship at some depth below the surface of the water has its dynamics greatly influenced by wave motion effects of the ship and ocean currents. These perturbations to the cage heading and position may easily move it away from the approaching path of the AUV. In order for the AUV to guide itself to the entrance of the cage, some a

priori knowledge of which way the cage is facing is required. This requires some form of a heading measuring device such as an electronic compass on the dock and an acoustic modem that can transmit the information to the AUV with the shortest possible delay. With the cage orientation known, the AUV will need to have the capability to compensate for the cage's motion when making its final approach to the cage in order for it to dock successfully.

B. SCOPE OF THIS WORK

The docking problem can be decoupled and linearized into a set of dynamics problems in the vertical and horizontal plane. The main area of interest in this study is in steering control of AUV to a swinging cage in the x-y plane. The NPS AUV platform, ARIES is used for the study.

The focus of this thesis is two-fold:

1. Develop a set of trajectory paths for approaching the docking cage and investigate its performance under fix and moving cage's headings.

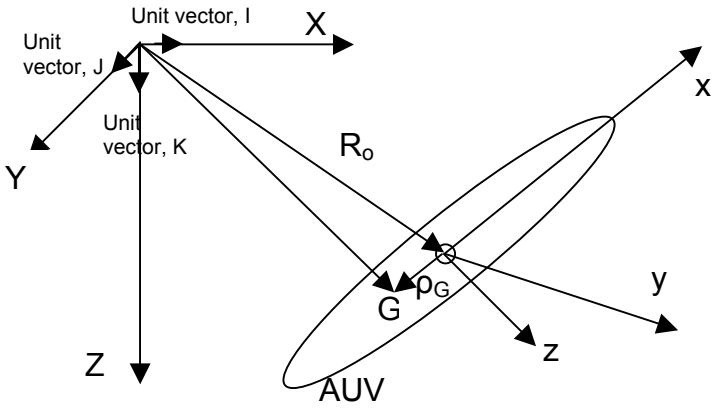
2. Investigate the impact of time delay in data update caused by the acoustic modem and further design a method to account for this time delay.

Chapter II will focus on the equations of motion for ARIES AUV and the associated steering control laws. Chapter III will discuss trajectory design for docking the AUV and its implementation. Chapter IV will present the simulation results and their limitations. Chapter V will discuss the theory and design of an estimator that can observe the extra state of the cage heading and present the simulation results. Finally, Chapter VI will offer some conclusions and recommendations for future studies.

II. EQUATIONS OF MOTION AND AUV MODELING

A. SYSTEM OF EQUATIONS OF RIGID BODY MOTION

This section presents a brief description of derivation of equations of motion for underwater rigid body which is essential for AUV modeling. For a start, the two different coordinate systems, global and body reference frame and their relationship are defined in Figure 1.



Coordinate System and Positional Definitions

According to Healy [2], the global navigation frame OXYZ assumes a NED coordinate convention and by ignoring the earth's rotational rate, it is also the "inertial" reference frame in which Newton's Laws of Motion will be valid. The vehicle body fixed reference frame is defined with Y axis pointing to the right, X axis along the longitudinal axis of the vehicle and the Z axis is positive downwards. The origin of the reference frame lies at the vehicle's center which is different from the vehicle's center of mass (G).

It follows that the vehicle position with respect to the global origin, R_o is defined as $R_o = [X_0 I + Y_o J + Z_o K]$ or in vector form, $R_o = [X_0, Y_o, Z_o]'$. Thus the global velocity

$$\frac{dR_o}{dt} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = T^{-1}(\phi,\theta,\psi) \bullet \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{1}$$

Where:

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

u, v, w = Vehicle's forward speed (surge), side slip velocity (sway) and component velocities in local Z direction (heave) respectively.

ϕ, θ, ψ = Euler angles

The rate of change of Euler angles can in turn be found through the following relationship:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2}$$

Where:

p, q, r = body fixed components of the angular velocity vector.

The inertial frame position of the vehicle's center of mass can be further expressed as $R_G = R_o + \rho_G$. Since the center of mass lies in a body that is translating and rotating, the total time derivative of R_G becomes

$$\frac{dR_G}{dt} = \dot{R}_o + \omega \times \rho_G = v + \omega \times \rho_G \tag{3}$$

Where:

$$\omega = [p, q, r]'$$

$$\rho_G = [x_G, y_G, z_G]' - [X_0, Y_o, Z_o]'$$

The translational Equation of Motion can then be expressed as

$$\begin{aligned} F &= m\frac{d^2R_G}{dt^2} \\ &= m\left(\dot{v}+\dot{\omega}\times\rho_G+\omega\times\omega\times\rho_G+\omega\times v\right) \end{aligned} \tag{4}$$

and the rotational Equation of Motion as

$$M_o = \frac{dH_o}{dt}+\rho_G\times\left(m\frac{d^2R_G}{dt^2}\right) \text{ or } M_o = I_o\dot{\omega}+\omega\times\left(I_o\omega\right)+m\left(\rho_G\times\dot{v}+\rho_G\times\omega\times v\right) \tag{5}$$

Where:

Angular momentum of body, $H_o = I_o\omega$

$$\frac{dH_o}{dt} = I_o\dot{\omega}+\omega\times H_o$$

Mass moment of inertia,

$$I_o = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} dm_i\left(y^2+z^2\right) & -\sum_{i=1}^{N} dm_i\left(xy\right) & -\sum_{i=1}^{N} dm_i\left(xz\right) \\ -\sum_{i=1}^{N} dm_i\left(xy\right) & \sum_{i=1}^{N} dm_i\left(x^2+z^2\right) & -\sum_{i=1}^{N} dm_i\left(yz\right) \\ -\sum_{i=1}^{N} dm_i\left(xz\right) & -\sum_{i=1}^{N} dm_i\left(yz\right) & \sum_{i=1}^{N} dm_i\left(x^2+y^2\right) \end{bmatrix} \tag{6}$$

To incorporating the weight and buoyancy forces into the equation of motion, the force and moment vector written in body coordinates as

$$F_g = \begin{bmatrix} f_g \\ m_g \end{bmatrix} = \begin{bmatrix} \left(W-B\right)\begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} \\ W\rho_G\times\begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} - B\rho_G\times\begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} \end{bmatrix} \tag{7}$$

can be added negatively to the right hand side of equation (4) and (5) resulting in equation (8) and (9) which form the six degrees of freedom equations of motion.

$$m\left(\frac{dv}{dt}+\dot{\omega}\times\rho_G\right)+m\left(\omega\times\omega\times\rho_G+\omega\times v\right)-f_g=\begin{bmatrix}X_f\\Y_f\\Z_f\end{bmatrix} \tag{8}$$

$$I_o\dot{\omega}+\omega\times\left(I_o\omega\right)+m\left(\rho_G\times\dot{v}+\rho_G\times\omega\times v\right)-m_g=\begin{bmatrix}K_f\\M_f\\N_f\end{bmatrix} \tag{9}$$

Further modifications can be done to account for ocean current by substituting the state vector of $[u \quad v \quad w \quad p \quad q \quad r]$ with relative velocities of $[u_r \quad v_r \quad w_r \quad p \quad q \quad r]$ and equation (1) will be modified to

$$\frac{dR_o}{dt}=\begin{bmatrix}\dot{X}\\\dot{Y}\\\dot{Z}\end{bmatrix}=T^{-1}\left(\phi,\theta,\psi\right)\bullet\begin{bmatrix}u_r\\v_r\\w_r\end{bmatrix}+\begin{bmatrix}U_{cx}\\U_{cy}\\U_{cz}\end{bmatrix} \tag{10}$$

Where:

U_{cx}, U_{cy}, U_{cz} = Ocean current velocities

and the equations of six degree of freedom motion will be expressed as:

SURGE EQUATION OF MOTION

$$m[\dot{u}_r-v_r r+w_r q-x_G\left(q^2+r^2\right)+y_G\left(pq-\dot{r}\right)+z_G\left(pr+\dot{q}\right)]$$

$$+\left(W-B\right)\sin\theta=X_f \tag{11}$$

SWAY EQUATION OF MOTION

$$m[\dot{v}_r+u_r r-w_r p+x_G\left(pq+\dot{r}\right)-y_G\left(p^2+r^2\right)+z_G\left(qr-\dot{p}\right)]$$

$$-\left(W-B\right)\cos\theta\sin\phi=Y_f \tag{12}$$

HEAVE EQUATION OF MOTION

$$m[\dot{w}_r - u_r q + v_r p + x_G(pr - \dot{q}) + y_G(qr + \dot{p}) - z_G(p^2 + q^2)]$$
$$+(W - B)\cos\theta\cos\phi = Z_f \quad (13)$$

ROLL EQUATION OF MOTION

$$I_x\dot{p} + (I_z - I_y)qr + I_{xy}(pr - \dot{q}) - I_{yz}(q^2 - r^2) - I_{xz}(pq + \dot{r}) + m[y_G(\dot{w} - u_r q + v_r p)$$
$$-z_G(\dot{v}_r + u_r r - w_r p)] - (y_G W - y_B B)\cos\theta\cos\phi + (z_G W - z_B B)\cos\theta\sin\phi = K_f \quad (14)$$

PITCH EQUATION OF MOTION

$$I_y\dot{q} + (I_z - I_z)pr - I_{xy}(qr + \dot{p}) + I_{yz}(pq - \dot{r}) + I_{xz}(p^2 - r^2) - m[x_G(\dot{w} - u_r q + v_r p)$$
$$-z_G(\dot{u}_r - v_r r + w_r q)] + (x_G W - x_B B)\cos\theta\cos\phi + (z_G W - z_B B)\sin\theta = M_f \quad (15)$$

YAW EQUATION OF MOTION

$$I_z\dot{r} + (I_y - I_x)pq - I_{xy}(p^2 - q^2) - I_{yz}(pr + \dot{q}) + I_{xz}(qr - \dot{p}) + m[x_G(\dot{v}_r + u_r r - w_r p)$$
$$-y_G(\dot{u}_r - v_r r + w_r q)] - (x_G W - x_B B)\cos\theta\sin\phi - (y_G W - y_B B)\sin\theta = N_f \quad (16)$$

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 AUV and center of buoyancy

x_G, y_G, z_G = Position difference between geometric center of AUV and center of gravity

X_f, Y_f, Z_f, K_f, M_f, N_f = Sums of all external forces acting on AUV in the particular body fixed direction

B. STEERING SYSTEM MODEL

By assuming symmetry of the AUV, we may simplify the six degree of freedom motion equations to horizontal plane maneuvering of the AUV and diving behavior in the vertical plane where the sway, yaw and roll are coupled but independent of heave, pitch and surge motions. Since the area of interest in this study is in steering control of AUV, the steering system model will be discussed in more details in this section.

Making further assumption that the center of mass of the vehicle lies below the origin (z_G is positive) while x_G and y_G are zero, motions in the vertical are negligible (i.e. [w_r, *p*, *q*, *r*, Z, ϕ, θ] = 0), and u_r equals the forward speed, U_o, equations 11 through 16 can be simplified as

$$u_r = U_o \tag{17}$$

$$m\dot{v}_r = -mU_o r + \Delta Y_f(t) \tag{18}$$

$$I_{zz}\dot{r} = \Delta N_f(t) \tag{19}$$

$$\dot{\psi} = r \tag{20}$$

$$\dot{X} = U_o \cos\psi - v_r \sin\psi + U_{cx} \tag{21}$$

$$\dot{Y} = U_o \sin\psi - v_r \cos\psi + U_{cy} \tag{22}$$

Through the assumption of 'small' motions, fluid forces can be linearized using Taylor series expansion. The expression for the transverse and rotational force can then be expressed as in equation (23) and (24).

$$Y_f = Y_{\dot{v}_r}\dot{v}_r + Y_{v_r}v_r + Y_{\dot{r}}\dot{r} + Y_r r \tag{23}$$

$$N_f = N_{\dot{v}_r}\dot{v}_r + N_{v_r}v_r + N_{\dot{r}}\dot{r} + N_r r \tag{24}$$

Where:

$Y_{\dot{v}_r} = \frac{\partial Y_f}{\partial \dot{v}_r}$ (Added mass in sway coefficient)

$\mathrm{Y}_{\dot{\mathrm{r}}} = \frac{\partial \mathrm{Y}_{\mathrm{f}}}{\partial \dot{\mathrm{r}}}$ (Added mass in yaw coefficient)

$\mathrm{Y}_{\mathrm{v}_\mathrm{r}} = \frac{\partial \mathrm{Y}_{\mathrm{f}}}{\partial \mathrm{v}_{\mathrm{r}}}$ (Coefficient of sway force induced by side slip)

$\mathrm{Y}_{\mathrm{r}} = \frac{\partial \mathrm{Y}_{\mathrm{f}}}{\partial \mathrm{r}}$ (Coefficient of sway force induced by yaw)

$\mathrm{N}_{\dot{\mathrm{v}}_\mathrm{r}} = \frac{\partial \mathrm{N}_{\mathrm{f}}}{\partial \dot{\mathrm{v}}_{\mathrm{r}}}$ (Added mass moment of inertia in sway coefficient)

$\mathrm{N}_{\dot{\mathrm{r}}} = \frac{\partial \mathrm{N}_{\mathrm{f}}}{\partial \dot{\mathrm{r}}}$ (Added mass moment of inertia in yaw coefficient)

$\mathrm{N}_{\mathrm{v}_\mathrm{r}} = \frac{\partial \mathrm{N}_{\mathrm{f}}}{\partial \mathrm{v}_{\mathrm{r}}}$ (Coefficient of sway moment from side slip)

$\mathrm{N}_{\mathrm{r}} = \frac{\partial \mathrm{N}_{\mathrm{f}}}{\partial \mathrm{r}}$ (Coefficient of sway moment from yaw)

In addition, the action of the rudder will produce forces that when linearized are $\mathrm{Y}_{\delta}\delta_{\mathrm{r}}(\mathrm{t})$ and $\mathrm{N}_{\delta}\delta_{\mathrm{r}}(\mathrm{t})$. The dynamics of the vehicle are thus defined as:

$$m\dot{v}_r = -mU_o r + Y_{\dot{v}_r}\dot{v}_r + Y_{v_r}v_r + Y_{\dot{r}}\dot{r} + Y_r r + Y_\delta \delta_r(t) \tag{25}$$

$$I_{zz}\dot{r} = N_{\dot{v}_r}\dot{v}_r + N_{v_r}v_r + N_{\dot{r}}\dot{r} + N_r r + N_\delta \delta_r(t) \tag{26}$$

$$\dot{\psi} = r \tag{27}$$

Expressed in matrix form yields

$$\begin{bmatrix} m - Y_{\dot{v}_r} & -Y_{\dot{r}} & 0 \\ -N_{\dot{v}_r} & I_{zz} - N_{\dot{r}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{v}_r \\ \dot{r} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} Y_{v_r} & Y_r - mU_o & 0 \\ N_{v_r} & N_r & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_r \\ r \\ \psi \end{bmatrix} + \begin{bmatrix} Y_\delta \\ N_\delta \\ 0 \end{bmatrix} \delta_r(t) \tag{28}$$

Johnson [8] further simplified the mass matrix by letting the cross coupling terms equal to zero resulting in a final vehicle dynamics define as:

$$\begin{bmatrix} m-Y_{\dot{v}_r} & 0 & 0 \\ 0 & I_{zz}-N_{\dot{r}} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \dot{v}_r \\ \dot{r} \\ \dot{\psi} \end{bmatrix}=\begin{bmatrix} Y_{v_r} & Y_r-mU_o & 0 \\ N_{v_r} & N_r & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} v_r \\ r \\ \psi \end{bmatrix}+\begin{bmatrix} Y_\delta \\ N_\delta \\ 0 \end{bmatrix}\delta_r(t) \qquad (29)$$

and determined the hydrodynamic coefficients and mass properties for ARIES as follows:

$m-Y_{\dot{v}_r}$	m	$I_{zz}-N_{\dot{r}}$	Y_δ	Y_{v_r}	Y_r	N_δ	N_{v_r}	N_r
456.76	222.26	215	69.90	-68.16	406.30	-35.47	-10.89	-88.34

Table 1. Calculated Hydrodynamic Steering Coefficients and mass properties of the ARIES AUV

Resulting in the following forward velocity dependent matrix form:

$$\begin{bmatrix} 456.76 & 0 & 0 \\ 0 & 215 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \dot{v}_r \\ \dot{r} \\ \dot{\psi} \end{bmatrix}=\begin{bmatrix} -68.16 & 406.3-(222.26)U_o & 0 \\ -10.89 & -88.34 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} v_r \\ r \\ \psi \end{bmatrix}+\begin{bmatrix} 69.9 \\ -35.47 \\ 0 \end{bmatrix}\delta_r(t) \qquad (30)$$

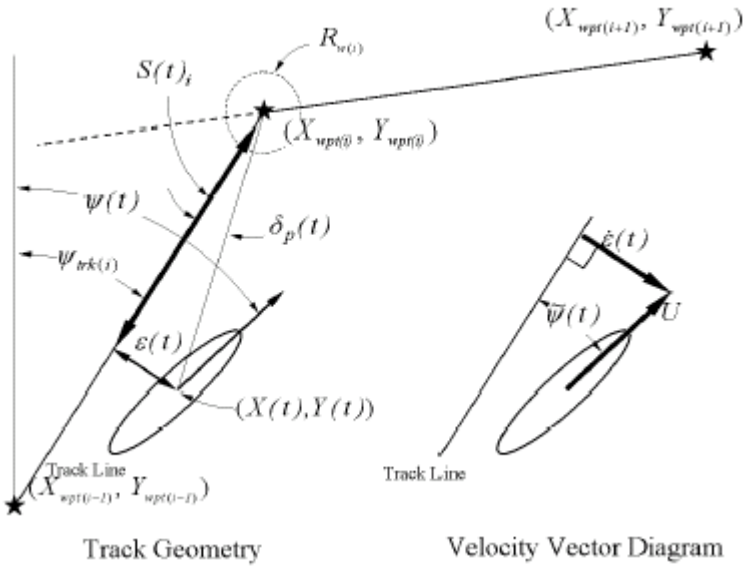
C. ARIES CONTROL LAWS FOR STEERING AND CROSS TRACK ERROR

In order to design a trajectory path for the ARIES AUV to steer towards the docking cage, control laws in the horizontal plane would need to be in placed. There is existing steering and cross track error controller design for ARIES based on sliding mode control theory presented by Healey and Lienard (1993) which can help to jump-start the trajectory design process. Using these existing controllers, the new trajectory path design will be presented in the next chapter and further investigation on compatibility and limitations of these control laws with the designed trajectory path will be studied.

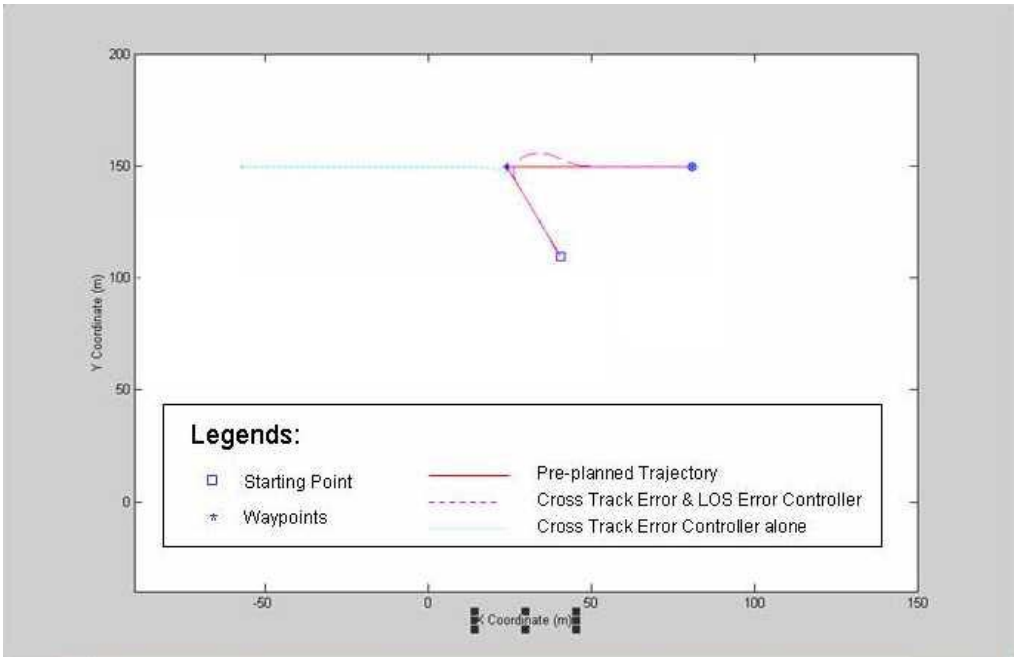
This section describes the existing control laws used by ARIES for steering and determining the cross track error which provides the necessary background knowledge for subsequent work in this thesis.

Marco and Healey [3] proposed the use of a combination of line of sight guidance and a cross track error (CTE) control. The cross track error control allows for ARIES to follow a straight-line path. However, as the cross track error control cannot guarantee stability for large heading errors as can be seen in

Figure 3, 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 2 shows the track geometry and velocity vector diagrams used in the design of the controllers.



