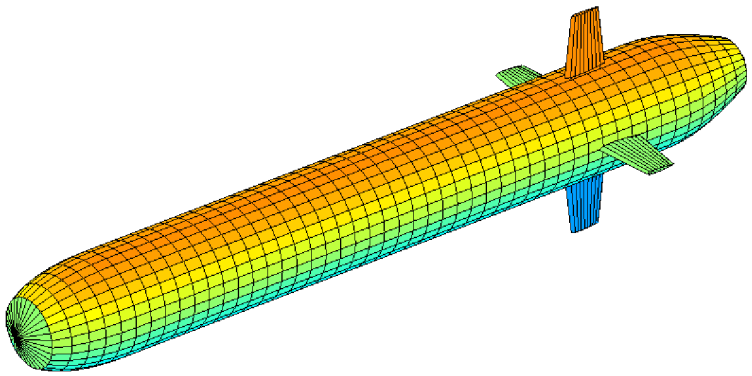
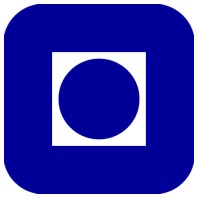
Hydrodynamic estimation and identification



Msc Thesis by:
Håvard Bø



Norwegian University of Science and Technology, NTNU
Department of Engineering Cybernetics
Trondheim, Norway

2nd August 2004

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Engineering Cybernetics

THESIS DESCRIPTION SHEET

Name of the Candidate: *Håvard Bø*

Field of Study: *Engineering Cybernetics*

Thesis Title (Norwegian): *Estimering og identifikasjon av hydrodynamiske koeffisienter*

Thesis Title (English): *Hydrodynamic Estimation and Identification*

Thesis Description: The candidate will consider the hydrodynamic parameter estimation problem and use an autonomous underwater vehicle (AUV) as case study. The following elements must be considered:

1. Hydrodynamic modeling of an AUV
2. Study the theory of adaptation of estimation techniques commonly used in the aviation industry. This should include the USAF Stability and Control program: DATCOM. The goal is to apply these methods to marine vehicles.
3. Develop a software package for estimation of hydrostatic and dynamic derivatives for a submerged vehicle.
4. Use commercial software for estimation of hydrodynamic properties like WAMIT and use MAYA in the case study. MAYA is an AUV which is currently being developed under a joint Indian-Portuguese project.
5. Compare the results from *ad-hoc-methods*, hydrodynamics software programs, and DATCOM, and present the results in your thesis.

Start date: *2004-02-02*
Due date: *2004-08-02*

Thesis performed at: *Department of Engineering Cybernetics, NTNU*
ISR/IST, University of Lisbon

Supervisor: *Professor Thor I. Fossen, NTNU*
Co-Advisors: *Professor António Pascoal, ISR/IST, University of Lisbon*
Professor Ettore A. Barros, University of São Paulo

Trondheim, 2004-02-02

Thor I. Fossen
Professor / Supervisor

Preface

With this document, five years of studies comes to an end. Five years? It certainly does not feel that long. What happened? What did we do? Moreover, what are we going to do? The series of questions are endless; and so are the series of answers, only that some of them are not known yet...

I remember when I first started at NTNU, these simple words of wisdom saying:

"...havet er stort,
ikke kødd med havet..."

or in English *"the ocean is huge, don't bullshit the ocean"*. Simple words, but after five years of study I still remember them. Why? It is hard to say, but during the last few years, my respect for the ocean has increased. Why? the last two years, I have seen that there are so many matters, regarding the ocean, that we do not know how to control. What... The series of questions are endless, and the solutions are yet to determine. I am looking forward to it...

Acknowledgment

I would first and foremost like to thank my supervisor at NTNU, Professor Thor Inge Fossen for being my supervisor and for teaching me the subject guidance and control. I would also like to thanks Professor António Pascoal (ISR/IST) and Professor Ettore A. Barros (University of São Paulo) for their kindness and help during my stay at ISR. They have all been great inspirational sources, and have supported me during the creation of this document.

Secondly, I would like to thank Dr. Peter Ridley (Queensland University of Technology) for providing me additional data and drawings belonging to the work of Ridley, Fontan & Corke (2003). Without his help, it would be difficult to verify the output from the MATLAB ToolBox created during this work. Ann Johanne Bjørge, at the Marine library at NTNU, thank you very much for providing me reports and articles to the other end of Europe. Last, but not least, I would like to thanks my friend and fellow student, Åsmund Hjulstad, for our mutual discussions regarding marine control and applications.

Håvard Bø
August 2004

Abstract

This thesis addresses the problem of estimating the dynamics of an autonomous underwater vehicle (AUV). The theory presented is not limited to a specific design, but an autonomous underwater vehicle that is being developed under a joint Indian-Portuguese project, MAYA, will be used as an example. In order to obtain the dynamic equations of motion, we make use of empirical methods to obtain the hydrodynamic derivatives for a body of revolution. The USAF Stability and Control Datcom, a data compendium developed to determine the analogous aerodynamic stability derivatives for aircrafts, is used as a tool for obtaining estimates of the non-dimensional derivatives in an incompressible medium. The theory and equations involved are rewritten into a standard form for underwater vehicles.

A MATLAB Toolbox, hde (short for HydroDynamic Estimation), is developed. The toolbox is capable of estimating static and dynamic derivatives in the horizontal and vertical plane, in addition to added mass in six degrees of freedom. A nonlinear simulation environment which retains all the nonlinearities inherent in the coupled dynamics equations of motion, as well as those inherent to the hydrodynamic relations which govern the forces acting on the hull and control planes, is also implemented as a part of the toolbox.

The results obtained from hde are compared with experimental data, where simulations in hde indicate that lift and turning rates are comparable with experimental data. The same yields for the added mass terms, which are very close to data reported from experimental results.

Estimation of added mass for an AUV with the aid of WAMIT, a commercial hydrodynamic computation program, has been executed. The result from these experiments reveals that WAMIT fails to accurately estimate the added mass contribution from the low aspect-ratio foils involved in a typical AUV design.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The design and development of an autonomous underwater vehicle (AUV) is a complex and expensive task. If the designer relies exclusively on prototype testing to develop the vehicle's geometry and controllers, the process can be lengthy and poses the additional risk of prototype loss. Each design iteration involves changes to the prototype vehicle, and may take considerably amount of time. By using mathematical models it is possible to estimate expected performance between different designs, and determine the inherent stability of a proposed design before the prototype is manufactured. Further-more, by using custom made software, it is possible to automatically obtain these mathematical models and compare different designs with *"a few mouse clicks"*.

1.1 Motivation

At the end of last semester, fall 2003, my supervisor at NTNU, Professor Thor Inge Fossen, offered me the opportunity to write my thesis as a visiting student at "Instituto de Sistemas e Robótica" (ISR) at the university "Instituto Superior Tecnico" (IST) in Lisbon, Portugal; an offer I simply could not refuse. During the past few years, ISR has carried out some very interesting projects and research in the field of dynamical systems applied to the design and operation of autonomous marine and aerial robots.

When I first arrived in Lisbon, it was decided that I should take part in the work of Mr. Ettore Barros[1], addressing the problem of autonomous underwater vehicle (AUV) modelling as a mean to predict the expected dynamic performance. There exists a number of methods for determination of the hydrodynamic coefficients used in the mathematical model which quantify the forces acting on the body as a function of its attitude and motion. Broadly, they can be broken down into *test-based* and *predictive* methods, where the former include direct experimental determination based on wind-tunnel or tow-tank experiments (Goheen 1991). A less direct, but perhaps more efficient test-based method, is system identification techniques, which can be applied to both free-swimming models and full-size vehicle tests (Pepijn, Johansen, Sørensen,

[1] a Brazilian professor from the University of Sao Paulo, visiting ISR on his sabbatical year

Flanagan & Toal 2004, Pereira & Duncan 2000). An overriding disadvantage of the above methods is the need for a vehicle, as well as laboratory or in field testing facilities. These are often not available, either for reasons of cost or, simply, because the vehicle has not yet been constructed.

Predictive methods offer an attractive alternative to test-based methods when the vehicle is still in the design stage (Nahon 1996). Using predictive methods, the vehicle overall design can for instance be determined by comparing expected characteristic for different vehicles, based on a optimal design criteria (Barros, Pascóal & de Sa 2004). Further- more, from the control point of view it may be advantageous if we could automatically obtain an "accurate" dynamic model of the vehicle; giving the opportunity to focus on the controllers instead of the time consuming, and sometimes tedious, work of designing a "plant" model (as a former instructor, Professor Asgeir Sørensen, would have called it).