Track Geometry and Velocity Vector (From: Marco and Healey [3])



Comparison of Cross Track Error & LOS Error Controller versus Cross Track Error Controller alone

1. Cross Track Error (CTE) Controller

Cross track error, $\varepsilon(t)$ is defined as the perpendicular distance between the center of the vehicle ($X(t), Y(t)$) and the adjacent track line. The objective of the cross track error controller is to minimize $\varepsilon(t)$ so that the vehicle will follow the pre-planned path as closely as possible. With CTE control in place, ARIES can be guided to follow pre-planned trajectory using waypoints navigation.

The track angle is defined as:

$$\psi_{trk(i)} = \arctan 2(Y_{wpt(i)} - Y_{wpt(i-1)}, X_{wpt(i)} - X_{wpt(i-1)}) \tag{31}$$

Where:

$(X_{wpt(i)}, Y_{wpt(i)})$ and $(X_{wpt(i-1)}, Y_{wpt(i-1)})$ are the current and previous way points respectively.

The cross track heading error for the i^{th} segment is then defined as:

$$\tilde{\psi}(t)_{CTE(i)} = \psi(t) - \psi_{trk(i)} \tag{32}$$

Where:

$\tilde{\psi}(t)_{CTE(i)}$ must be normalized to lie between ± 180 degrees.

The difference between the current vehicle position and the next waypoint is:

$$\tilde{X}(t)_{wpt(i)} = X_{wpt(i)} - X(t) \tag{33}$$

$$\tilde{Y}(t)_{wpt(i)} = Y_{wpt(i)} - Y(t) \tag{34}$$

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

$$S(t)_i = [\tilde{X}_{wpt(i)} \quad \tilde{Y}_{wpt(i)}] \bullet [(X_{wpt(i)} - X_{wpt(i-1)}) \quad (Y_{wpt(i)} - Y_{wpt(i-1)})] / L_i \tag{35}$$

where L_i is the total track length is defined as the distance between the i^{th} and $(i-1)^{th}$ waypoints and is given by:

$$L_i = \sqrt{(X_{wpt(i)} - X_{wpt(i-1)})^2 + (Y_{wpt(i)} - Y_{wpt(i-1)})^2} \tag{36}$$

and $S(t)_i$ ranges between 0-100 percent of L_i.

The cross track error may now be defined as:

$$\varepsilon(t) = S(t)_i \sin(d_p(t)) \tag{37}$$

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:

$$d_p(t) = \arctan 2(Y_{wpt(i)} - Y_{wpt(i-1)}, X_{wpt(i)} - X_{wpt(i-1)}) - \arctan 2(\tilde{Y}(t)_{wpt(i)}, \tilde{X}(t)_{wpt(i)}) \tag{38}$$

Marco and Healey [3] 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:

$$\sigma(t) = Ur(t)\cos(\tilde{\psi}(t)_{CTE(i)}) + \lambda_1 U \sin(\tilde{\psi}(t)_{CTE(i)}) + \lambda_2 \varepsilon(t) \tag{39}$$

The rudder input is thus expressed as:

$$\begin{aligned}\delta_r(t) = (Ub\cos(\tilde{\psi}(t)_{CTE(i)})^{-1}(&-Uar(t)\cos(\tilde{\psi}(t)_{CTE(i)}) + U(r(t))^2 \sin(\tilde{\psi}(t)_{CTE(i)}) \\ &-\lambda_1 Ur(t)\cos(\tilde{\psi}(t)_{CTE(i)}) - \lambda_2 U \sin(\tilde{\psi}(t)_{CTE(i)}) - \eta(\sigma(t)/\phi)\end{aligned} \tag{40}$$

Where: $0 < \tilde{\psi}(t)_{CTE(i)} < \frac{\pi}{2}$, λ_1 = 0.6, λ_1 = 0.1, η = 0.1 and ϕ = 0.5.

When $\tilde{\psi}(t)_{CTE(i)} > \frac{\pi}{2}$, ARIES will still 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.

2. Line Of Sight Error Controller

According to Marco and Healey [3], the second order heading control model is:

$$\dot{r}(t) = ar(t) + b\delta_r(t) + (disturbances) \tag{41}$$

$$\dot{\psi}(t) = r(t) \tag{42}$$

Where:

a = -0.30 $\sec^{-1}$ (determined from past in-water experiments)

b = -0.1125 $\sec^{-2}$ (determined from past in-water experiments)

$\delta_r(t)$ = Stern rudder angle.

The sliding surface and stern rudder command for heading control is thus defined as:

$$\sigma(t) = -0.9499r(t) + 0.1701(\psi_{com} - \psi(t)) \tag{43}$$

$$\delta_r(t) = -1.543(2.5394r(t) + \eta \tanh(\sigma(t)/\phi)) \tag{44}$$

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

The heading command and LOS error could be determined from:

$$\psi(t)_{com(LOS)} = \arctan 2(\tilde{Y}(t)_{wpt(i)}, \tilde{X}(t)_{wpt(i)}) \tag{45}$$

$$\tilde{\psi}(t)_{LOS} = \psi(t)_{com(LOS)} - \psi(t) \tag{46}$$

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 exceeds 40 degrees. Once the cross track heading error reduces to less than 40 degrees, ARIES utilizes the CTE controller.

III. TRAJECTORY DESIGN FOR DOCKING ARIES

A. INTRODUCTION AND BACKGROUND

As introduced in Chapter 1, the challenge in docking ARIES into the docking cage is the ability of ARIES to navigate and orient itself to align with the cage entrance accurately as it approaches the cage. The task becomes more daunting when the tethered cage is constantly perturbed by ocean current creating oscillatory swinging motions in the horizontal plane at different rates which is difficult to predict and thus model the cage dynamics.

One way to overcome this unknown is to have some angle and position measuring devices on-board the docking cage and an acoustic modem which can transmit the information to ARIES with the shortest possible delay. In experimental work conducted by Marr [6], two commercially available modem systems were installed in the ARIES AUV and the acoustic transmission performance of each was thoroughly investigated. According to Marr [6], the minimum one-way acoustic transmission time for tactical control of the ARIES, neglecting the channel delay due to sound velocity is three seconds. This minimum three seconds update rate of cage position and heading could potentially cause ARIES to miss the cage entrance. This is especially so for fast moving cage dynamics.

With the 3 seconds update rate constraint in mind, trajectory for steering ARIES into the cage entrance is designed using waypoints navigation building on top of existing cross track error and line of sight error controller.

The approach taken for the design process was first to determine the conditions which qualify for successful docking. The number of waypoints and waypoints location were then automatically determined based on the starting point of ARIES and the cage position through a set of algorithms. These waypoints were initially designed to be fixed throughout the docking rendezvous based on fix cage heading. The design was simulated and tested for all conditions to determine the docking operation envelop. The next incremental

efforts in the design were to incorporate the three seconds transmission delay into the simulation followed by simulating a swinging cage dynamics. The simulation results were then analyzed to determine if the proposed design is satisfactory.

B. DOCKING ACCEPTANCE CRITERIA

To determine whether ARIES has entered the cage successfully, first, some details on ARIES and the docking platform would need to be known. According to Johnson [9], the overall dimensions of ARIES were measured as follows:

- Length from nose to end of tail section (not including thrusters) 3.25m
- Width determined longitudinally at the center section 0.4m
- Height determined vertically at the center section 0.25m

Based on ARIES dimensions, the docking system assumed in this study is as shown in Figure 4. It has a cage opening to receive the incoming AUV and a docking tube to provide protective housing for AUV once it has successfully docked. The cage entrance was assumed to be conical in shape with diameter of 1 meter and apex angle of 60°.

In most studies performed on ARIES previously, the authors treated the position of ARIES as a point located at the vehicle center and were more concerned on steering and bringing this point to other locations. In this study, since docking into the cage is the primary concern, the position of the nose of ARIES is also an important parameter to monitor. Therefore, expressed in global reference frame, the coordinate of vehicle's nose at any point in time is $\left(X(t)+1.625\cos\psi(t),\ \ Y(t)+1.625\sin\psi(t)\right)$. The steering and cross track error control in the simulation, however, was still about moving the point at the vehicle center to the point at the apex of the conical cage.

Due to the difficulty in simulating the constraint of motion cause by the conical cage as ARIES made its entrance into the cage, the physical constraint

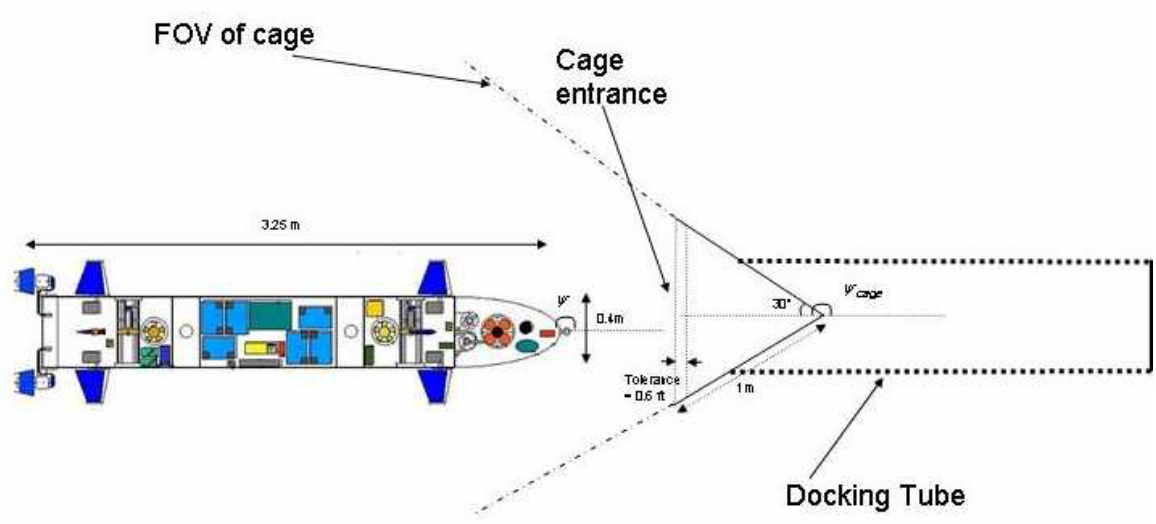
caused by the cage is not modeled, i.e. in the simulation; ARIES could have entered the cage and left it from the side of the cone or it could have entered the cage from the side of the cone. Instead, an acceptance zone was set such that once the nose of ARIES has entered this zone; it was considered to have entered the cage disregard of whether the AUV would eventually move out of the cage boundary in the simulation.

The acceptance zone was defined by two boundaries:

1) ARIES must be within the field of view (FOV) of the conical cage and,

2) An imaginary line of 0.6 ft away from the cage entrance together with the cage entrance as shown in Figure 4 was set as the tolerance limit. ARIES must fall within this tolerance limit.

The tolerance limit was set at 0.6ft to handle the worst-case situation due to discretization in the simulation. The time step for the simulation is set at 0.125sec which means computation of the nose position of ARIES is done at every discrete time interval. The worst-case scenario is when $\psi_{cage} - \psi = 0$ and the nose position of ARIES is calculated just before the cage entrance. With a forward velocity of 1.4m/s, the next calculated position would be 0.574ft away along track. To give some margin, the tolerance limit was thus set at 0.6ft. As the general direction of ARIES is always going towards the apex of the cage, it would also help to eliminate false conclusion of proper docking when ARIES closes into the apex point of the cage from the side of the conical cage.



Pictorial view of ARIES approaching the Docking Platform

C. WAYPOINTS DESIGN

Waypoints navigation is one of the most intuitive autonomous navigation techniques around and has been widely used in Aerospace and Naval industries. Different waypoints can be pre-programmed into ARIES to shape the path that ARIES is expected to follow. In practice, ARIES could have been programmed using a list of objectives in its mission file. These objectives are tasks that the vehicle must achieve before moving on to the next objective. An objective can consist of a set of many waypoints. Docking is therefore, accomplished using the "dock" objective. Alternatively, while ARIES is running a pre-programmed mission, she will be expected to receive modem commands from other vehicles that will trigger the "dock" objective which overwrites the pre-programmed sequence.

To design the waypoints, two important parameters, number of waypoints required and position of waypoints would need to be defined. Correct placement and number of waypoints is important as it will determine the path towards the cage entrance with sufficient settling time for the controller to track the path. This is especially important when the vehicle is very close to the cage where path tracking accuracy is crucial. For a cage with fix heading, the challenge in the design of waypoints is the capability in handling different vehicle's starting

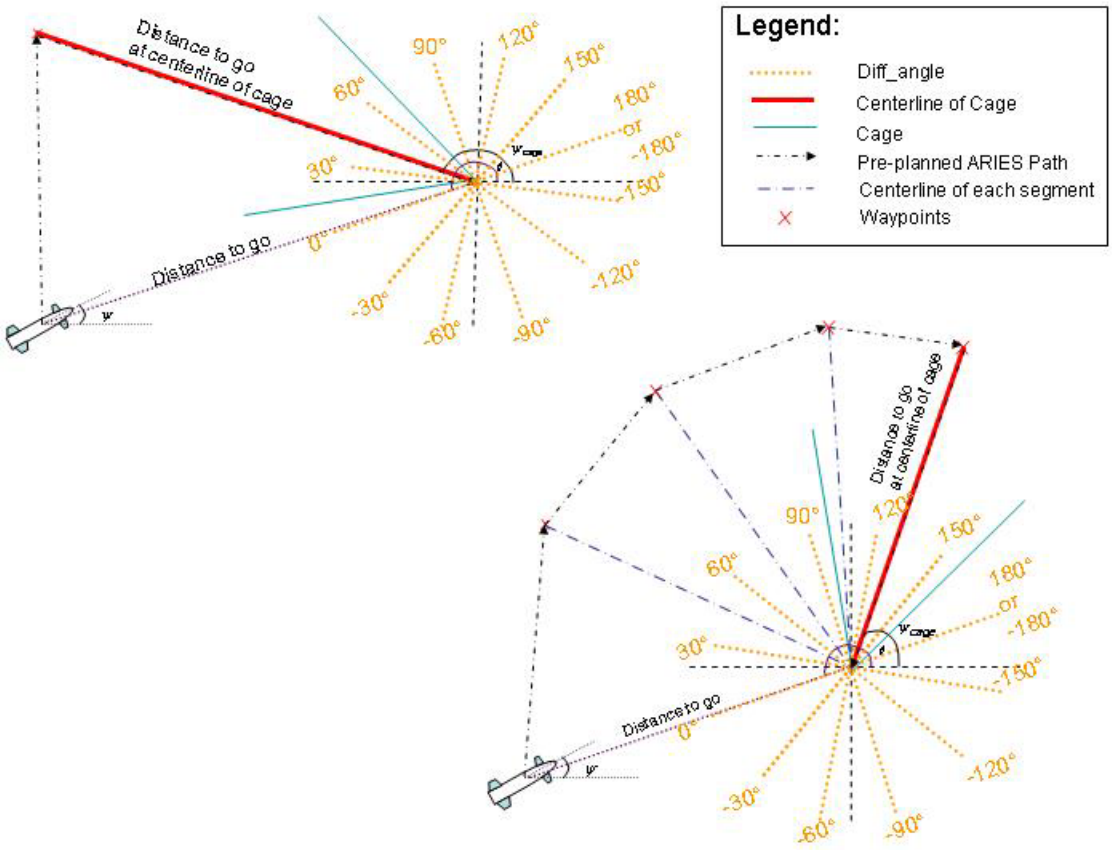
position and heading in respect to the docking cage's position and heading during the docking sequence. To achieve this, a parameter "Diff_angle" is defined as the difference between ψ_{cage} and the line of sight angle between ARIES' starting point in the docking sequence and apex point of the conical cage, θ where Diff_angle ranges from 0 to ± 180 and the centerline of the cage is used to determine the cage heading angle, ψ_{cage}. The range of Diff_angle is further divided into 12 segments as shown in Figure 4. These segments are 0° – $\pm 30°$, $\pm 30°$ – $\pm 60°$, $\pm 60°$ – $\pm 90°$, $\pm 90°$ – $\pm 120°$, $\pm 120°$ – $\pm 150°$ and $\pm 150°$ – $\pm 180°$. Depending on which segment Diff_angle is in, the number of waypoints will vary as shown in Table 2.

Diff_angle Segment	No. of waypoints
0° – $\pm 30°$	1
$\pm 30°$ – $\pm 60°$	2
$\pm 60°$ – $\pm 90°$	3
$\pm 90°$ – $\pm 120°$	4
$\pm 120°$ – $\pm 150°$	5
$\pm 150°$ – $\pm 180°$	6

Table 2. Allocating number of waypoints

In practice, ARIES will automatically determine the number of waypoints based on the parameter "diff_angle". For example, if $-30^{\circ} \le Diff\_angle \le 30^{\circ}$, there will only be one waypoint which is located at the apex of the cage. Since ARIES is within the FOV of the cage, there is no reason to have more waypoints to steer path of the vehicle's direction. However, for cases where $Diff\_angle > 30^{\circ} \quad or \quad Diff\_angle < -30^{\circ}$, waypoints are needed to steer ARIES into the right direction as shown in Figure 5. Now, having determined the

number of waypoints, the next task is to decide on the location of these waypoints.



Waypoints Allocation

If ARIES is going to determine the position of the waypoints automatically, certain relationship between the Diff_angle segments and the waypoints will need to be established. One obvious choice is to place the waypoint of each segment along the centerline of that segment with the only two exceptions of the last second waypoint along the centerline of the cage and the last waypoint on the apex of the cage. With this choice, the next decision is to determine where

along the centerline the waypoint would be set. Two possibilities were considered for fix heading cage:

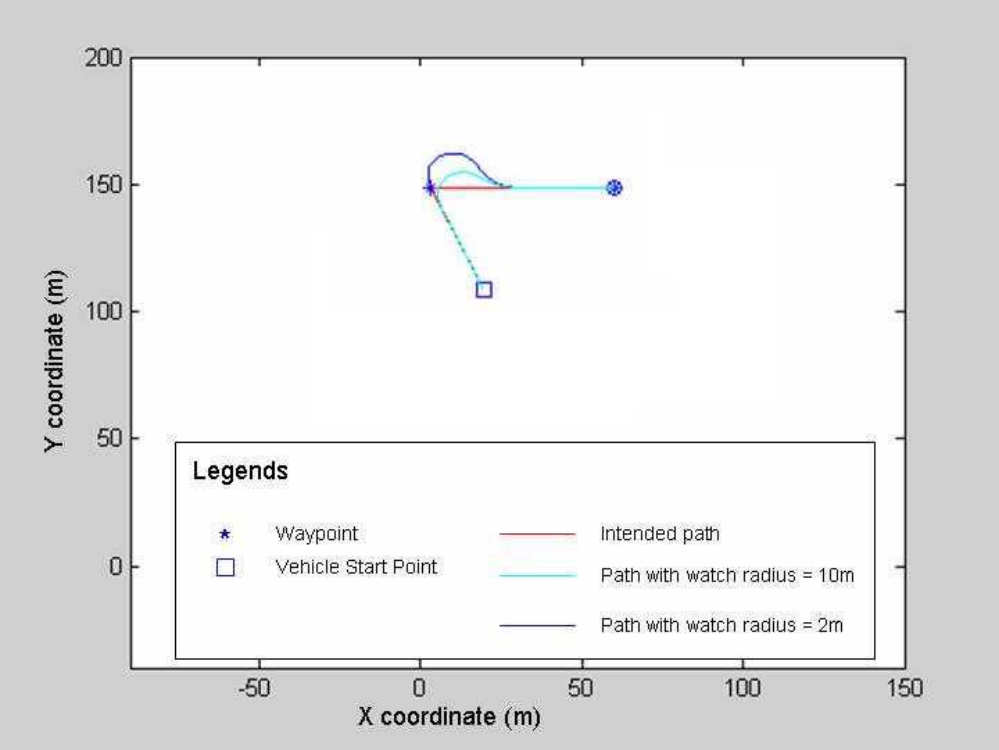
- The waypoints were placed at a length of vehicle's starting distance-to-go along the centerlines of each segments and the cage. Thus, ARIES would travel along the track paths on these waypoints. Upon reaching the last second waypoint which happens to fall along the centerline of the cage, ARIES would steer itself towards the last waypoint at the apex of the cage.
- The second approach was to set the waypoints spiral inward towards the final waypoint at the apex of the cage. Laterally, the waypoints would still be along the centerlines of each segments and the cage but length of the waypoint along the centerline reduces to a fraction of the previous waypoint.

Both methods did have their pros and cons. The latter method took a shorter path and time to the cage but it made sharper turns when ARIES got closer to the cage and thus it was not ideal for docking operation which requires more accuracy and closer tracking of the path when ARIES got closer to the cage. This disadvantage became more evident for moving cage where ARIES might end up spiraling to the last waypoint through the side of the conical cage which in reality, meant a miss in the docking process.

The former approach though took a longer time and traveled a long distance before reaching the cage; it gave sufficient time for the controller's transient overshoot to settle down. As there is no requirement to optimize the docking time, the former approach was adopted for the study and would be used in all the subsequent analysis.