Datcom

In 1973, due to lack of methods to accurate estimate hydrodynamic characteristics of submerged vehicles, the U.S Navy funded a systematic approach for analytically predicting these parameters (Humphreys 1981). The initial face of this effort involved adaption of the USAF Stability and Control Datcom. Datcom (1978) is a public-domain stability and control handbook, devised by McDonnell Douglas for the U.S. Air Force for determination of aircraft stability properties. Datcom is a comprehensive collection of aerodynamic stability and control techniques which is widely used through the aviation industry (Nelson 1997). It provides semi-empirical methods for calculation of the subsonic, transonic, and supersonic aircraft stability. In the following years, several authors have used the Datcom to estimate stability derivatives for autonomous underwater vehicles e.g. (Nahon 1993, Barros et al. 2004); some also provide comparison with experimental data, e.g. (Maeda & Tatsuta 1989, Nahon 1996, Ridley et al. 2003), and report remarkably good agreement between estimated and measured derivatives.

The Datcom can deal with a broad scope of aircraft shapes, some of which bear a strong resemblance to streamlined underwater vehicles, which are designed for a steady-state cruising condition. The major difference between aircrafts and underwater vehicles is the medium in which they operate; differences in operating medium, which must be accounted for, are principally those due to density and viscous effects (Nahon 1993). Essentially, density effects are removed by making the variables involved non-dimensional. Viscosity effects may be accounted for through the Reynolds number at which the vehicle operates, i.e.,

$$\mathrm{R}_n = \frac{U\, l}{\nu}\,.$$

If we look at the kinematic viscosity, ν, for water, it is about one order of magnitude smaller than that of air, while the speed (U) of a typical AUV is in the order of two magnitudes smaller than a typical subsonic aircraft. In addition, the length (l) of an AUV is in the order of one magnitude smaller than a typical

aircraft; thus, the Reynolds number for a typical AUV is comparable with that of an airplane in the lower subsonic range.

1.2 Contribution and organization

Chapter 2

We start the next chapter with a brief discussion of the various coordinate systems, commonly used in marine (local) navigation. Then we discuss some issues regarding the shape and design of an autonomous underwater vehicle. At the end of the chapter, we present the dynamic model; and how to obtain estimates of restoring forces, Coriolis and added mass.

Chapter 3

Here, we will present the theory and equations used for calculation of lift, drag and dynamic derivatives for a typical AUV. In general, it is well known that there exists some mutual interference between the components in a foil-body combination; for example, the total lift on a combination is greater than the sum of the lift of the foil and body alone. In Chapter 3, great effort has been placed on determining the interference between these components.

All equations and expressions, which are mainly based on Datcom (1978) and the references therein, have been rewritten into a form, which is standard for submerged bodies (SNAME 1950).

Chapter 4

In Chapter 4, we will calculate the expected performance for the MAYA AUV in the vertical plane. The vehicle considered here, is identical to the one planned to undergo towing tank experiments in the Marine Cybernetics Laboratory (Fossen 2004*b*), during the fall of 2004. We will construct a linearized model in the vertical plane, and develop a transfer functions to estimate the expected performance.

Chapter 5

Here we will take the theory presented in Chapter 3 and 4 one step further. An MATLAB toolbox, hde, developed for estimating the static and dynamic parameters discussed in the previous chapters is introduced. The toolbox includes a nonlinear MATLAB Simulink interface, which can be used for simulating vehicles constructed in the hde environment. This particular application, can be used to simulate expected performance of vehicles, as well as for testing the robustness of controllers in a non ideal environment. The output from the application is compared with WAMIT, and experimental results.

Chapter 6

We end this thesis with a chapter where we discuss the results obtained from the previous chapters; we also state some conclusions and recommendations for future work.

Chapter 2

Preliminaries

In the following, it is assumed that the reader has some basic knowledge regarding the notation used to express the kinematic and dynamic of a marine vehicle.

2.1 Coordinate systems

In this section we will briefly describe the coordinate systems which are commonly used to express the kinematic and dynamic of a marine vehicle. The theory presented is far from complete; only equations and theory needed in the next-coming chapters are described in some detail. For a thorough and detailed explanation of the different coordinate systems, the reader is advised to visit e.g. Fossen (2002).

2.1.1 NED

The *North-East-Down* coordinate system is defined relative to the earths reference ellipsoid. For this system the x-axis points towards the true *North*, the y-axis points towards *East*, and the z-axis points *downwards* normal to the Earth's surface (Fossen 2002).

If we assume that the operating radius of the vehicle considered is limited, it is common to approximate NED to flat earth navigation. Which means that the tangent plane is fixed on the Earth surface- i.e. the *latitude*, μ, and *longitude*, l, are constant (see Figure 2.1).

2.1.2 Body

The body-fixed reference frame $[x_b,\, y_b,\, z_b]^T$ is a moving coordinate frame which is fixed to the vessel. The position and orientation of the vessel are described relative to an inertial reference frame while the linear and angular velocities of the vessel should be expressed in the body fixed coordinate system. The body fixed velocities, $\boldsymbol{\nu}$, may be transformed into NED by a transformation provided in Appendix A.3.

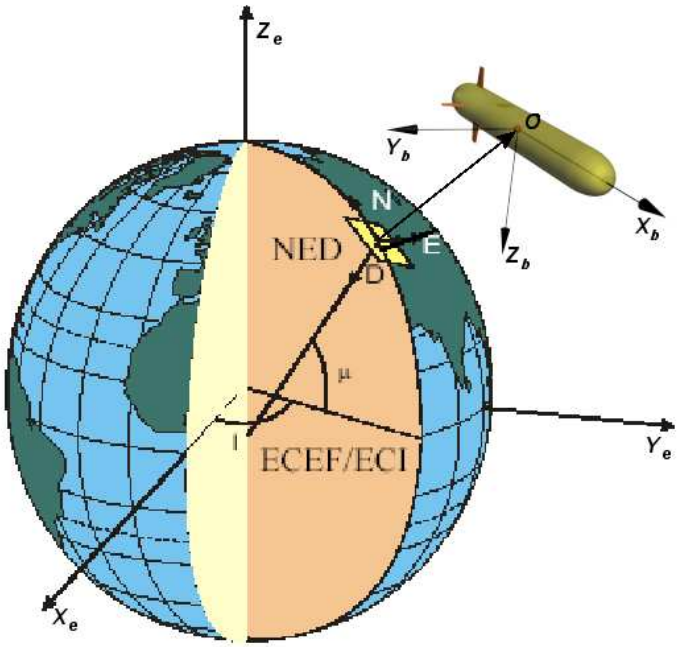


Figure 2.1: Reference frames (courtesy Fossen (2004*a*))

2.1.3 Current and Stability Axes

When deriving the hydrodynamic derivatives for a marine vessel it might be advantageous to rotate the body fixed axes such that the direction of the speed

$$U = \sqrt{u_r^2 + v_r^2 + w_r^2} \tag{2.1}$$

points in the opposite direction of the new x-axis (Fossen 2002). The drag force will then be along the body fixed x-axis, while the lift force will be along the z-axis. Lift and drag forces can then be computed as a function of forward speed and transformed back to the body frame coordinates. The transformation between the body frame and stability/current axes may be written as

$$\begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} = \boldsymbol{R}_{y,\alpha}^T \, \boldsymbol{R}_{z,-\beta}^T \begin{bmatrix} U \\ 0 \\ 0 \end{bmatrix} \tag{2.2}$$

where $\boldsymbol{R}_{y,\alpha}$ and $\boldsymbol{R}_{z,-\beta}$ are given by (2.3) and α and β are defined below. Note that in (2.1) and (2.2), we have assumed that the relative velocity vector is decomposed in the body frame (possibly through an angle of roll, γ; defined below).

angle of attack, $\boldsymbol{\alpha}$; *the angle to the longitudinal body axis from the projection into the principal plane of symmetry of the velocity of the origin of the body axes relative to the fluid, positive in the positive sense of rotation about the y-axis* (SNAME 1950).

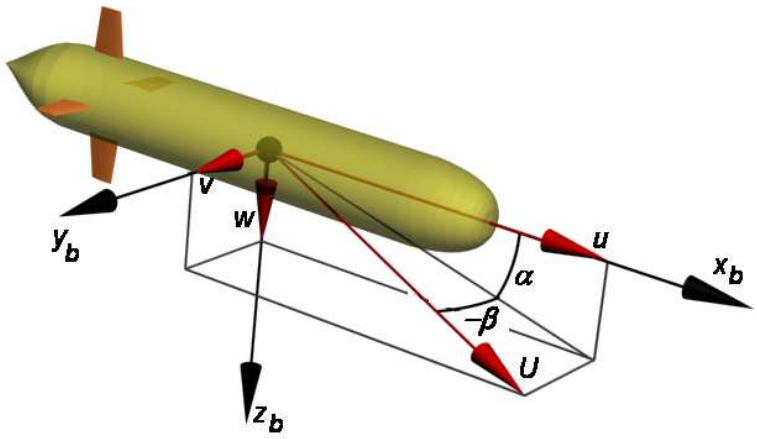


Figure 2.2: Definition of current and stability axes

sideslip angle, β; *the angle to the principal plane of symmetry from the velocity of the origin of the body axes relative to the fluid, positive in the positive sense of rotation about the z-axis* (SNAME 1950).