Watch radius is defined as the tolerance radius around the waypoint whereby that waypoint is considered reached when ARIES falls within this radius. A good design of trajectory path includes an appropriate choice of the watch radius. For trajectory that has very sharp change in direction, as can be seen from Figure 6, a large watch radius is preferred. Large watch radius means that vehicle traveling along the track to one waypoint will reach the watch radius of that waypoint earlier and thus transit to the next waypoint earlier. This in fact,

improves the tracking for sharp trajectory path. For the cases of a moving cage or when the vehicle is very close to the cage, a large watch radius can mean a loss in tracking accuracy and thus missing the cage's entrance. Therefore, smaller watch radii are preferred in these cases. In the simulation the watch radius started from 10m and reduces as the waypoints get closer to the cage since terminal accuracy is of outmost important in docking application. It was also reckoned that for fixed cage heading, the sharp trajectory usually happens at the last second waypoint where the vehicle is trying to steer a sharp turn inward towards the cage. Depending on where with respect to the position of the cage ARIES is when the docking sequence is activated, the last second waypoint is always of the same distance away from the cage as the starting point. As long as certain minimum distance constraint is placed on the distance of starting point away from the cage such that it is far enough to give sufficient time for the tracking transient overshoot to settle down, the waypoints design for the docking process is completed.



Trajectory Path with Different Watch Radius

D. SIMULATING DELAYED DATA TRANSFER

Thus far, it was assumed that the data transfer between the acoustic modem of docking platform and ARIES was instantaneous. Unfortunately, Marr's experimental study [6] showed that the minimum acoustic data transfer rate for ARIES is about 0.3 Hz. That was equivalent to one update per 3 second. The update rate of 0.3 Hz can be easily simulated using sample and hold technique where cage heading data is held constant for 3 seconds between updates. This update which contains the Cage heading ψ_{cage} serves as the commanded reference input for the horizontal steering control. For fixed cage heading, a delayed data transfer does not affect the result. However, for moving cage, with erroneous commanded reference input, ARIES will steer towards the wrong point within the updates. This will persist until a new measurement update is obtained for the cage heading which can be used to correct ARIES' path. Two obvious implications of such implementation are

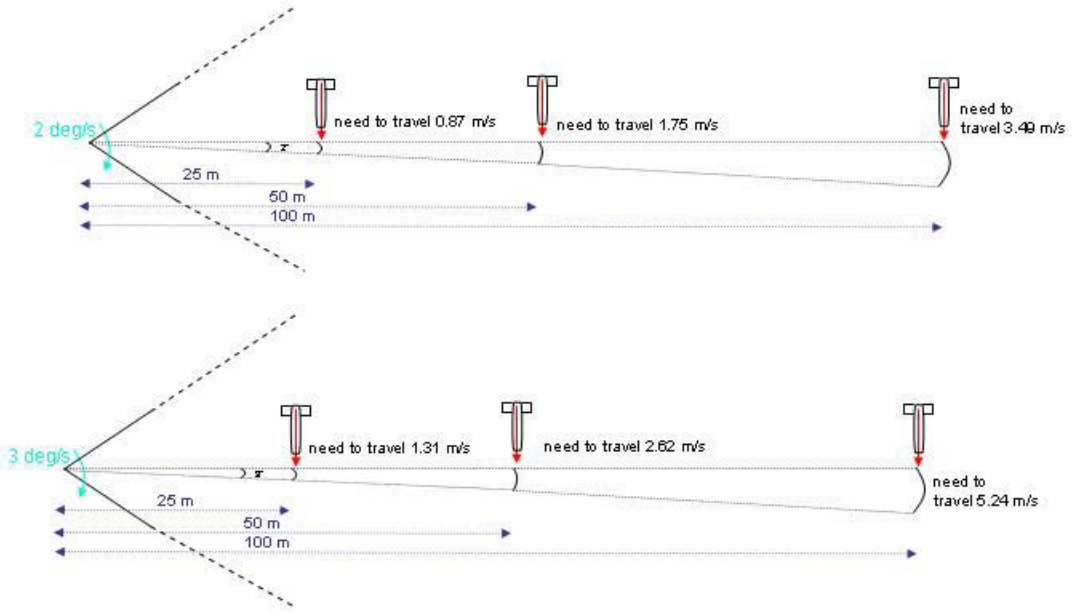
- As the ARIES approaches the last three second of the docking operation, it will not have anymore cage heading updates. This can cause ARIES to miss the cage entrance if the cage is swinging at high amplitude and frequency.
- If the cage is swinging at high rate, with limited forward speed of ARIES coupled with erroneous commanded reference input, ARIES can be easily out-maneuvered.

E. DYNAMIC WAYPOINTS ALLOCATION (DWA)

Dynamic waypoints allocation is a term coined to describe continuous allocation of waypoints based on ARIES and cage's position and heading. This implementation is especially useful for cage in motion as ARIES can adjust its waypoints according to the cage heading when it approaches the cage. The methodology of where to allocate the waypoints and number of waypoints to be allocated is as per stated in section C. The only different is these waypoints allocation changes based on current ARIES and cage's position and heading. In the case where update rate of 0.3 Hz is considered, the allocation of waypoints will be done every 3 seconds when new cage's data arrive.

F. MOVING CAGE DYNAMICS

The problem of homing ARIES into a cage that has fixed heading but oscillating in horizontal position is very different from one where the cage is in fixed position but swinging about a pivot point. The former is a tracking problem in the horizontal plane while the latter is a steering problem to minimum the cross track error and LOS error. With varying cage's heading, the steering problem is made more complicate as shown in Figure 7. For a 2°/s turn rate of the cage, at 25m away in order for ARIES to keep track with it, ARIES need to travel at about 0.87 m/s. At 100m away, the forward velocity requirement increases to 3.49 m/s. For a turn rate of 3°/s, at 100m away, the forward velocity requirement of 5.24 m/s has exceeded ARIES maximum velocity of 3.5 m/s. Based on the worst case scenario illustrated in Figure 7 and assuming that the docking sequence is typically within 100m, the turn rate of the cage should not be more than 2-3°/s. In addition, a cage that tends to change its direction very often would be harder for ARIES to dock. This is because based on DWA method; ARIES has quite a large turn radius and that sluggishness limits it to have a fast change in its direction.



Impact of Turning Rate of Cage on the Requirement of Vehicle's Maneuverability

In this study, the cage motion in the horizontal plane is first assumed to go in one direction at a rate of 2°/s. Subsequently, a sinusoidal swinging motion about a pivot located at the apex of the cage is then considered. The cage motion thus can be modeled with the resulting dynamics matrix:

$$A_r = \begin{pmatrix} 1 & 0 \\ -\omega^2 & 0 \end{pmatrix} \tag{47}$$

To test the steering controller, a model with a period of 180 seconds and amplitude of 60° was chosen. The resulting frequency is 0.035 rad/sec. Implementing in Matlab, the cage dynamics is

initial_cage_psi = 60*sin(0.035*time).

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SIMULATION RESULTS

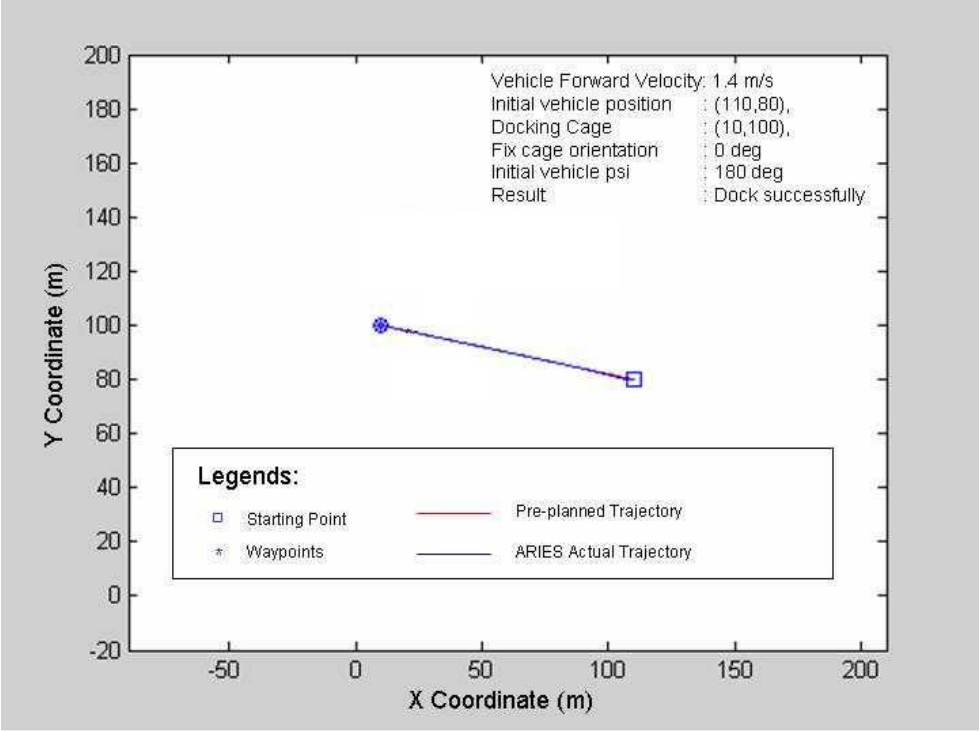
A. FIXED CAGE HEADING

The simulation was performed incrementally starting from a fixed cage heading scenario. Different simulation runs were designed to test the limitation and performance of the ARIES under different initial conditions. The different simulations conducted and their test objectives were summarized in table 3.

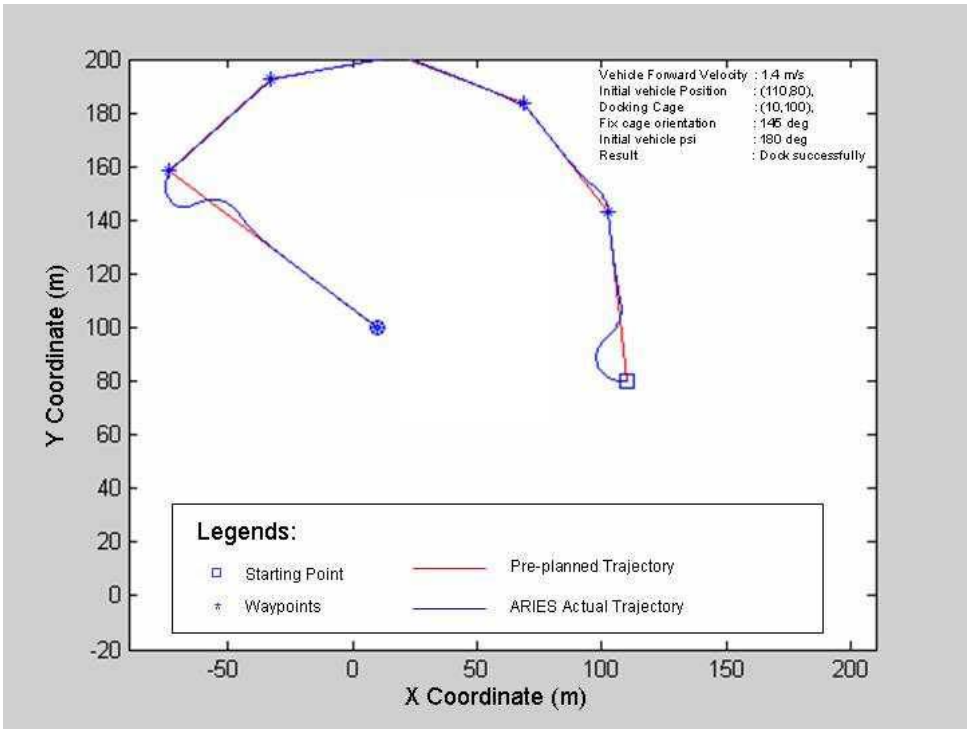
Sim No.	ARIES Forward Velocity	ARIES Initial Position (m)	ARIES Initial Heading	Cage Position (m)	Cage Heading	Test Objective
1	1.4 m/s	(110,80)	180°	(10,100)	0°	Effectiveness of waypoints
2	1.4 m/s	(110,80)	180°	(10,100)	145°	Effectiveness of waypoints
3	1.4 m/s	(65,100)	0°	(10,100)	180°	Minimum Distance requirement
4	1.4 m/s	(50,100)	0°	(10,100)	180°	Minimum Distance requirement
5	1.4 m/s	(50,100)	0°, 50°, 180°, 270°	(10,100)	180°	Impact of Initial heading
6	0.5 m/s	(19,100)	50°	(10,100)	180°	Impact of different initial forward velocity (watch radius set to 2m)
7	2.5 m/s	(145,100)	50°	(10,100)	180°	Impact of different initial forward velocity

Table 3. List of Simulation Runs for Fixed Cage Heading

Simulation # 1 and 2 were designed to verify the trajectory design and docking testing conditions stipulated in chapter 3. Distance between the initial ARIES position and cage's position was set such that there were sufficient time for the controller to settle down from its transient overshoot after switching from one waypoint to the other. Simulation #1 has ARIES initial docking position placed within the FOV of the cage. Under such case, ARIES should follow the line-of-sight track towards the cage entrance. In Simulation #2, since the Diff_angle is 156.3°, there would be six waypoints. Though the initial heading of ARIES was not aligned with the pre-planned path, the LOS error and cross track error controller steered ARIES towards the pre-planned path subsequently. Another observation made was that the last second waypoint always made a very sharp turn inward. That created a large transient overshoot for the controller and it took a substantial amount of time to settle back to its intended track. This characteristic was thus the key determinant on the minimum distance from the cage that the docking sequence could be activated. The results of simulation #1 and 2 were plotted in Figure 7 and 8. In both cases, ARIES managed to dock into the cage successfully.

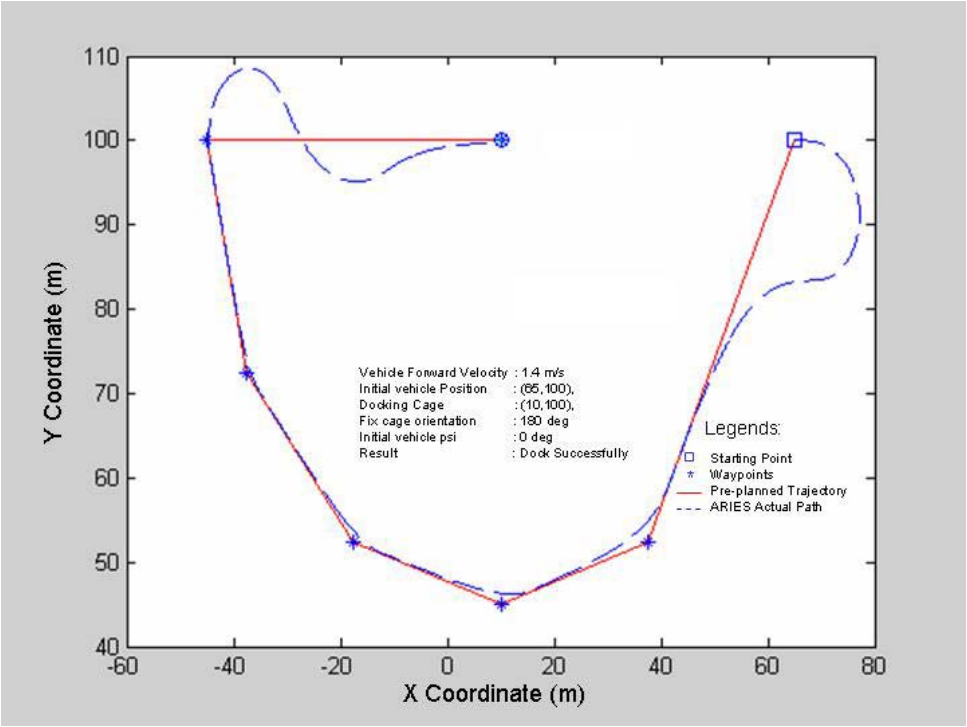


ARIES: (110,80) at 0° @ 1.4 m/s, Cage: (10,100) at 180° (Fixed cage Scenario)

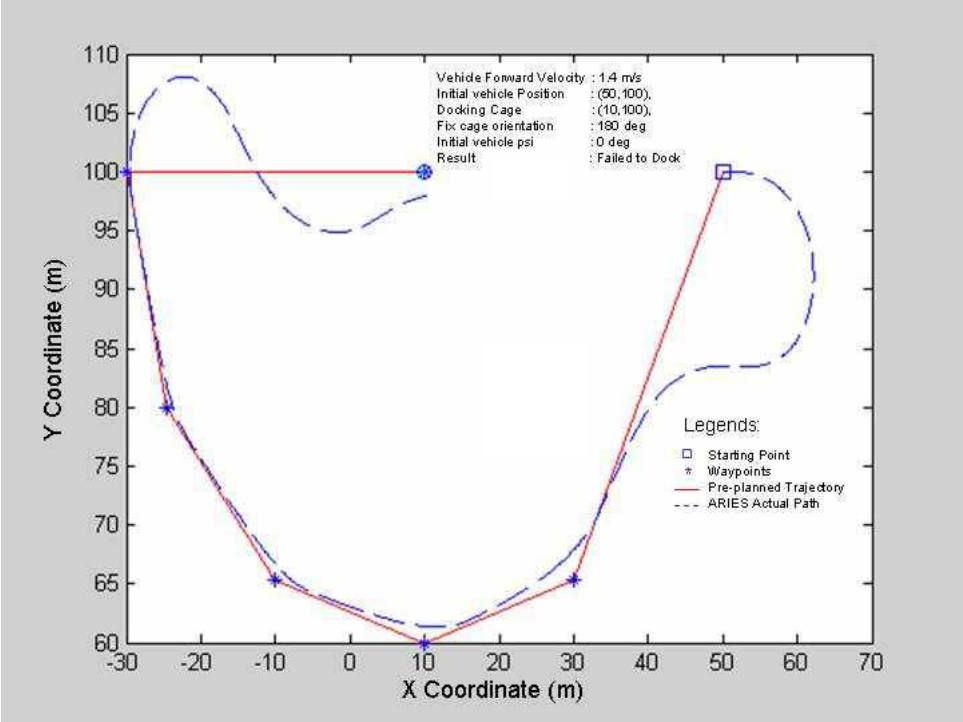


ARIES: (110,80) at 180° @ 1.4 m/s, Cage: (10,100) at 145° (Fixed cage Scenario)

Simulation #3 and #4 attempted to determine the minimum required distance between the ARIES starting position to the cage position. Starting from a distance of 100m away, the distance was reduced with the rest of the parameters kept constant. At a distance 55m away, as shown in Figure 9, ARIES barely made it to the cage entrance. With the rest of the parameters remained the same, at 40m away, ARIES could not make it to the cage as shown in Figure 10. The minimum required distance was found to be around 55m away from the cage.



ARIES: (65,100) at 0° @ 1.4 m/s, Cage: (10,100) at 180° (Fixed cage Scenario)

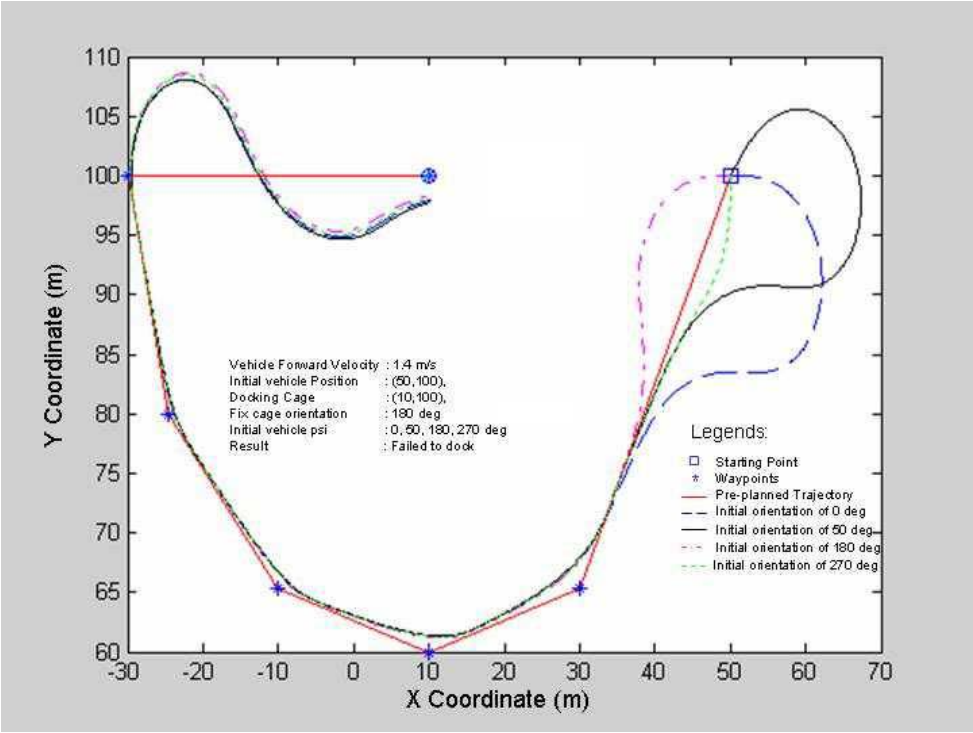


ARIES: (50,100) at 0° @ 1.4 m/s, Cage: (10,100) at 180° (Fixed cage Scenario)

Next, the initial vehicle heading was varied in simulation #5 with the rest of the parameters kept at:

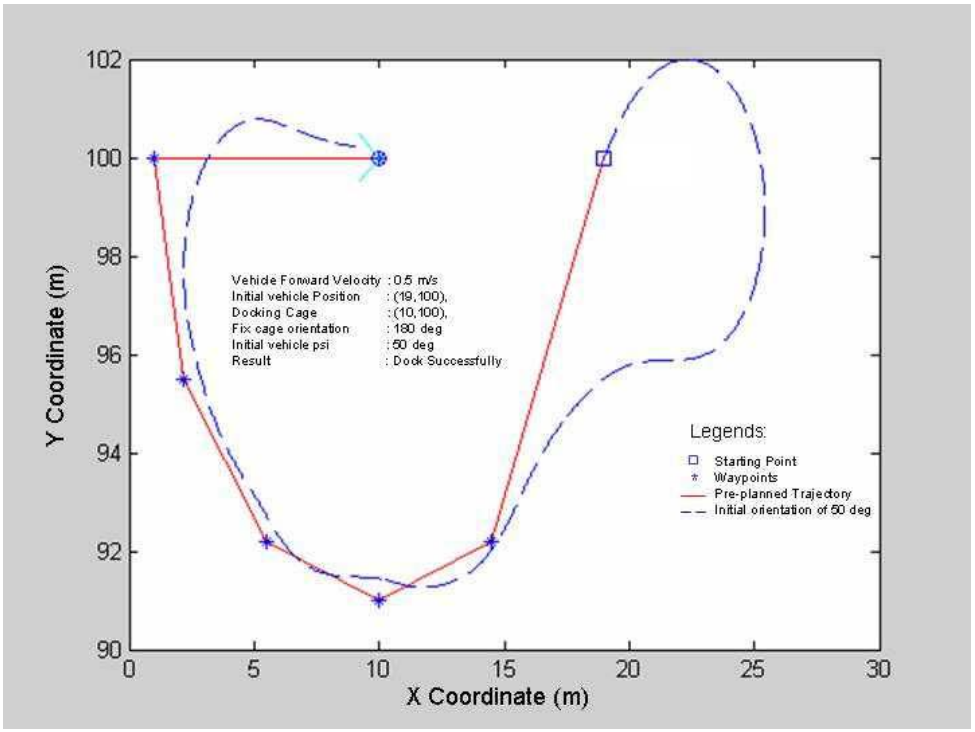
- AREIS Forward Velocity = 1.4 m/s
- ARIES Starting Position = (50,100)
- Cage Position = (10,100)
- Cage Heading = 180°

This was done to determine the impact of initial vehicle heading on the terminal steering accuracy. It was found that when the initial vehicle heading was 180° off the intended path, the controller created the highest error as shown in Figure 11 and thus presented the worst-case scenario. Therefore, in subsequent simulations, the initial vehicle heading was set at this condition so that the most critical condition was tested.

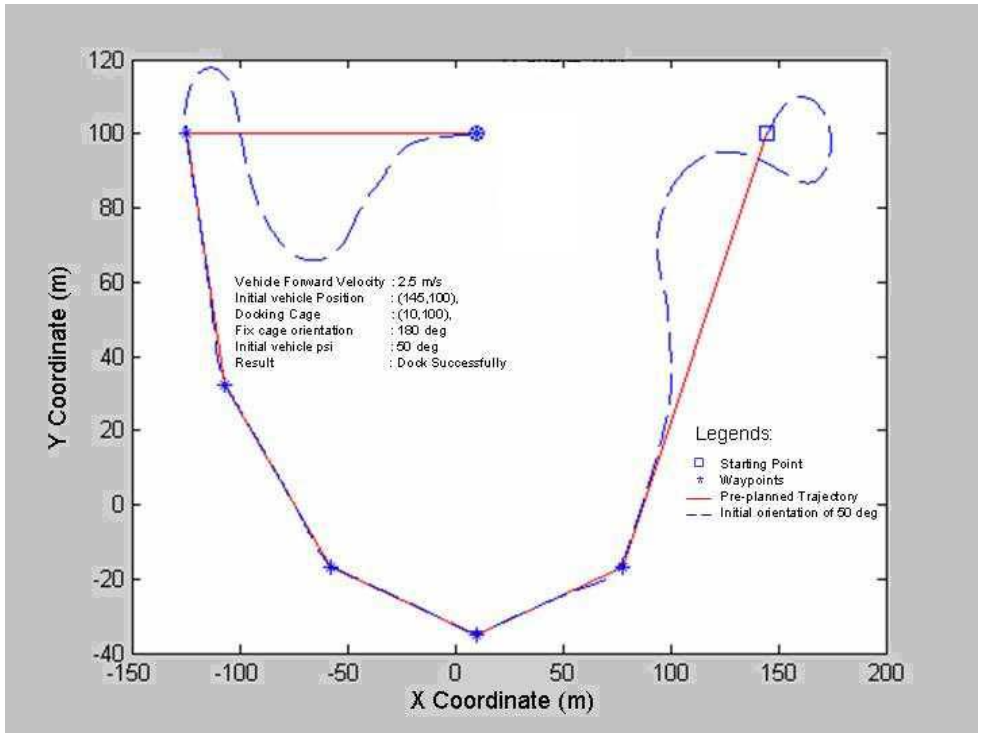


ARIES: (50,100) at 0°, 50°, 180° and 270° @ 1.4 m/s, Cage: (10,100) at 0°
(Fixed cage Scenario)

One of the observations made from the simulations conducted so far was that the turn radius made by the steering controller was large. Referring back to the equations of motion, the forward velocity of the vehicle did play a part in influencing the turn radius. The higher the forward velocity, the larger would be the turn radius. Therefore, in simulation # 6 and 7, the impact of varying the vehicle's forward velocity was investigated. It was found that the minimum distance required reduced to 9m for forward velocity of 0.5 m/s versus a minimum distance of 135m for forward velocity of 2.5 m/s. The conclusion was therefore, for fixed cage heading where timing for docking was not critical; a slower vehicle's forward velocity was desired in order to overcome a sharper turn radius. To put this finding into implementation, a lookup table of vehicle's forward velocity could be created for different vehicle to cage distance.



ARIES: (19,100) at 50° @ 0.5 m/s, Cage: (10,100) at 180° (Fixed cage Scenario)



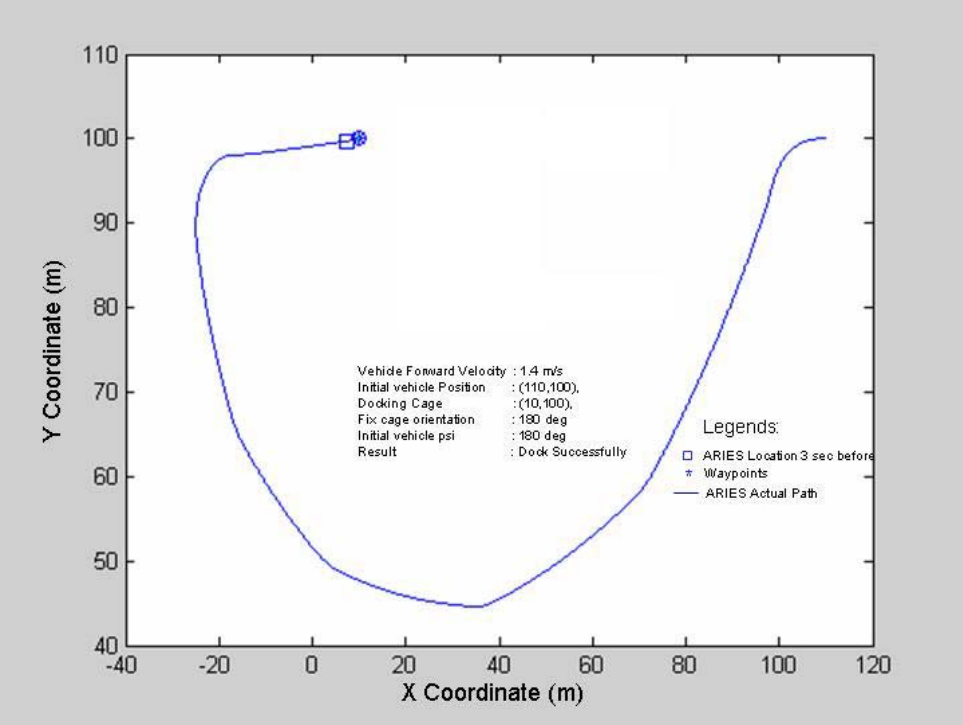
ARIES: (145,100) at 50° @ 2.5 m/s, Cage: (10,100) at 180° (Fixed cage Scenario)

B. DYNAMIC WAYPOINT ALLOCATION

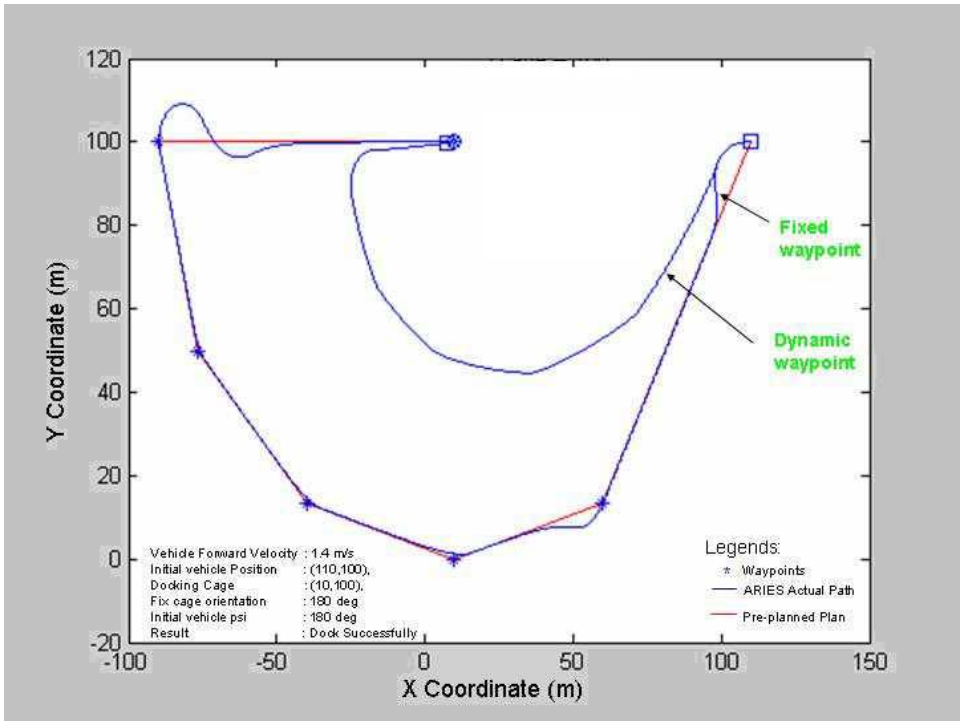
In chapter 3, the concept of dynamic waypoint allocation (DWA) was introduced. The results and performance of such implementation are discussed in this section.

To study the performance of allocation of waypoint based on relative heading and position of vehicle and cage dynamically, the cage heading was first kept constant throughout the simulation. For a start, more benign initial conditions were set with ARIES' starting position set at 100m away, forward velocity 1.4 m/s and, both cage and vehicle headings set at 180°. The waypoints were updated every 3 sec based on the relative heading and position of the vehicle and cage at that instant. The result of the successful docking was showed in Figure 14. It was also observed in Figure 15 that when comparing the trajectory path between fixed and dynamic waypoints allocation, the latter took a shorter and smaller turn radius path since it incrementally corrected its heading every 3 sec as ARIES approached the cage entrance. The result also showed a smoother path and smaller transient overshoot even ARIES made a sharp maneuver. Being able to reduce the transient overshoot was keyed in the reduction of minimum vehicle-cage starting distance apart.

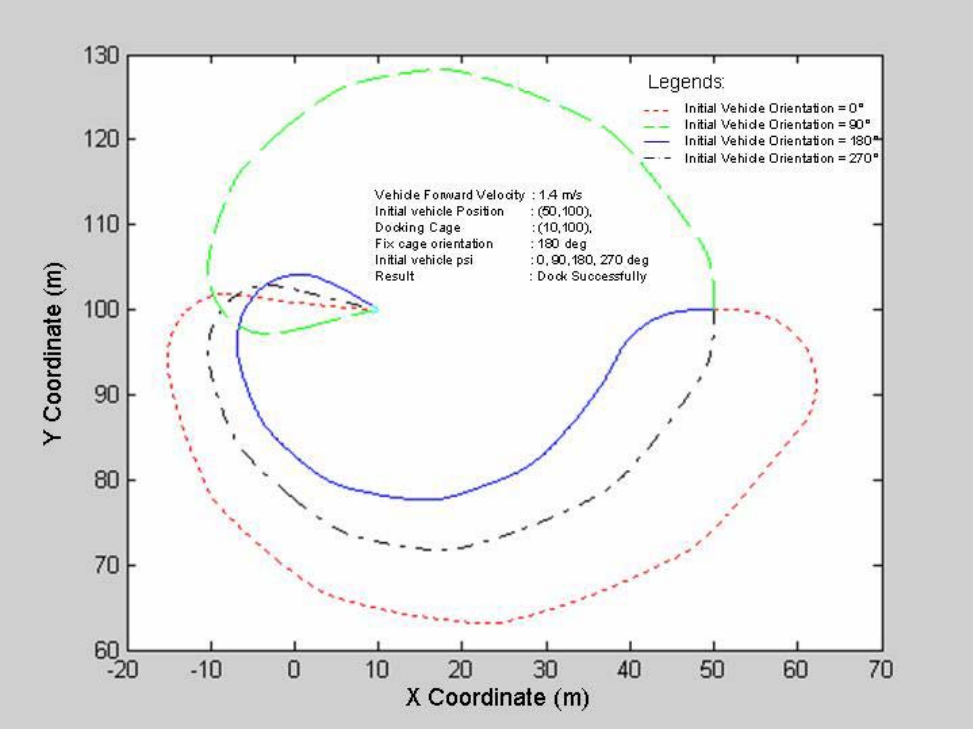
The next simulation conducted was to identify the impact of initial vehicle heading on the terminal steering accuracy. Figure 16 showed the result for initial vehicle headings of 0°, 90°, 180° and 270° where vehicle-cage distance is 40m apart and initial cage heading at 180°. Though all the four initial vehicle headings managed to dock into the cage successfully, the run with initial heading of 180° just marginally met the docking requirement. This presented the most critical initial heading since ARIES needed to make the tightest turn and it nearly missed the cage. The most critical initial vehicle heading thus is when ARIES' initial heading has the same angle as the initial cage heading.



ARIES: (110,100) at 180° @ 2.5 m/s, Cage: (10,100) at 180° (Fixed cage with Dynamic waypoint allocation Scenario)

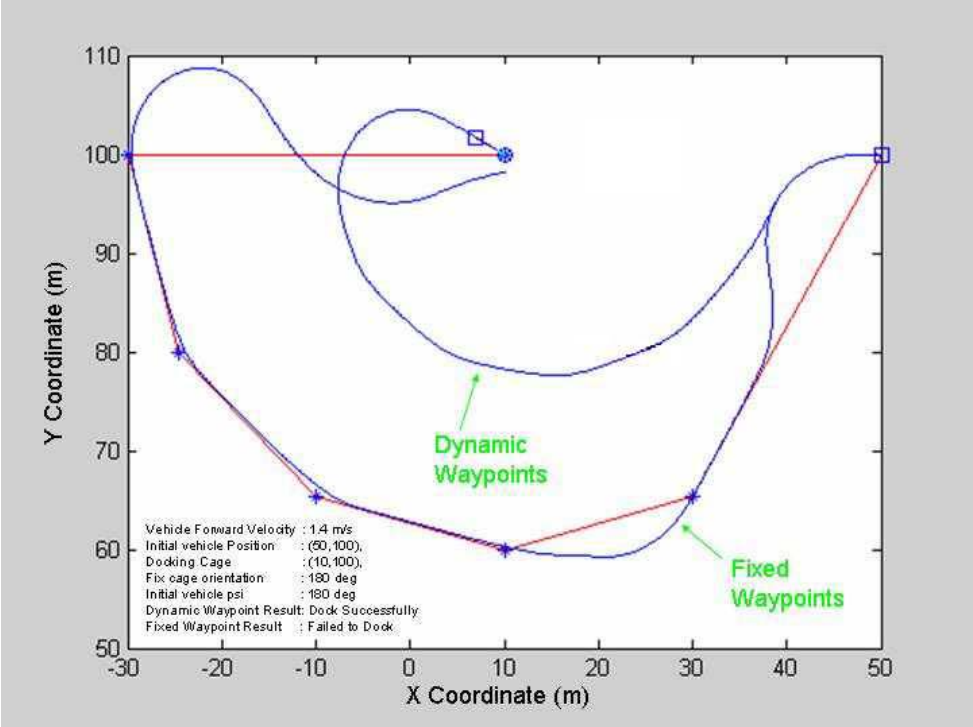


Comparison between Fixed Waypoint allocation and Dynamic Waypoint Allocation Technique



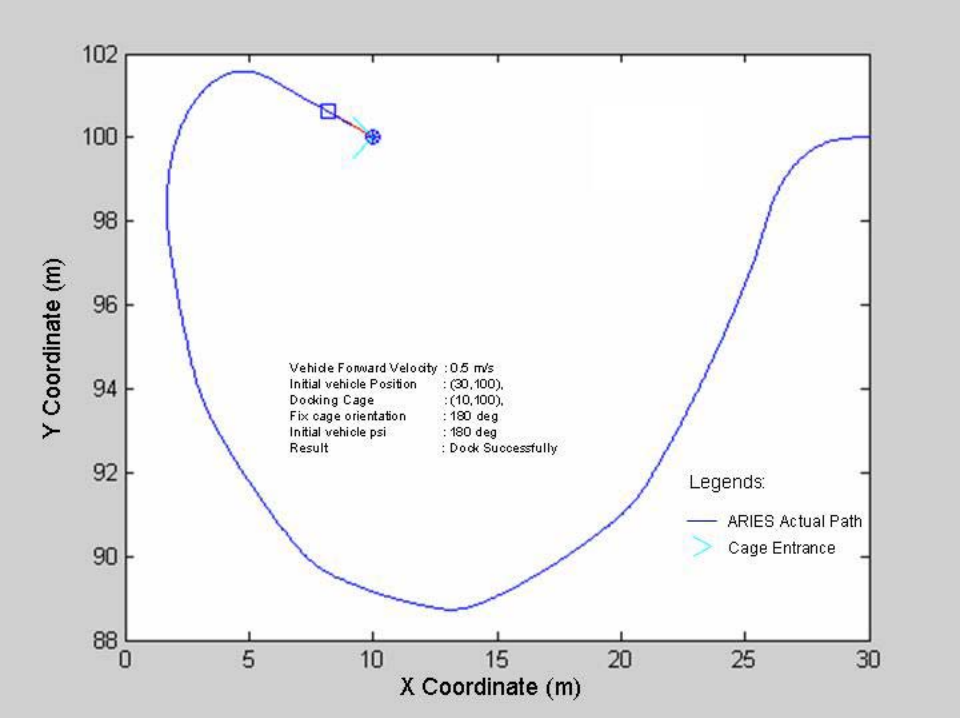
Impact of Initial Vehicle Heading on Terminal Accuracy of Dynamic Waypoints Allocation Method

To further prove that DWA method was more superior in performance as compared to fixed waypoint allocation method, simulations using both methods were ran at conditions where ARIES failed to dock under the fixed waypoint allocation method. At vehicle's forward velocity of 1.4 m/s, vehicle-cage distance of 40 m/s, and initial vehicle and cage heading of 180°, the DWA method proved to be more superior as it managed to dock successfully. This has a lot to do with the reduction of transient overshoot when using the DWA method. Figure 17 showed the performance comparison of both methods.



Performance Comparison between Dynamic and Fixed Waypoints Allocation Methods

However, at forward velocity of 0.5 m/s as seen in Figure 18, the minimum vehicle-cage distance achieved for the DWA case was only 20m as compared to fixed waypoint allocation of 9m. That was again caused by the tight turn radius which is the characteristics of DWA method. The DWA method though has the turn radius constraint at low vehicle forward velocity which capped the minimum vehicle-cage distance at higher value, its ability of always adjusting itself based on dynamic conditions of both the cage and vehicle far outweighed its shortfall. DWA method is the obvious choice for cage with swinging motion scenario.



ARIES: (30,100) at 180° @ 0.5 m/s, Cage: (10,100) at 180° (Fixed cage, Dynamic Waypoints Scenario)

C. SWINGING CAGE SCENARIO

1. Swinging Cage at a rate of 2°/s in Single Direction (Anti-Clockwise)

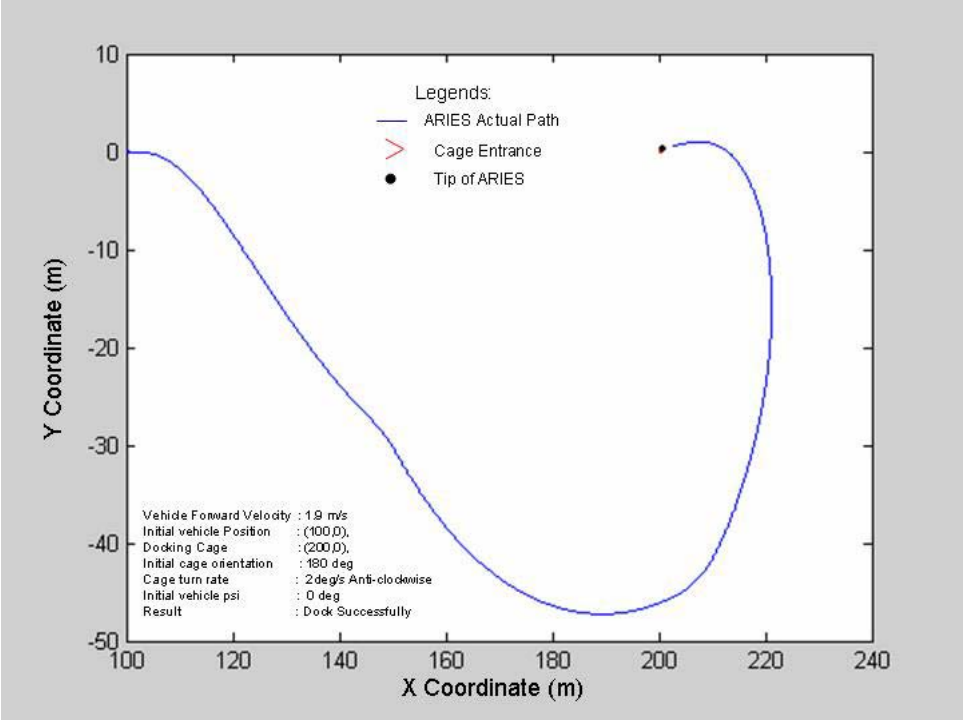
The seemingly straightforward swinging cage problem was in fact made challenging with the constraint imposed in ARIES' forward velocity coupled with the tighter requirement on turn radius as ARIES approached the cage. The problem was not only on tracking the turn angle of the cage but also when and how much to close in to the cage taking into consideration the turn radius of ARIES. The primary parameters that affected the docking accuracy include vehicle forward velocity, waypoint watch radius and location, and vehicle turn radius. Different vehicle-cage distance, initial vehicle and cage heading also affected the trajectory path of ARIES towards the cage and thus these parameters were closely interrelated with some of the primary influencing parameters. It was found through iterative method that at forward velocity of about 1.9 m/s, a good trade-off between forward velocity and required turn radius was achieved. Also, in cases where $diff\_angle \geq 60^{\circ}$ or $diff\_angle \leq -60^{\circ}$, instead of traveling along the track path of waypoints which have same distance to the cage as the vehicle, the last two waypoints were set at 80% of the vehicle-cage distance. That implementation was tailored to allow a faster convergence into the cage.

The simulation was running at a time step of 0.05 sec and for a start, perfect sensor updates of the cage's data were assumed. Also, in the simulation, the cage has a turn rate of 2°/s rotating in anti-clockwise direction. Different simulation runs were performed to verify the ability of the design in steering into a rotating cage.

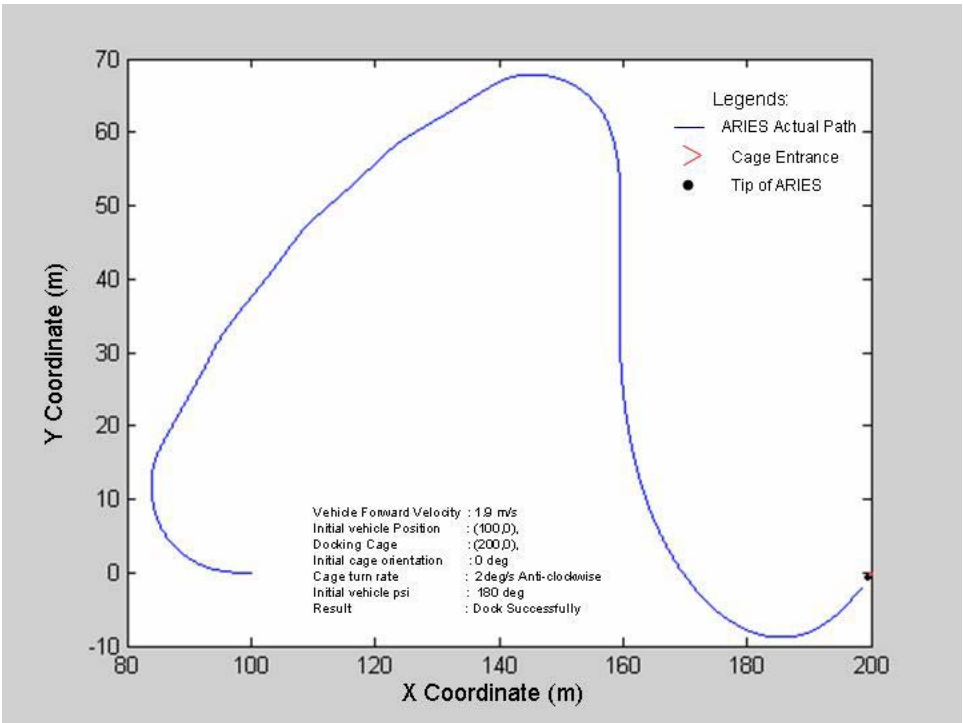
Sim No.	ARIES Forward Velocity	ARIES Initial Position (m)	ARIES Initial Heading	Cage Position (m)	Cage Starting Heading	Test Objective
8	1.9 m/s	(100,0)	0°	(200,0)	180°	Impact of initial heading on the accuracy of docking
9	1.9 m/s	(100,0)	180°	(200,0)	0°	Impact of initial heading on the accuracy of docking
10	1.9 m/s	(100,0)	90°	(200,0)	270°	Impact of initial heading on the accuracy of docking

Table 4. List of Simulation Runs for Rotating Cage

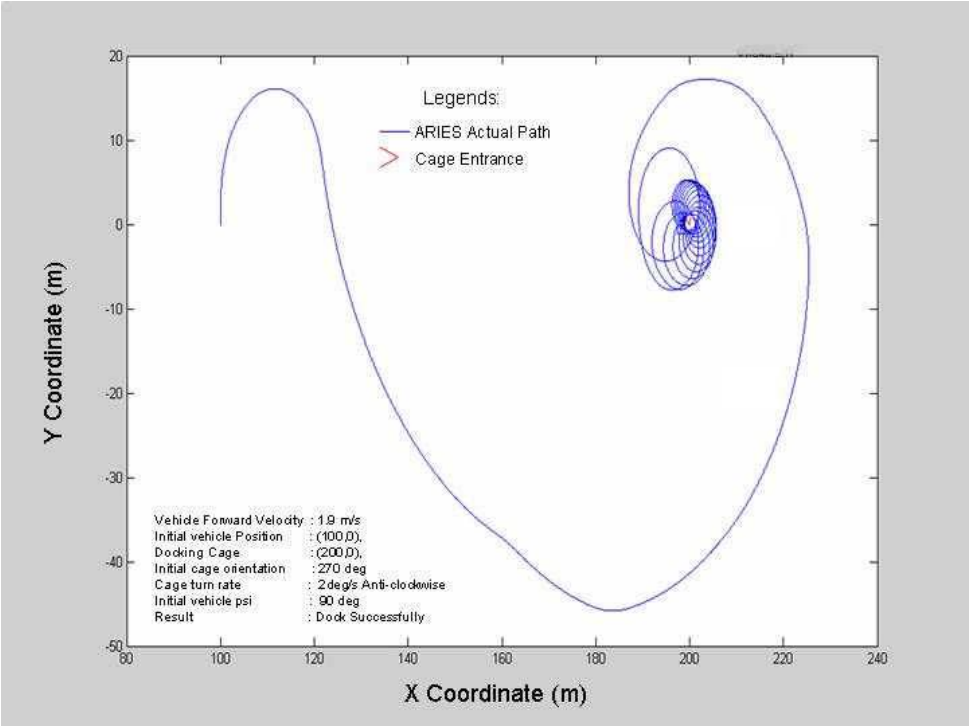
All the three simulations in Table 4 were performed with ARIES having a docking distance of 100m. The only differences were the initial heading of ARIES and the docking cage. From the simulations, it was found that in all three cases as shown in Figure 20-22, ARIES managed to dock successfully. However, it was observed that for simulation #10 as shown in Figure 22, ARIES made multiple attempts before it could finally dock into it. Further investigation on this set of initial condition showed that it happened to be the worst-case scenario.



ARIES: (100,0) at 0° @ 1.9 m/s, Cage: (200,0) at initial heading of 180° at 2°/s Anti-clockwise (Swinging cage, Dynamic Waypoints Scenario)



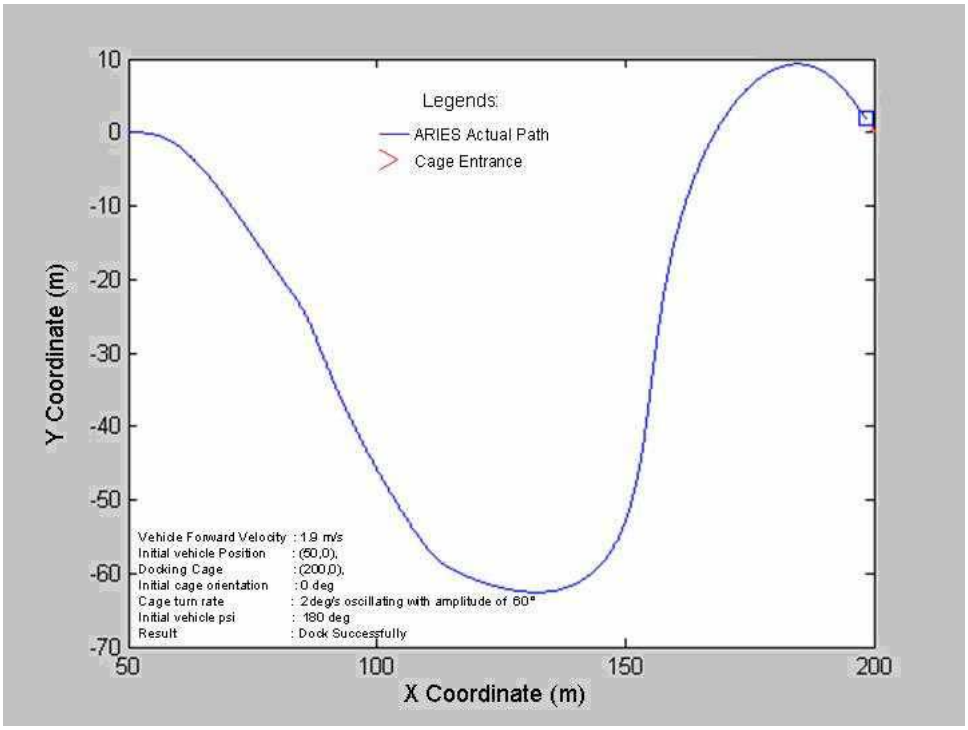
ARIES: (100,0) at 180° @ 1.9 m/s, Cage: (200,0) at initial heading of 0° at 2°/s Anti-clockwise (Swinging cage, Dynamic Waypoints Scenario)



ARIES: (100,0) at 90° @ 1.9 m/s, Cage: (200,0) at initial heading of 270° at 2°/s Anti-clockwise (Swinging cage, Dynamic Waypoints Scenario)

2. Swinging Cage at a rate of 2°/s and Oscillating with Amplitude of 60°

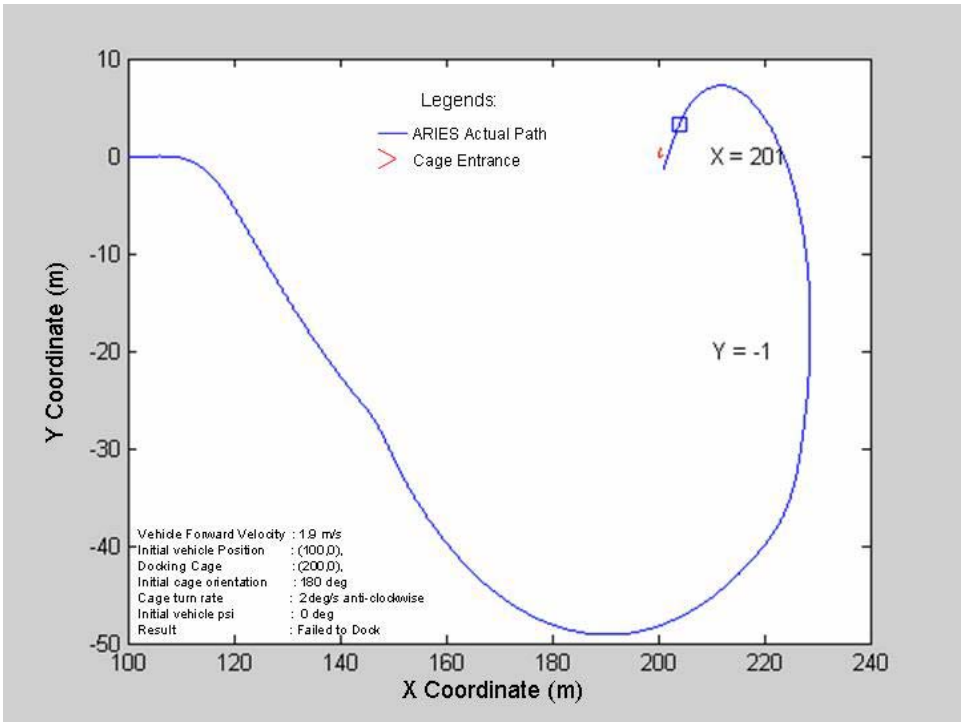
The next simulation (Simulation #11) targeted on the ability of the design in docking into swinging cage with oscillating motion. This is one of the more likely undersea cage motions. In the simulation, the cage was set to rotate in oscillatory motion of amplitude 60° and a turn rate of 2°/s. Figure 23 showed a successful docking result for starting docking distance of 150m.



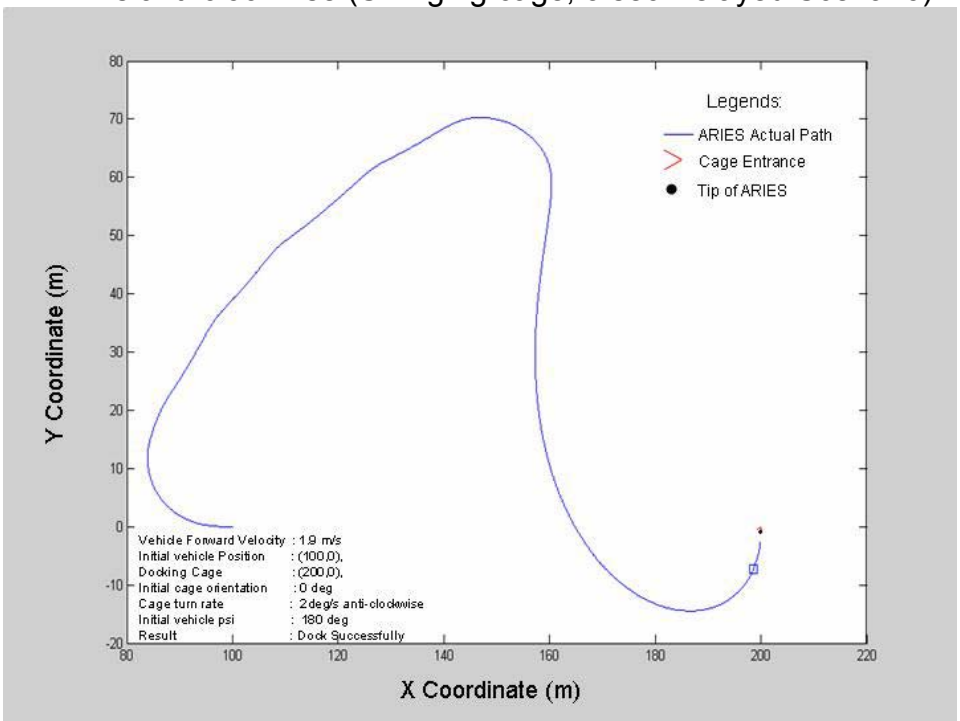
ARIES: (50,0) at 0° @ 1.9 m/s, Cage: (200,0) at initial heading of 180° at 2°/s oscillating with 60° (Swinging cage, Dynamic Waypoints Scenario)

3. Swinging Cage at a rate of 2°/s (With Data Transfer Delay at 3sec)

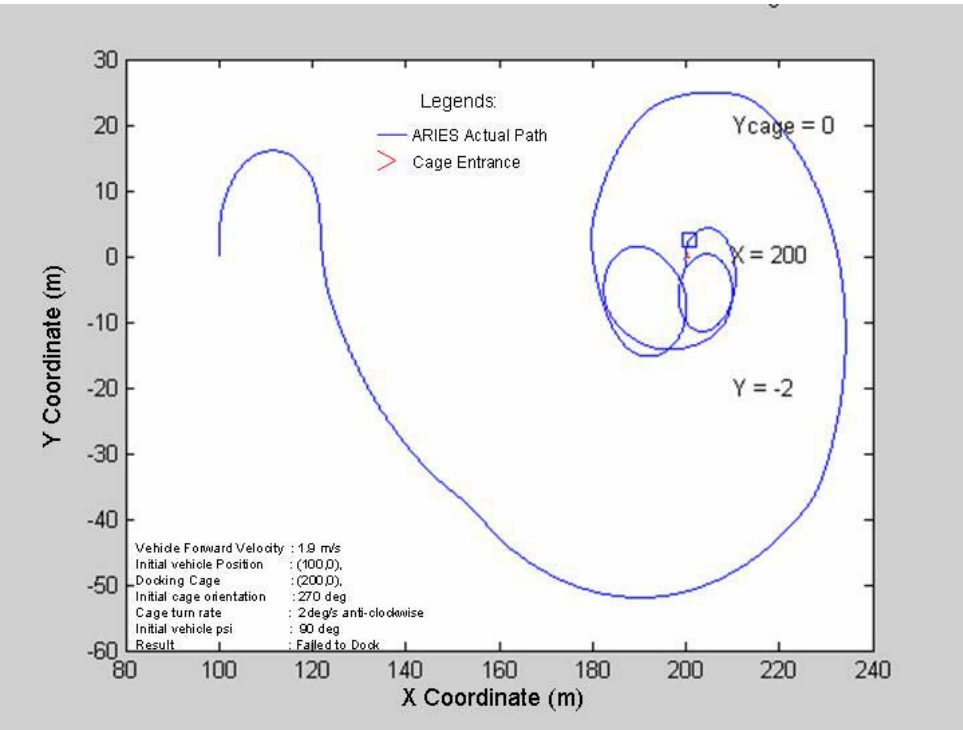
The final set of simulations were to re-run simulation #8-11 but this time with data from the cage updated only once every 3 sec. The results in figure 24-27 showed that out of the 4 simulations, only simulation #9 managed to dock successfully. This set of simulations gave a good indication on the impact of a 3sec delay data on the accuracy of the final docking which led us to the solution elaborated in the next chapter.



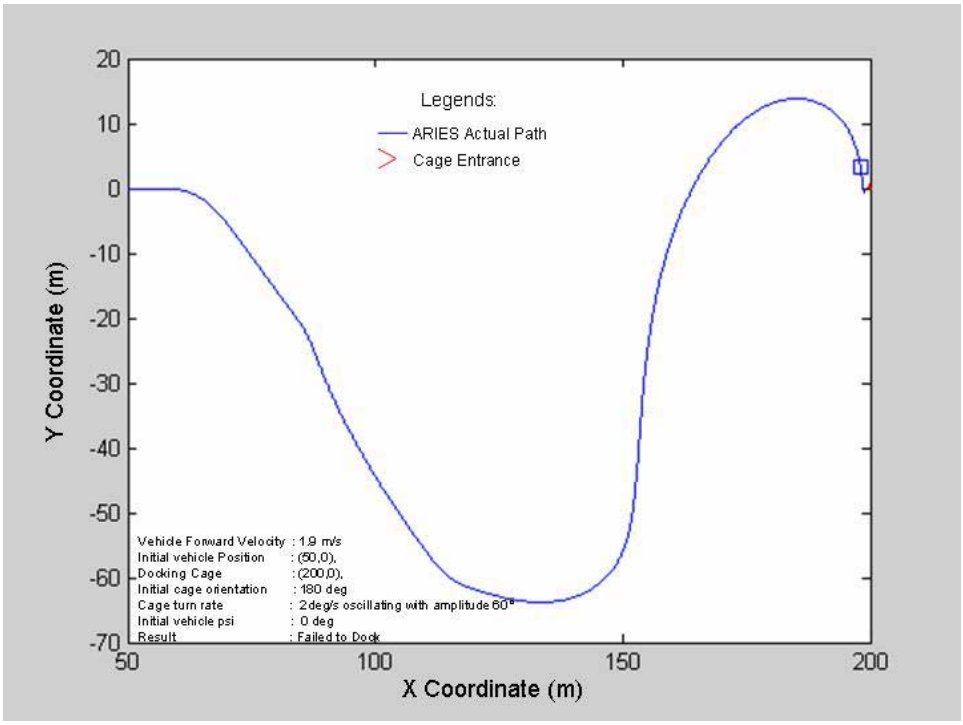
ARIES: (100,0) at 0° @ 1.9 m/s, Cage: (200,0) at initial heading of 180° at 2°/s anti-clockwise (Swinging cage, 3 sec Delayed Scenario)



ARIES: (100,0) at 180° @ 1.9 m/s, Cage: (200,0) at initial heading of 0° at 2°/s anti-clockwise (Swinging cage, 3 sec Delayed Scenario)



ARIES: (100,0) at 90° @ 1.9 m/s, Cage: (200,0) at initial heading of 270° at 2°/s anti-clockwise (Swinging cage, 3 sec Delayed Scenario)



ARIES: (50,0) at 0° @ 1.9 m/s, Cage: (200,0) at initial heading of 180° at 2°/s oscillating with 60° (Swinging cage, 3 sec Delayed Scenario)

THIS PAGE INTENTIONALLY LEFT BLANK

V. SOLUTIONS TO DELAYED DATA TRANSFER

In the absence of any data from the cage with the 3 sec updates, most of the estimation techniques which use a higher update rate sensor to predict the lower update rate sensor's data like complimentary filtering or Kalman filtering would not work. In the previous implementation, the 3 sec transmission delay was simulated using sample and hold technique which meant that within the 3 seconds update ARIES would be steering to the last update cage heading until a new data updates appeared. That implementation could be jerky since every update would create a sudden jump in steering command and that was especially true if the cage was turning at higher rate. Also as mentioned before, at the terminal approach, ARIES could be not receiving any update for up to 3 seconds which was yet another source of error. In addition the sample and hold technique always left the controller chasing after the new cage heading on every new updates which could be too late for ARIES to react.

The proposed solution is to patch and smooth out the emptiness in between the 3 second updates with predicted cage ψ angle value. One of the techniques that can be adopted is first order Euler integration. This method is greatly used in simulation software like in Simulink solver. Basically, the new ψ estimated in every time step in the simulation can be computed using

$$\psi_{new} = \psi_{old} + \dot{\psi}_{old} dt \tag{48}$$

Where:

dt = time step in the simulation

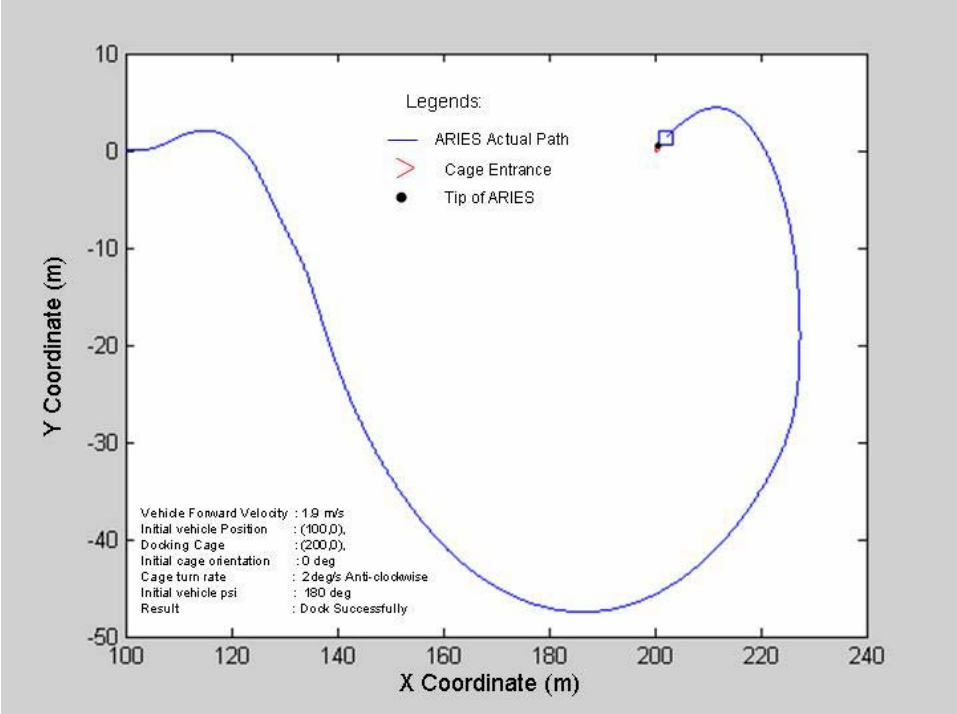
$\dot{\psi}_{old}$ = Cage rotation rate (Positive – Anti-clockwise direction)

In the above implementation, the assumption is that the cage is rotating at either a fix rate or slow rate and thus within the 3 seconds of data updates, a constant cage rotation rate $\dot{\psi}_{old}$ base on last update is used to predict ψ_{new} in the next three seconds. To implement this, data transferred from the cage must include both the cage heading and its rotation rate. For a cage rotation rate of 2-

3°/s, this technique should be sufficient and the implementation is as shown in Appendix C.

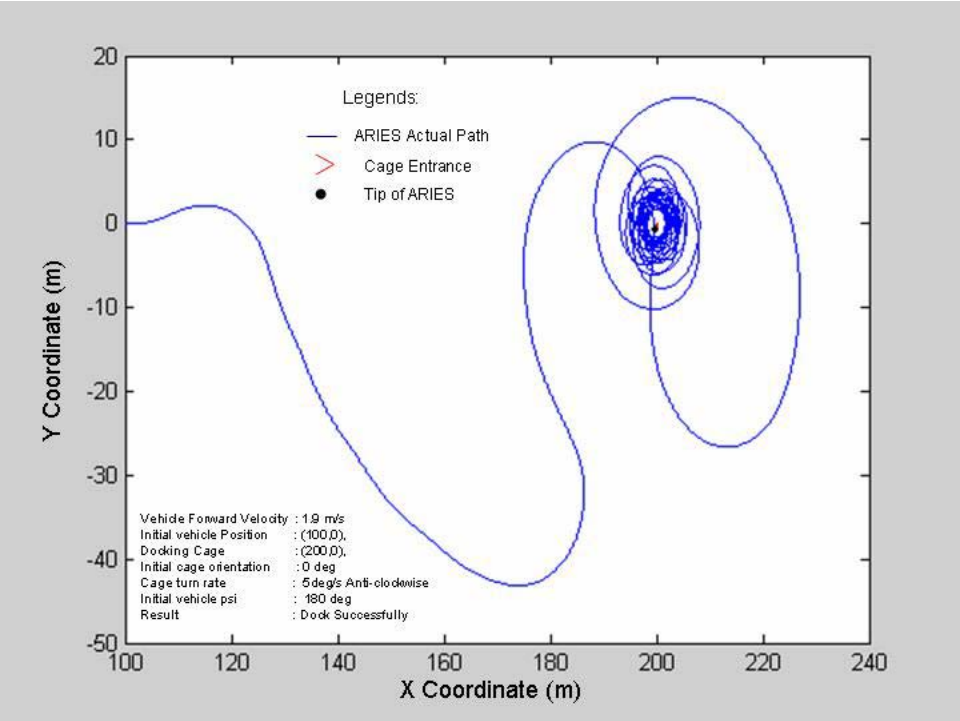
In fact as expected, if the model of the dynamics of the cage is known, the Euler integration method is essentially performing equally well as if there is no data transmission delay. The new cage ψ calculated at each simulation time step just need to follow certain reference cage model and ARIES will have exact cage's data at every simulation time step. However, perfect knowledge of the cage's dynamics and model at all time is not possible in most cases. Therefore, more simulations were done to validate the effectiveness of this method.

For a single directional rotating cage, in order to add in more reality in the simulation, a randomizer varying from 0 to 2 was added to the rotation rate of the cage. That was included to further verify the robustness of the design under uncertain rotation rate. Figure 28 showed that even under slight randomness in the rotation rate, the design is robust enough for ARIES to dock successfully.



ARIES: (100,0) at 0° @ 1.9 m/s, Cage: (200,0) at initial heading of 180° at 2°/s anti-clockwise (3 sec Delayed, with randomizer, Euler Integration implemented Scenario)

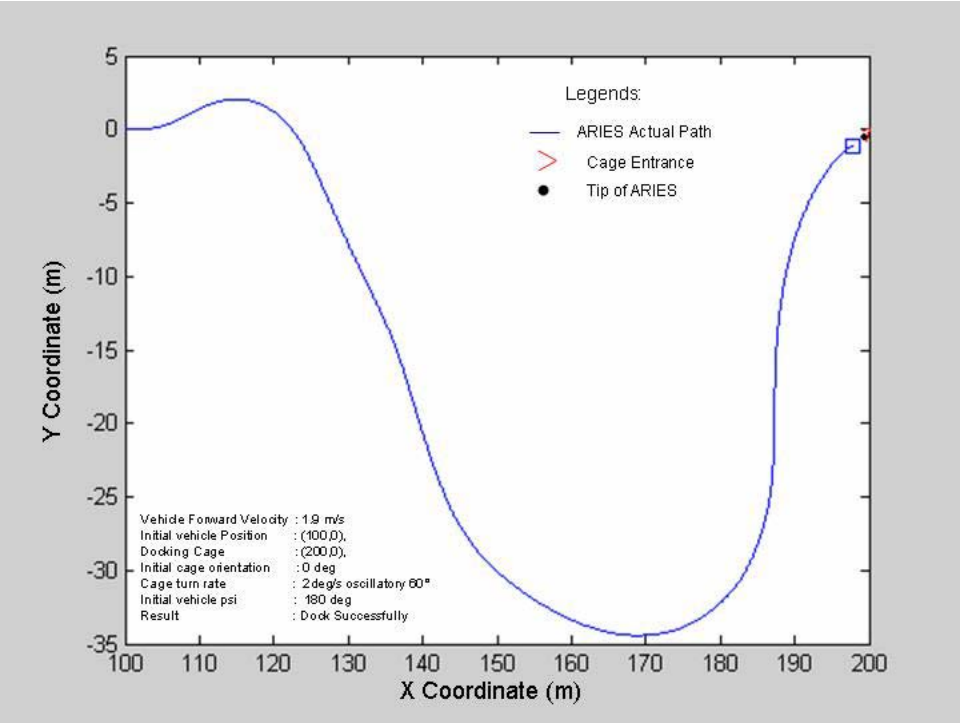
Pushing the rotation rate boundary, in the next simulation, the rotation rate was set to 5°/s with randomizer as well to create the uncertain. The result in Figure 29 showed that with sufficient time given, the design is robust enough to dock into the cage after multiple attempts.



ARIES: (100,0) at 0° @ 1.9 m/s, Cage: (200,0) at initial heading of 180° at 5°/s anti-clockwise (3 sec Delayed, with randomizer, Euler Integration implemented Scenario)

So far in the simulation, the motion considered was in single direction. Finally, the next simulation was conducted to examine the robustness of the design in handling oscillatory swinging cage. Since a sine term that is non-linear was included to simulate the oscillatory motion and Euler integration is linear in nature, more error between the actual and predicted cage heading was expected. In addition, when the actual cage heading changed direction in between updates, the discrepancy between actual and predicted cage heading widened until the next update where the predicted heading, rotation rate and direction of turn were corrected. For the case of 2°/s rotation rate, under the worst-case situation, the maximum error possible was 12° and depending on

when this error occur during the simulation, it could cause ARIES to miss the cage entrance. However, in most cases, the error caused by the change in direction would be much less that 12° and also, for this error to do any impact on the accuracy, this error would need to happen when the vehicle was very close to the cage. The result for swinging cage at 2°/s oscillating with 60° amplitude and initial condition of ARIES: (100,0) at 0° @ 1.9 m/s, Cage: (200,0) at initial heading of 180° was shown in Figure 30 which demonstrated that the design is still robust enough to handle a slow oscillatory motion in this case.



ARIES: (100,0) at 0° @ 1.9 m/s, Cage: (200,0) at initial heading of 180° at 2°/s oscillating with 60° (Swinging cage, Euler Integration implemented Scenario)

VI. CONCLUSION AND RECOMMENDATIONS

A. CONCLUSION

The problem of horizontal steering control of ARIES into a swinging cage is very complex. Many factors affect the final outcome of the docking results. These factors include the choice of controller, position and number of waypoints, how often the waypoints are updated, the choice of watch radius, vehicle forward velocity, vehicle turn radius, initial conditions that affect the trajectory and finally the data transmission rate of the acoustic modem.

This thesis has shown that cross track error and LOS error controllers together with a dynamic waypoints allocation technique if properly tuned, can be effective in steering ARIES into a cage with slow swinging motion. The design has also proven to be fairly robust in different initial conditions. However, this design starts to be less reliable and predictable when the rotation rate of the cage increases. One of the reasons is that as the rotation rate of the cage increases, the AUV cannot keep up with it and the vehicle can end up closer to the cage but in the wrong direction. This will require the vehicle to make a sharp turn around the cage, which in most cases exceeds the vehicle turning radius capability, and thus constitute to a miss attempt.

In addition, this thesis shows that the 3 seconds transmission delay does have a negative impact on the final outcome of the docking. Thus, in the absence of cage data updates, this thesis attempts to use an Euler integration technique to predict the cage heading. The result shows that for scenarios where the docking cage has a low rotation rate, in most cases the Euler integration technique can predict the cage heading with reasonable accuracy and contribute positively to the success of docking.

B. RECOMMENDATIONS

One of the problems encountered in the horizontal steering is that when the cage turns, the current design requires ARIES to make a large steer when it was still a long distance away from the cage in order to keep track with the cage heading. This is demanding for ARIES, which has a limited forward velocity and thus limits the feasibility of the design for a cage with a fast rotation rate. Further research can be done to investigate algorithms that will approach directly the cage and only steer towards the cage entrance when the vehicle is closer to the cage. This optimal distance where such steering begins can depend on the rotation rate of the cage and the cage heading when ARIES reaches it. In addition, more research can be conducted to couple the current design described in this thesis with a speed controller which can regulate the forward velocity of ARIES as it approaches the cage. Finally, besides using just the acoustic modem for data transmission, more researches in other forms of data transmission with higher update rate or guidance method such as electromagnetic guidance [13] can be conducted to resolve the problem of the limitation in the data transfer rate of the acoustic modem.

APPENDIX A: MATLAB FILE MOTION3D.M

```
clear
clc

TRUE  = 1;
FALSE = 0;
check_condition =0;
DegRad = pi/180;
xcg = 5.33;                 % Length from tip of the nose to CG position in X-co-ordinate
                                    (assuming xcg=0.5xLength of Aries
                                % Length of Aries = 128 inches = 10.67ft. xcg=10.67/2 = 5.33ft
% Jay Johnson Model

  m = 222.26;
  U = 1.4*3.28;                 % Forward Speed
  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-(m*U),0;Nv,Nr,0;0,1,0];
  BB = [Ydr;Ndr;0];
  A  = inv(M)*AA;
  B  = inv(M)*BB;

% Below are in British Units for CTE Sliding Mode

Lam1 = 2.0;
Lam2 = 1.0;
cage_length =3/3.28;            % length cone of the cage in meter
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 = [];

Sigma = [];
Depth_com = [];
dr=[];
a = -.3;
b = (9/24)*a;
dt = 0.125;                     % delta T per iteration
```

```
t = [0:dt:1000]';
time = 0;
r_com = 0.0;
SurfaceTime = 25;

% Set initial conditions

v(1)   = 0.0;                          % Initial Side Slip Velocity
r(1)   = 0.0;                          % Initial Yaw
rRM(1) = r(1);
Xpos = input('Initial X position of vehicle in meters (Default:100.0) :'); % Initial Position in meters
if isempty(Xpos),
   X(1)=100.0;
else
   X(1)=Xpos;
end
Ypos = input('Initial Y position of vehicle in meters (Default:0.0) :'); % Initial Position in meters
if isempty(Ypos),
   Y(1)=0.0;
else
   Y(1)=Ypos;
end

psideg =input('Initial vehicle orientation in degree (Default:0.0) :'); % Initial vehicle orientation
if isempty(psideg),
   psi(1)=0.0;
else
   psi(1)=psideg;
end
psi(1) = psi(1)*DegRad;              % Initial Heading of ARIES

W_R(1) = 1.0;                          % Sets initial Watch Radius
x(:,1) = [v(1);r(1);psi(1)];

SurfPhase(1) = 0;
No_tracks =1;
Depth_com(1) = 5.0;
WayPointVertDist_com = 5.0;
SURFACE_TIMER_ACTIVE = FALSE;

Xcage_initial = input('Initial X position of cage in meters (Default:200.0) :'); % Initial Position of
      cage in meters
if isempty(Xcage_initial),
   Xcage_initial=200.0;
end
Ycage_initial = input('Initial Y position of cage in meters (Default:0.0) :'); % Initial Position of cage
      in meters
if isempty(Ycage_initial),
   Ycage_initial=0.0;
end

initial_cage_psi = input('Initial cage orientation in degree from 0-360 (Default:180.0) :'); % Initial
      cage orientation
if isempty(initial_cage_psi),
   initial_cage_psi=180.0;
   initial_cage_psi=mod(initial_cage_psi,360);
```

```
end



% Set start position
X_Way_c(1) = Xcage_initial;      % current waypoint is the cage position
Y_Way_c(1) = Ycage_initial;
PrevX_Way_c(1) = X(1);           % initial waypoint is the vehicle starting point
PrevY_Way_c(1) = Y(1);

%To determine LOS angle
theta = atan((Y(1)-Y_Way_c(1))/(X(1)-X_Way_c(1)))*180/pi;  %LOS angle between initial vehicle
                                                            position and cage position

% Ensure correct LOS angle between initial vehicle position and cage position for different
    conditions
%-----------------------------------------------
if ((X(1)-X_Way_c(1))<0) & ((Y(1)-Y_Way_c(1))<0)
   theta=180+theta;
end
if ((X(1)-X_Way_c(1))>0) & ((Y(1)-Y_Way_c(1))>0)
   theta=theta;
end
if ((X(1)-X_Way_c(1))<0) & ((Y(1)-Y_Way_c(1))>0)
   theta=180+theta;
end
if ((X(1)-X_Way_c(1))>0) & ((Y(1)-Y_Way_c(1))<0)
   theta=360+theta;
end
if ((X(1)-X_Way_c(1))==0) & ((Y(1)-Y_Way_c(1))<0)
   theta=360+theta;
end
if ((X(1)-X_Way_c(1))==0) & ((Y(1)-Y_Way_c(1))>0)
   theta=360+theta;
end
if ((X(1)-X_Way_c(1))>0) & ((Y(1)-Y_Way_c(1))==0)
   theta=360+theta;
end
if ((X(1)-X_Way_c(1))<0) & ((Y(1)-Y_Way_c(1))==0)
   theta=180+theta;
end
%-----------------------------------------------
%LOS distance from initial vehicle position to cage position
Length = sqrt((X_Way_c(1)-X(1))^2+(Y_Way_c(1)-Y(1))^2);
initial_Length = Length;
% To determine the cage orientation
%X co-ordinate of the initial vehicle position from the cage position - for plotting
X_LOS = X_Way_c(1)+initial_Length*cos(theta *DegRad)
%Y co-ordinate of the initial vehicle position from the cage position - for plotting
Y_LOS = Y_Way_c(1)+initial_Length*sin(theta *DegRad)
)% X co-ordinate of centre of cone which is of distance "Length" away from cage position
X_cage = X_Way_c(1)+initial_Length*cos(initial_cage_psi *DegRad
% Y co-ordinate of centre of cone which is of distance "Length" away from cage position
Y_cage = Y_Way_c(1)+initial_Length*sin(initial_cage_psi *DegRad)
% X co-ordinate of left limit of cone of the cage (2D)
L_cone_x = X_Way_c(1)+cage_length*cos((initial_cage_psi +30)*DegRad);
```

```
% Y co-ordinate of left limit of cone of the cage (2D)
L_cone_y = Y_Way_c(1)+cage_length*sin((initial_cage_psi +30)*DegRad);
% X co-ordinate of right limit of cone of the cage (2D)
R_cone_x = X_Way_c(1)+cage_length*cos((initial_cage_psi -30)*DegRad);
% Y co-ordinate of right limit of cone of the cage (2D)
R_cone_y = Y_Way_c(1)+cage_length*sin((initial_cage_psi -30)*DegRad);

%To allow plotting when part of cone co-ordinate equals cage position co-ordinates
%--------------------------------
if ((L_cone_x-X_Way_c(1)) == 0)
   x_lint = 1;
else
   x_lint = L_cone_x-X_Way_c(1);
end
if ((L_cone_y-Y_Way_c(1)) == 0)
   y_lint = 1;
else
   y_lint = L_cone_y-Y_Way_c(1);
end
if ((R_cone_x-X_Way_c(1)) == 0)
   x_rint = 1;
else
   x_rint = R_cone_x-X_Way_c(1);
end
if ((R_cone_y-Y_Way_c(1)) == 0)
   y_rint =1;
else
   y_rint = R_cone_y-Y_Way_c(1);
end
%--------------------------------
% angle between LOS angle and cage initial orientation angle
Diff_angle = theta-initial_cage_psi;

% DeWrap Diff_angle to within +/- pi;  Normalized to Lie between +/- 180
% degrees
while(abs(Diff_angle) > 180)
   Diff_angle = Diff_angle - sign(Diff_angle)*360;
end;

% Compute the number of waypoints and their positions
[X_Way_c,Y_Way_c,SurfPhase,W_R,No_tracks] =
     waypoint2(X_cage,Y_cage,Xcage_initial,Ycage_initial,Diff_angle,initial_cage_psi,Length)

% 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));

% Track Length and track angle for subsequent tracks
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;
for i = 1:length(t)-1,
Depth_com(i) =WayPointVertDist_com;
time = time + dt;

% Cage position updates to the vehicle at every three seconds
if (mod(time,3) == 0)
   % Set initial conditions
   j=1;
   time
   %if time < 50
   initial_cage_psi = 180+3*time;       % Initial cage yaw angle
   %else
   %initial_cage_psi = 180-2*time;
   %end


   % Set start position
   X_Way_c(1) = Xcage_initial;              % Reset waypoint #1 as the cage position
   Y_Way_c(1) = Ycage_initial;
   W_R(1) = 1.0;                            % Sets initial Watch Radius
   SurfPhase(1) = 0;
   No_tracks =1;
   Depth_com(1) = 5.0;
   WayPointVertDist_com = 5.0;
   SURFACE_TIMER_ACTIVE = FALSE;

   PrevX_Way_c(1) = X(i);                   % Set initial waypoint to current vehicle position
   PrevY_Way_c(1) = Y(i);

%To determine LOS angle
%LOS angle from current position of vehicle to waypoint #1(also cage position)
   theta = atan((Y(i)-Y_Way_c(1))/(X(i)-X_Way_c(1)))*180/pi;

% Ensure correct LOS angle between current vehicle position and waypoint #1(also cage
     position) for different conditions
   %---------------------------------------------
   if ((X(i)-X_Way_c(1))<0) & ((Y(i)-Y_Way_c(1))<0)
      theta=180+theta;
   end
   if ((X(i)-X_Way_c(1))>0) & ((Y(i)-Y_Way_c(1))>0)
      theta=theta;
   end
   if ((X(i)-X_Way_c(1))<0) & ((Y(i)-Y_Way_c(1))>0)
      theta=180+theta;
   end
   if ((X(i)-X_Way_c(1))>0) & ((Y(i)-Y_Way_c(1))<0)
      theta=360+theta;
   end
   if ((X(i)-X_Way_c(1))==0) & ((Y(i)-Y_Way_c(1))<0)
      theta=360+theta;
   end
   if ((X(i)-X_Way_c(1))==0) & ((Y(i)-Y_Way_c(1))>0)
      theta=360+theta;
   end
```

```
   if ((X(i)-X_Way_c(1))>0) & ((Y(i)-Y_Way_c(1))==0)
      theta=360+theta;
   end
   if ((X(i)-X_Way_c(1))<0) & ((Y(i)-Y_Way_c(1))==0)
      theta=180+theta;
   end
   %---------------------------------------------
   %LOS distance from current vehicle position to cage position
   Length = sqrt((X_Way_c(1)-X(i))^2+(Y_Way_c(1)-Y(i))^2);

% To determine the cage orientation
%X co-ordinate of the current vehicle position from the cage position - for plotting
   X_LOS = X_Way_c(1)+Length*cos(theta *DegRad);
%Y co-ordinate of the current vehicle position from the cage position - for plotting
   Y_LOS = Y_Way_c(1)+Length*sin(theta *DegRad);

   X_cage = X_Way_c(1)+Length*cos(initial_cage_psi *DegRad);
   Y_cage = Y_Way_c(1)+Length*sin(initial_cage_psi *DegRad);
% X co-ordinate of left limit of cone of the cage (2D)
   L_cone_x = X_Way_c(1)+cage_length*cos((initial_cage_psi +30)*DegRad);
% Y co-ordinate of left limit of cone of the cage (2D)
   L_cone_y = Y_Way_c(1)+cage_length*sin((initial_cage_psi +30)*DegRad);
% X co-ordinate of right limit of cone of the cage (2D)
   R_cone_x = X_Way_c(1)+cage_length*cos((initial_cage_psi -30)*DegRad);
% Y co-ordinate of right limit of cone of the cage (2D)
   R_cone_y = Y_Way_c(1)+cage_length*sin((initial_cage_psi -30)*DegRad);

%To allow plotting when part of cone co-ordinate equals cage position co-ordinates
%------------------------------
   if ((L_cone_x-X_Way_c(1)) == 0)
      x_lint = 1;
   else
      x_lint = L_cone_x-X_Way_c(1);
   end
   if ((L_cone_y-Y_Way_c(1)) == 0)
      y_lint = 1;
   else
      y_lint = L_cone_y-Y_Way_c(1);
   end
   if ((R_cone_x-X_Way_c(1)) == 0)
      x_rint = 1;
   else
      x_rint = R_cone_x-X_Way_c(1);
   end
   if ((R_cone_y-Y_Way_c(1)) == 0)
      y_rint =1;
   else
      y_rint = R_cone_y-Y_Way_c(1);
   end
%--------------------------------

% angle between LOS angle and cage initial orientation angle
    Diff_angle = theta-initial_cage_psi;
% DeWrap Diff_angle to within +/- pi;  Normalized to Lie between +/- 180
% degrees
   while(abs(Diff_angle) > 180)
```

```
      Diff_angle = Diff_angle - sign(Diff_angle)*360;
   end;

% Compute the number of waypoints and their positions
   [X_Way_c,Y_Way_c,SurfPhase,W_R,No_tracks] =
     waypoint2(X_cage,Y_cage,Xcage_initial,Ycage_initial,Diff_angle,initial_cage_psi,Length);

% 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));

% Track Length and track angle for subsequent tracks
   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;
end



% 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


  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;


  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);


 Uc = 0.0;
 Vc = 0.0;

 % Kinematics
```

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

  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 if Aries falls within the cage
  X_tip(i) = X(i) + xcg/3.28*cos(psi(i));       % X co-ordinate of tip of the vehicle's nose
  Y_tip(i) = Y(i) + xcg/3.28*sin(psi(i));       % Y co-ordinate of tip of the vehicle's nose
  %X_centre_cage = Xcage_initial+ cage_length*cos(initial_cage_psi *pi/180);
  %Y_centre_cage = Ycage_initial+ cage_length*sin(initial_cage_psi *pi/180);

%FOV extended to the initial vehicle start point's distance
  L_cone_xFOV = Xcage_initial+initial_Length*cos(mod(initial_cage_psi +30,360)*DegRad);
  L_cone_yFOV = Ycage_initial+initial_Length*sin(mod(initial_cage_psi +30,360)*DegRad);
  R_cone_xFOV = Xcage_initial+initial_Length*cos(mod(initial_cage_psi -30,360)*DegRad);
  R_cone_yFOV = Ycage_initial+initial_Length*sin(mod(initial_cage_psi -30,360)*DegRad);
  Xcurrent = X_tip(i);
  Ycurrent = Y_tip(i);
  psicurrent = psi(i);
%Angle of vehicle tip w.r.t cage position
vehicle_cage_angle = atan2(Y_tip(i)-Ycage_initial,X_tip(i)-Xcage_initial)*180/pi;
% DeWrap vehicle_cage_angle to within +/- pi;  Normalized to Lie between +/- 180
% degrees

   while(abs(vehicle_cage_angle) > 180)
      vehicle_cage_angle = vehicle_cage_angle - sign(vehicle_cage_angle)*360;
   end;
   vehicle_cage_angle = mod(vehicle_cage_angle,360);
   Delta_angle = vehicle_cage_angle-initial_cage_psi; %in degree
   LengthLOS = sqrt((Y_tip(i)-Ycage_initial)^2+(X_tip(i)-Xcage_initial)^2);
   LengthLOSreal = sqrt((Y(i)-Ycage_initial)^2+(X(i)-Xcage_initial)^2);

   maxdistance = cage_length*cos(30*DegRad)
   currentdistance = LengthLOS*cos(Delta_angle*DegRad)
   currentdistancereal = LengthLOSreal*cos(Delta_angle*DegRad)
   mindistance = cage_length*cos(30*DegRad)-0.6

   minangle = mod(initial_cage_psi-30,360)
   vehicle_cage_angle
   maxangle = mod(initial_cage_psi+30,360)

 %Set acceptance criteria tolerances -0.6ft from cage cone length
  %L_cone_xmax = Xcage_initial+(cage_length+(0.3/cos(30*pi/180)))*cos((initial_cage_psi
     +30)*pi/180);
  %L_cone_ymax = Ycage_initial+(cage_length+(0.3/cos(30*pi/180)))*sin((initial_cage_psi
     +30)*pi/180);
  %R_cone_xmax = Xcage_initial+(cage_length+(0.3/cos(30*pi/180)))*cos((initial_cage_psi -
     30)*pi/180);
  %R_cone_ymax = Ycage_initial+(cage_length+(0.3/cos(30*pi/180)))*sin((initial_cage_psi -
     30)*pi/180);

  L_cone_xmin = Xcage_initial+(cage_length-(0.6/cos(30*DegRad)))*cos(mod(initial_cage_psi
     +30,360)*DegRad);
```

```
  L_cone_ymin = Ycage_initial+(cage_length-(0.6/cos(30*DegRad)))*sin(mod(initial_cage_psi
    +30,360)*DegRad);
  R_cone_xmin = Xcage_initial+(cage_length-(0.6/cos(30*DegRad)))*cos(mod(initial_cage_psi -
    30,360)*DegRad);
  R_cone_ymin = Ycage_initial+(cage_length-(0.6/cos(30*DegRad)))*sin(mod(initial_cage_psi -
    30,360)*DegRad);

status =0;
if (mod(initial_cage_psi,360) < 30)| (mod(initial_cage_psi,360) >330)
   if (vehicle_cage_angle>=mod(initial_cage_psi-30,360))|
     (vehicle_cage_angle<=mod(initial_cage_psi+30,360))
      disp(sprintf('Vehicle is within the FOV of cage'));
      status =1;
   end
end

if (vehicle_cage_angle>=mod(initial_cage_psi-30,360)) &
     (vehicle_cage_angle<=mod(initial_cage_psi+30,360))
      disp(sprintf('Vehicle is within the FOV of cage'));
      status =1;
end

if (status == 1)
  R_cone_x
  R_cone_xmin
  if (R_cone_x == L_cone_x)
    if (abs(vehicle_cage_angle) <= 90)|(abs(vehicle_cage_angle) >= 270)
       if (X_tip(i) <= L_cone_x) & (X_tip(i) >= L_cone_xmin)
          disp(sprintf('Vehicle is about to dock into the cage'));
          if ((Y_tip(i) > L_cone_y) | (Y_tip(i) < R_cone_y))
            disp(sprintf('Vehicle misses the cage'));
          else
            disp(sprintf('Vehicle docks into the cage'));
          end
       end
    end
    if (abs(vehicle_cage_angle) > 90)&(abs(vehicle_cage_angle) < 270)
       if (X_tip(i) >= L_cone_x) & (X_tip(i) <= L_cone_xmin)
          disp(sprintf('Vehicle is about to dock into the cage'));
          if ((Y_tip(i) > R_cone_y)|(Y_tip(i) < L_cone_y))
            disp(sprintf('Vehicle misses the cage'));
          else
            disp(sprintf('Vehicle docks into the cage'));
          end
       end
    end

  end


if (R_cone_x ~= L_cone_x)
   if (LengthLOS*cos(Delta_angle*DegRad) <= cage_length*cos(30*DegRad)) & ...
     (LengthLOS*cos(Delta_angle*DegRad) > cage_length*cos(30*DegRad)-0.6)
      disp(sprintf('Vehicle is about to dock into the cage'));
      disp(sprintf('Vehicle docks into the cage'));
   end
```

```
end
else

disp(sprintf('Vehicle is NOT within the FOV of cage'));

end
status =0;


   % Check to See if we are Within the Watch_Radius
   test = j;
   test1 =No_tracks;
   test2 = sqrt(X_Way_Error(i)^2.0 + Y_Way_Error(i)^2.0);
   test3 =X_Way_Error(i);
   test4 = Y_Way_Error(i);
   test5 = s(i);
    if(sqrt(X_Way_Error(i)^2.0 + Y_Way_Error(i)^2.0) <= W_R(j) | s(i) < 0.0)
      disp(sprintf('WayPoint %d Reached',j));
        if(j==No_tracks)
           X_Way_Error(i);
           Y_Way_Error(i);
          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);


 % Planned Track
   for z =1:No_tracks
    plot([X_Way_c(z) PrevX_Way_c(z)],[Y_Way_c(z) PrevY_Way_c(z)],'r');
    hold on
    plot(Xcage_initial,Ycage_initial,'o');
    hold on
    plot(X_Way_c(z),Y_Way_c(z),'*');
    hold on
   end
   %text(0,0,['theta= ' int2str(theta)])
   %hold on

   plot(X_LOS,Y_LOS,'s')
   hold on
   plot(X,Y);

   hold on
   plot(Xcage_initial:x_rint:R_cone_x,Ycage_initial:y_rint:R_cone_y,'c')
   hold on
   plot(Xcage_initial:x_lint:L_cone_x,Ycage_initial:y_lint:L_cone_y,'c')
```

```
  text(Xcage_initial+10,Ycage_initial+40,['Xcage = ' int2str(Xcage_initial)]);
  text(Xcage_initial+10,Ycage_initial+20,['Ycage = ' int2str(Ycage_initial)]);
  text(Xcage_initial+10,Ycage_initial,['X = ' int2str(X(i))]);
  text(Xcage_initial+10,Ycage_initial-20,['Y = ' int2str(Y(i))]);
  hold on
  if (Xcage_initial> X(1)) & (Ycage_initial> Y(1))
  Axis([X(1)-100 Xcage_initial+100 Y(1)-100 Ycage_initial+100]);
  end
  if (Xcage_initial< X(1)) & (Ycage_initial< Y(1))
  Axis([Xcage_initial-100 X(1)+100 Ycage_initial-100 Y(1)+100]);
  end
  if (Xcage_initial< X(1)) & (Ycage_initial> Y(1))
  Axis([Xcage_initial-100 X(1)+100 Y(1)-100 Ycage_initial+100]);
  end
  if (Xcage_initial> X(1)) & (Ycage_initial< Y(1))
  Axis([X(1)-100 Xcage_initial+100 Ycage_initial-100 Y(1)+100]);
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: MATLAB FILE WAYPOINT2.M

```
function [X_Way_c,Y_Way_c,SurfPhase,W_R,No_tracks] =
     waypoint(X_cage,Y_cage,Xcage_initial,Ycage_initial,Diff_angle,initial_cage_psi,Length)
SurfPhase =0.0;
if (Diff_angle>=30)
   if ((Diff_angle>=30) & (Diff_angle<=60))
      X_Way_c(1) = X_cage;
      Y_Way_c(1) = Y_cage;
      X_Way_c(2) = Xcage_initial;
      Y_Way_c(2) = Ycage_initial;
      No_tracks =2;
      W_R(2) =5;
      SurfPhase(2) = 0;
   end
   if ((Diff_angle>60) & (Diff_angle<=90))
      No_tracks =3;
   end
   if ((Diff_angle>90) & (Diff_angle<=120))
      No_tracks =4;
   end
   if ((Diff_angle>120) & (Diff_angle<=150))
      No_tracks =5;
   end
   if ((Diff_angle>150) & (Diff_angle<=180))
      No_tracks =6;
   end
   for z = 2:No_tracks-1
      W_R(z) =10.0-z;
   end
   W_R(No_tracks) =1.0;
   for z = 2:No_tracks
     SurfPhase(z) = 0;
   end

   for z =1:No_tracks-1
      if ((No_tracks>=3) & (z<=No_tracks-3))
         X_cage = Xcage_initial+Length*cos((initial_cage_psi+(No_tracks*30-30)-z*30) *pi/180);
         Y_cage = Ycage_initial+Length*sin((initial_cage_psi+(No_tracks*30-30)-z*30) *pi/180);
      else
         %X_cage = Xcage_initial+Length/2*cos((initial_cage_psi+(No_tracks*30-30)-z*30)
    *pi/180);
         %Y_cage = Ycage_initial+Length/2*sin((initial_cage_psi+(No_tracks*30-30)-z*30)
    *pi/180);
         X_cage = Xcage_initial+Length*cos((initial_cage_psi+(No_tracks*30-30)-z*30) *pi/180);
         Y_cage = Ycage_initial+Length*sin((initial_cage_psi+(No_tracks*30-30)-z*30) *pi/180);
      end

      X_Way_c(z) = X_cage;
      Y_Way_c(z) = Y_cage;
      Length = sqrt((Xcage_initial-X_Way_c(z))^2+(Ycage_initial-Y_Way_c(z))^2);
   end
   X_Way_c(No_tracks) = Xcage_initial;
```

```
   Y_Way_c(No_tracks) = Ycage_initial;
end

if (Diff_angle<=-30)

   if ((Diff_angle<=-30) & (Diff_angle>=-60))
      X_Way_c(1) = X_cage;
      Y_Way_c(1) = Y_cage;
      X_Way_c(2) = Xcage_initial;
      Y_Way_c(2) = Ycage_initial;
      No_tracks =2;
      W_R(2) =1;
      SurfPhase(2) = 0;
   end

   if ((Diff_angle<-60) & (Diff_angle>=-90))
      No_tracks =3;
   end
   if ((Diff_angle<-90) & (Diff_angle>=-120))
      No_tracks =4;
   end
   if ((Diff_angle<-120) & (Diff_angle>=-150))
      No_tracks =5;
   end
   if ((Diff_angle<-150) & (Diff_angle>-180))
      No_tracks =6;
   end
   for z = 2:No_tracks-1
      W_R(z) =10.0-z;
   end
   W_R(No_tracks) =1.0;
   for z = 2:No_tracks
      SurfPhase(z) = 0;
   end
   for z =1:No_tracks-1
      if ((No_tracks>=3) & (z<=No_tracks-3))
      X_cage = Xcage_initial+Length*cos((initial_cage_psi-(No_tracks*30-30)+z*30)*pi/180);
      Y_cage = Ycage_initial+Length*sin((initial_cage_psi-(No_tracks*30-30)+z*30)*pi/180);
      else
      X_cage = Xcage_initial+Length*cos((initial_cage_psi-(No_tracks*30-30)+z*30)*pi/180);
      Y_cage = Ycage_initial+Length*sin((initial_cage_psi-(No_tracks*30-30)+z*30)*pi/180);
      %X_cage = Xcage_initial+Length/2*cos((initial_cage_psi-(No_tracks*30-30)+z*30)*pi/180);
      %Y_cage = Ycage_initial+Length/2*sin((initial_cage_psi-(No_tracks*30-30)+z*30)*pi/180);

      end
      X_Way_c(z) = X_cage;
      Y_Way_c(z) = Y_cage;
      Length = sqrt((Xcage_initial-X_Way_c(z))^2+(Ycage_initial-Y_Way_c(z))^2);
   end
X_Way_c(No_tracks) = Xcage_initial;
Y_Way_c(No_tracks) = Ycage_initial;

end

if ((Diff_angle>-30) & (Diff_angle<30))
No_tracks = 1;
```

```
X_Way_c(No_tracks) = Xcage_initial;
Y_Way_c(No_tracks) = Ycage_initial;
end

if (No_tracks < 2)
   W_R(1) =1.0;
else
   W_R(1) =10.0;
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: MATLAB FILE DELAY2DEG.M

```
clear
clc
num=0;
TRUE  = 1;
FALSE = 0;
direction=0;                        %initialise the direction of cage motion
check_condition =0;
DegRad = pi/180;
% Length from tip of the nose to CG position in X-co-ordinate (assuming xcg=0.5xLength of Aries
xcg = 5.33;
% Length of Aries = 128 inches = 10.67ft. xcg=10.67/2 = 5.33ft
% Jay Johnson Model

  m = 222.26;
  U = 1.9*3.28;              % Forward Speed
  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-(m*U),0;Nv,Nr,0;0,1,0];
  BB = [Ydr;Ndr;0];
  A  = inv(M)*AA;
  B  = inv(M)*BB;

% Below are in British Units for CTE Sliding Mode

Lam1 = 2.0;
Lam2 = 1.0;
cage_length =1;               % length cone of the cage in meter
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 = [];

Sigma = [];
Depth_com = [];
dr=[];
a = -.3;
b = (9/24)*a;
```

```
dt = 0.05;                    % delta T per iteration
t = [0:dt:550]';
time = 0;
r_com = 0.0;
SurfaceTime = 25;

% Set initial conditions

v(1)   = 0.0;                  % Initial Side Slip Velocity
r(1)   = 0.0;                  % Initial Yaw
rRM(1) = r(1);
Xpos = input('Initial X position of vehicle in meters (Default:100.0) :'); % Initial Position in meters
if isempty(Xpos),
   X(1)=100.0;
else
   X(1)=Xpos;
end
Ypos = input('Initial Y position of vehicle in meters (Default:0.0) :'); % Initial Position in meters
if isempty(Ypos),
   Y(1)=0.0;
else
   Y(1)=Ypos;
end

psideg =input('Initial vehicle orientation in degree (Default:0.0) :'); % Initial vehicle orientation
if isempty(psideg),
   psi(1)=0.0;
else
   psi(1)=psideg;
end
psi(1) = psi(1)*DegRad;        % Initial Heading of ARIES

W_R(1) = 1.0;                  % Sets initial Watch Radius
x(:,1) = [v(1);r(1);psi(1)];

SurfPhase(1) = 0;
No_tracks =1;
Depth_com(1) = 5.0;
WayPointVertDist_com = 5.0;
SURFACE_TIMER_ACTIVE = FALSE;

Xcage_initial = input('Initial X position of cage in meters (Default:200.0) :'); % Initial Position of
    cage in meters
if isempty(Xcage_initial),
   Xcage_initial=200.0;
end
Ycage_initial = input('Initial Y position of cage in meters (Default:0.0) :'); % Initial Position of cage
    in meters
if isempty(Ycage_initial),
   Ycage_initial=0.0;
end

initial_cage_psi = input('Initial cage orientation in degree from 0-360 (Default:180.0) :'); % Initial
    cage orientation
if isempty(initial_cage_psi),
```

```
   initial_cage_psi=180.0;
   initial_cage_psi=mod(initial_cage_psi,360);
end



% Set start position
% current waypoint is the cage position
X_Way_c(1) = Xcage_initial;
Y_Way_c(1) = Yca ge_initial;
% initial waypoint is the vehicle starting point
PrevX_Way_c(1) = X(1);
PrevY_Way_c(1) = Y(1);
% initialise the cage angular sensor
cage_psi_old = initial_cage_psi;

%To determine LOS angle
%LOS angle between initial vehicle position and cage position
theta = atan((Y(1)-Y_Way_c(1))/(X(1)-X_Way_c(1)))*180/pi;

% Ensure correct LOS angle between initial vehicle position and cage position for different
     conditions
%-----------------------------------------------
if ((X(1)-X_Way_c(1))<0) & ((Y(1)-Y_Way_c(1))<0)
   theta=180+theta;
end
if ((X(1)-X_Way_c(1))>0) & ((Y(1)-Y_Way_c(1))>0)
   theta=theta;
end
if ((X(1)-X_Way_c(1))<0) & ((Y(1)-Y_Way_c(1))>0)
   theta=180+theta;
end
if ((X(1)-X_Way_c(1))>0) & ((Y(1)-Y_Way_c(1))<0)
   theta=360+theta;
end
if ((X(1)-X_Way_c(1))==0) & ((Y(1)-Y_Way_c(1))<0)
   theta=360+theta;
end
if ((X(1)-X_Way_c(1))==0) & ((Y(1)-Y_Way_c(1))>0)
   theta=360+theta;
end
if ((X(1)-X_Way_c(1))>0) & ((Y(1)-Y_Way_c(1))==0)
   theta=360+theta;
end
if ((X(1)-X_Way_c(1))<0) & ((Y(1)-Y_Way_c(1))==0)
   theta=180+theta;
end
%-----------------------------------------------
%LOS distance from initial vehicle position to cage position
Length = sqrt((X_Way_c(1)-X(1))^2+(Y_Way_c(1)-Y(1))^2);
initial_Length = Length;
% To determine the cage orientation
%X co-ordinate of the initial vehicle position from the cage position - for plotting
X_LOS = X_Way_c(1)+initial_Length*cos(theta *DegRad);
%Y co-ordinate of the initial vehicle position from the cage position - for plotting
Y_LOS = Y_Way_c(1)+initial_Length*sin(theta *DegRad);
```

```
% X co-ordinate of centre of cone which is of distance "Length" away from cage position
X_cage = X_Way_c(1)+initial_Length*cos(initial_cage_psi *DegRad);
% Y co-ordinate of centre of cone which is of distance "Length" away from cage position
Y_cage = Y_Way_c(1)+initial_Length*sin(initial_cage_psi *DegRad);
% X co-ordinate of left limit of cone of the cage (2D)
L_cone_x = X_Way_c(1)+cage_length*cos((initial_cage_psi +30)*DegRad);
% Y co-ordinate of left limit of cone of the cage (2D)
L_cone_y = Y_Way_c(1)+cage_length*sin((initial_cage_psi +30)*DegRad);
% X co-ordinate of right limit of cone of the cage (2D)
R_cone_x = X_Way_c(1)+cage_length*cos((initial_cage_psi -30)*DegRad);
% Y co-ordinate of right limit of cone of the cage (2D)
R_cone_y = Y_Way_c(1)+cage_length*sin((initial_cage_psi -30)*DegRad);

%To allow plotting when part of cone co-ordinate equals cage position co-ordinates
%--------------------------------
if ((L_cone_x-X_Way_c(1)) == 0)
   x_lint = 1;
else
   x_lint = L_cone_x-X_Way_c(1);
end
if ((L_cone_y-Y_Way_c(1)) == 0)
   y_lint = 1;
else
   y_lint = L_cone_y-Y_Way_c(1);
end
if ((R_cone_x-X_Way_c(1)) == 0)
   x_rint = 1;
else
   x_rint = R_cone_x-X_Way_c(1);
end
if ((R_cone_y-Y_Way_c(1)) == 0)
   y_rint =1;
else
   y_rint = R_cone_y-Y_Way_c(1);
end
%--------------------------------
% angle between LOS angle and cage initial orientation angle
Diff_angle = theta-initial_cage_psi;

% DeWrap Diff_angle to within +/- pi;  Normalized to Lie between +/- 180
% degrees
while(abs(Diff_angle) > 180)
   Diff_angle = Diff_angle - sign(Diff_angle)*360;
end;

% Compute the number of waypoints and their positions
[X_Way_c,Y_Way_c,SurfPhase,W_R,No_tracks]                                                    =
     waypoint3(X_cage,Y_cage,Xcage_initial,Ycage_initial,Diff_angle,initial_cage_psi,Length)

% 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));
```

```
% Track Length and track angle for subsequent tracks
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;

cage_rate =2;                                          %Cage rotation rate
cage_psi = initial_cage_psi;
j=1;
for i = 1:length(t)-1,
Depth_com(i) =WayPointVertDist_com;
time = time + dt;
% Cage position updates to the vehicle at every three seconds
%Real cage orientation at any point in time
real_cage_psi= 60*sin(cage_rate*pi/180*time)+initial_cage_psi;

% determine the direction of rotation: direction=1 ==> anti-clockwise
if (direction == 1)
estimate_cage_psi = cage_rate*dt+cage_psi;
else
estimate_cage_psi = -cage_rate*dt+cage_psi;
end

% sensor update from the cage every 3 seconds
if ((round(time)-time)<1e-6) & ((round(time)-time)>-1e-6) & (mod(round(time),3) == 0)
      cage_psi = real_cage_psi;
      cage_psi_old;
      cage_rate = 2                        % cage rate measurement from cage
% determine the direction of rotation: direction=1 ==> anti-clockwise
      if (cage_psi>cage_psi_old)
         direction = 1;
      else
         direction = 0;
      end
%cage_psi = initial_cage_psi+2*time      % Initial cage yaw angle
      estimate_cage_psi = cage_psi;
      cage_psi_old = cage_psi;
end

   % Set initial conditions
   j=1;
   time;
   % Set start position
   X_Way_c(1) = Xcage_initial;            % Reset waypoint #1 as the cage position
   Y_Way_c(1) = Ycage_initial;
   W_R(1) = 1.0;                          % Sets initial Watch Radius
   SurfPhase(1) = 0;
   No_tracks =1;
   Depth_com(1) = 5.0;
   WayPointVertDist_com = 5.0;
   SURFACE_TIMER_ACTIVE = FALSE;

   PrevX_Way_c(1) = X(i);            % Set initial waypoint to current vehicle position
   PrevY_Way_c(1) = Y(i);
```

```
   %To determine LOS angle
   %LOS angle from current position of vehicle to waypoint #1(also cage position)
theta = atan((Y(i)-Y_Way_c(1))/(X(i)-X_Way_c(1)))*180/pi;

   % Ensure correct LOS angle between current vehicle position and waypoint #1(also cage
     position) for different conditions
   %-----------------------------------------------
   if ((X(i)-X_Way_c(1))<0) & ((Y(i)-Y_Way_c(1))<0)
      theta=180+theta;
   end
   if ((X(i)-X_Way_c(1))>0) & ((Y(i)-Y_Way_c(1))>0)
      theta=theta;
   end
   if ((X(i)-X_Way_c(1))<0) & ((Y(i)-Y_Way_c(1))>0)
      theta=180+theta;
   end
   if ((X(i)-X_Way_c(1))>0) & ((Y(i)-Y_Way_c(1))<0)
      theta=360+theta;
   end
   if ((X(i)-X_Way_c(1))==0) & ((Y(i)-Y_Way_c(1))<0)
      theta=360+theta;
   end
   if ((X(i)-X_Way_c(1))==0) & ((Y(i)-Y_Way_c(1))>0)
      theta=360+theta;
   end
   if ((X(i)-X_Way_c(1))>0) & ((Y(i)-Y_Way_c(1))==0)
      theta=360+theta;
   end
   if ((X(i)-X_Way_c(1))<0) & ((Y(i)-Y_Way_c(1))==0)
      theta=180+theta;
   end
%-----------------------------------------------
%LOS distance from current vehicle position to cage position
   Length = sqrt((X_Way_c(1)-X(i))^2+(Y_Way_c(1)-Y(i))^2);

 To determine the cage orientation
%X co-ordinate of the current vehicle position from the cage position - for plotting
   X_LOS = X_Way_c(1)+Length*cos(theta *DegRad);
%Y co-ordinate of the current vehicle position from the cage position - for plotting
   Y_LOS = Y_Way_c(1)+Length*sin(theta *DegRad);

   X_cage = X_Way_c(1)+Length*cos(estimate_cage_psi *DegRad);
   Y_cage = Y_Way_c(1)+Length*sin(estimate_cage_psi *DegRad);
% X co-ordinate of left limit of cone of the cage (2D)
   L_cone_x = X_Way_c(1)+cage_length*cos((real_cage_psi +30)*DegRad);
% Y co-ordinate of left limit of cone of the cage (2D)
   L_cone_y = Y_Way_c(1)+cage_length*sin((real_cage_psi +30)*DegRad);
% X co-ordinate of right limit of cone of the cage (2D)
   R_cone_x = X_Way_c(1)+cage_length*cos((real_cage_psi -30)*DegRad);
% Y co-ordinate of right limit of cone of the cage (2D)
   R_cone_y = Y_Way_c(1)+cage_length*sin((real_cage_psi -30)*DegRad);

   %To allow plotting when part of cone co-ordinate equals cage position co-ordinates
   %------------------------------
   if ((L_cone_x-X_Way_c(1)) == 0)
```

```
      x_lint = 1;
   else
      x_lint = L_cone_x-X_Way_c(1);
   end
   if ((L_cone_y-Y_Way_c(1)) == 0)
      y_lint = 1;
   else
      y_lint = L_cone_y-Y_Way_c(1);
   end
   if ((R_cone_x-X_Way_c(1)) == 0)
      x_rint = 1;
   else
      x_rint = R_cone_x-X_Way_c(1);
   end
   if ((R_cone_y-Y_Way_c(1)) == 0)
      y_rint =1;
   else
      y_rint = R_cone_y-Y_Way_c(1);
   end
%--------------------------------
% angle between LOS angle and cage initial orientation angle
   Diff_angle = theta-estimate_cage_psi;
% DeWrap Diff_angle to within +/- pi;  Normalized to Lie between +/- 180
% degrees
   while(abs(Diff_angle) > 180)
      Diff_angle = Diff_angle - sign(Diff_angle)*360;
   end;

% Compute the number of waypoints and their positions
   [X_Way_c,Y_Way_c,SurfPhase,W_R,No_tracks]                                                    =
     waypoint3(X_cage,Y_cage,Xcage_initial,Ycage_initial,Diff_angle,estimate_cage_psi,Length);

% 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));

% Track Length and track angle for subsequent tracks
   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;

% 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



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


   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);
```

```
 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 if Aries falls within the cage
 X_tip(i) = X(i) + xcg/3.28*cos(psi(i));       % X co-ordinate of tip of the vehicle's nose
 Y_tip(i) = Y(i) + xcg/3.28*sin(psi(i));       % Y co-ordinate of tip of the vehicle's nose

%FOV extended to the initial vehicle start point's distance

 Xcurrent = X_tip(i);
 Ycurrent = Y_tip(i);
 psicurrent = psi(i);
%Angle of vehicle tip w.r.t cage position
 vehicle_cage_angle = atan2(Y_tip(i)-Ycage_initial,X_tip(i)-Xcage_initial)*180/pi;
% DeWrap vehicle_cage_angle to within +/- pi;  Normalized to Lie between +/- 180
% degrees

  while(abs(vehicle_cage_angle) > 180)
     vehicle_cage_angle = vehicle_cage_angle - sign(vehicle_cage_angle)*360;
  end;
  vehicle_cage_angle = mod(vehicle_cage_angle,360);


  Delta_angle = vehicle_cage_angle-real_cage_psi; %in degree
  LengthLOS = sqrt((Y_tip(i)-Ycage_initial)^2+(X_tip(i)-Xcage_initial)^2);
  LengthLOSreal = sqrt((Y(i)-Ycage_initial)^2+(X(i)-Xcage_initial)^2);

  maxdistance = cage_length*cos(30*DegRad);
  currentdistance = LengthLOS*cos(Delta_angle*DegRad);
  currentdistancereal = LengthLOSreal*cos(Delta_angle*DegRad);
  mindistance = cage_length*cos(30*DegRad)-0.3125;

  minangle = mod(real_cage_psi-30,360);
  vehicle_cage_angle;
  maxangle = mod(real_cage_psi+30,360);

  num=num+1;
  vca(num) =vehicle_cage_angle;
  time1(num) = time;
  psi1(num) = psi(i);
  delta(num) = Delta_angle;
  diff(num) =Diff_angle;

%Set acceptance criteria tolerances -0.6ft from cage cone length

 L_cone_xmin  =  Xcage_initial+(cage_length-(0.3125/cos(30*DegRad)))*cos(mod(real_cage_psi
    +30,360)*DegRad);
 L_cone_ymin  =  Ycage_initial+(cage_length-(0.3125/cos(30*DegRad)))*sin(mod(real_cage_psi
    +30,360)*DegRad);
```

```
  R_cone_xmin = Xcage_initial+(cage_length-(0.3125/cos(30*DegRad)))*cos(mod(real_cage_psi
    -30,360)*DegRad);
  R_cone_ymin = Ycage_initial+(cage_length-(0.3125/cos(30*DegRad)))*sin(mod(real_cage_psi
    -30,360)*DegRad);

status =0;
if (mod(real_cage_psi,360) < 30)| (mod(real_cage_psi,360) >330)
   if                    (vehicle_cage_angle>=mod(real_cage_psi-30,360))                    |
     (vehicle_cage_angle<=mod(real_cage_psi+30,360))
      disp(sprintf('Vehicle is within the FOV of cage'));
      status =1;
   end
end

if                    (vehicle_cage_angle>=mod(real_cage_psi-30,360))                    &
     (vehicle_cage_angle<=mod(real_cage_psi+30,360))
      disp(sprintf('Vehicle is within the FOV of cage'));
      status =1;
end

if (status == 1)
  R_cone_x;
  R_cone_xmin;
  if (R_cone_x == L_cone_x)
    if (abs(vehicle_cage_angle) <= 90)|(abs(vehicle_cage_angle) >= 270)
       if (X_tip(i) <= L_cone_x) & (X_tip(i) >= L_cone_xmin)
          disp(sprintf('Vehicle is about to dock into the cage'));
          if ((Y_tip(i) > L_cone_y) | (Y_tip(i) < R_cone_y))
             disp(sprintf('Vehicle misses the cage'));
          else
             disp(sprintf('Vehicle docks into the cage'));
             break;
          end
       end
    end
    if (abs(vehicle_cage_angle) > 90)&(abs(vehicle_cage_angle) < 270)
       if (X_tip(i) >= L_cone_x) & (X_tip(i) <= L_cone_xmin)
          disp(sprintf('Vehicle is about to dock into the cage'));
          if ((Y_tip(i) > R_cone_y)|(Y_tip(i) < L_cone_y))
             disp(sprintf('Vehicle misses the cage'));
          else
             disp(sprintf('Vehicle docks into the cage'));
             break;
          end
       end
    end

  end

if (R_cone_x ~= L_cone_x)
   if (LengthLOS*cos(Delta_angle*DegRad) <= cage_length*cos(30*DegRad)) & ...
     (LengthLOS*cos(Delta_angle*DegRad) > cage_length*cos(30*DegRad)-0.3125)
      disp(sprintf('Vehicle is about to dock into the cage'));
      disp(sprintf('Vehicle docks into the cage'));
      break;
   end
```

```
end
else

disp(sprintf('Vehicle is NOT within the FOV of cage'));

end
status =0;

   % Check to See if we are Within the Watch_Radius
   test = j;
   test1 =No_tracks;
   test2 = sqrt(X_Way_Error(i)^2.0 + Y_Way_Error(i)^2.0);
   test3 =X_Way_Error(i);
   test4 = Y_Way_Error(i);
   test5 = s(i);
    if(sqrt(X_Way_Error(i)^2.0 + Y_Way_Error(i)^2.0) <= W_R(j) | s(i) < 0.0)
      disp(sprintf('WayPoint %d Reached',j));
        if(j==No_tracks)
            X_Way_Error(i);
            Y_Way_Error(i);
          break;
        end;
      PrevX_Way_c(j+1) = X_Way_c(j);
      PrevY_Way_c(j+1) = Y_Way_c(j);
      j=j+1;
  end;

cage_psi=estimate_cage_psi;
end;

   plot(X_LOS,Y_LOS,'s')
   hold on
   plot(X,Y);
   hold on
   plot(Xcage_initial:x_rint:R_cone_x,Ycage_initial:y_rint:R_cone_y,'r')
   hold on
   plot(Xcage_initial:x_lint:L_cone_x,Ycage_initial:y_lint:L_cone_y,'r')
   hold on
   plot([X_tip(i)],[Y_tip(i)],'k.');
   hold on

   if (Xcage_initial> X(1)) & (Ycage_initial> Y(1))
   Axis([X(1)-100 Xcage_initial+100 Y(1)-100 Ycage_initial+100]);
   end
   if (Xcage_initial< X(1)) & (Ycage_initial< Y(1))
   Axis([Xcage_initial-100 X(1)+100 Ycage_initial-100 Y(1)+100]);
   end
   if (Xcage_initial< X(1)) & (Ycage_initial> Y(1))
   Axis([Xcage_initial-100 X(1)+100 Y(1)-100 Ycage_initial+100]);
   end
   if (Xcage_initial> X(1)) & (Ycage_initial< Y(1))
   Axis([X(1)-100 Xcage_initial+100 Ycage_initial-100 Y(1)+100]);
                                                  end
```

LIST OF REFERENCES

[1] Stokey, R., M. Purcell, N. Forrester, T. Austin, R. Goldsborough, B. Allen and C. von Alt. 1997. "A Docking System for REMUS, an Autonomous Underwater Vehicle." Proceedings from the IEEE Oceans '97 Conference, Vol. II. Halifax, Nov Scotia, Canada, October 6-9, 1997, pp 1132-1136.

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

[3] 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.

[4] John J. Keegan, "Trajectory Planning For the ARIES AUV," M.S Thesis Naval Postgraduate School, Monterey, CA, June 2002.

[5] Joseph J. Keller, "Tracking Control of Autonomous Underwater Vehicles," M.S Thesis Naval Postgraduate School, Monterey, CA, June 2002.

[6] William J. Marr, "Acoustic Based Tactical Control of Underwater Vehicles," P.H.D Dissertation Naval Postgraduate School, Monterey, CA, June 2003.

[7] Franklin G.F, Powell, Emami-Naeini. Feedback Control of Dynamic Systems.4th Edition New Jersey:Prentice Hall 2002.

[8] Franklin G.F., Powell. Digital Control of Dynamic Systems Massachusetts: Addisson Wesley 1980.

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

[10] 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.

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

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

[13] Michael D. Feezor, Paul R. Blankinship, James G. Bellingham, F. Yates Sorrell, "Autonomous Underwater Vehicle Homing/Docking Via Electromagnetic Guidance." Proceedings from the IEEE Oceans '97 Conference, Vol. II. Halifax, Nov Scotia, Canada, October 6-9, 1997, pp 1132-1136.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Anthony J. Healey
   Chairman, Department of Mechanical and Astronautical Engineering
   Naval Postgraduate School
   Monterey, California

4. Professor Fotis Papoulias
   Naval Postgraduate School
   Monterey, California

5. Professor Yeo Tat Soon
   Director, Temasek Defence System Institute
   National University of Singapore
   Singapore