angle of roll, γ; *the angular displacement about the x_0-axis of the principal plane of symmetry from the vertical, positive in the positive sense of rotation about the x_0-axis* (SNAME 1950).

$$\boldsymbol{R}_{y,\alpha} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}, \quad \boldsymbol{R}_{z,-\beta} = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

2.2 AUV design (typical)

Most AUVs maneuver by means of deflecting control surfaces (hydroplanes), thrusters or a combination of both. Lateral or vertical thrusters are not normally used, as they are less effective when lengthy transits at a constant "cruise" speed is required and a low-speed hovering capability is not. The vehicles computer uses the hydroplanes for closed-loop control of heading, pitch, roll and depth.

2.2.1 Control surfaces

A hydrofoil immersed in a fluid at angle of attack, is intended to provide a large force normal to the free stream (lift) and as little drag as possible. Conventional design practice has evolved a shape not unlike a bird's wing with a rounded leading edge and a sharp trailing edge (White 1998). For our purpose, we will consider straight tapered, non cambered foil sections; which is symmetric about the chord line between the leading- and trailing edge. Definitions of various parameters used for calculating the lift and drag forces in Chapter 3, are provided in Table 2.2.1 and Figure 2.4

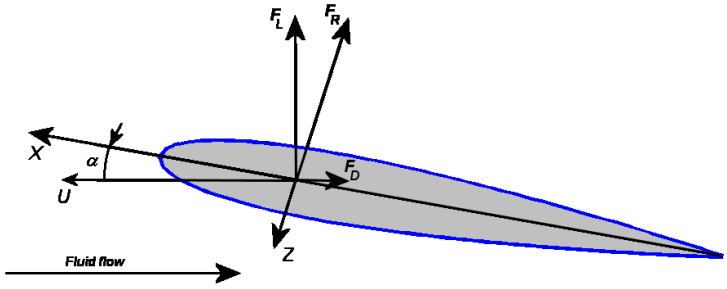


Figure 2.3: Forces on a hydrofoil

$\mathcal{R}_e$	aspect ratio, exposed foil	$\frac{(b-d)^2}{S_e}$
b	foil span	
c	chord of foil at any given span station y (parallel to X_B)	
$\bar{c}$	mean hydrodynamic chord	$\frac{2\,c_r}{3} \cdot \frac{1+\lambda+\lambda^2}{1+\lambda}$
c_r	root chord	
c_t	tip chord	
S_e	exposed foil area	$\frac{(b-d)\,c_r\,(1+\lambda)}{2}$
S_{wf}	foil, wetted area	
m, n	nondimensional chordwise stations in terms of c	
y_{hc}	spanwise location of mean hydrodynamic center	$\frac{d}{2} + \frac{b-d}{6} \cdot \frac{1+2\lambda}{1+\lambda}$
λ	taper ratio	$\frac{c_t}{c_r}$
Λ_{le}	sweep angle of leading edge	
Λ_{te}	sweep angle of trailing edge	
Λ_m, Λ_n	sweep angles of arbitrary chordwise locations	
	$\tan\Lambda_n = \tan\Lambda_m - \frac{4\,(n-m)}{\mathcal{R}_e} \cdot \frac{1-\lambda}{1+\lambda}$	

Table 2.1: Straight tapered foil, parameters

2.2.2 Body shape

The shape of an AUV determines the propulsion energy required, as well as the stability and maneuverability at various operating speeds. A hull form may also impose limitations on launch and recovery, vehicle access, and maintenance.

Skin friction and form drag contribute to overall vehicle drag. Friction drag varies with speed and the exposed surface area , which imply that smaller hulls with less surface area have less friction drag. Form drag is a function of the how well the hull shape minimizes flow separation. Longer, more slender shapes tend to be better in this respect. Test results indicate that streamlined bodies

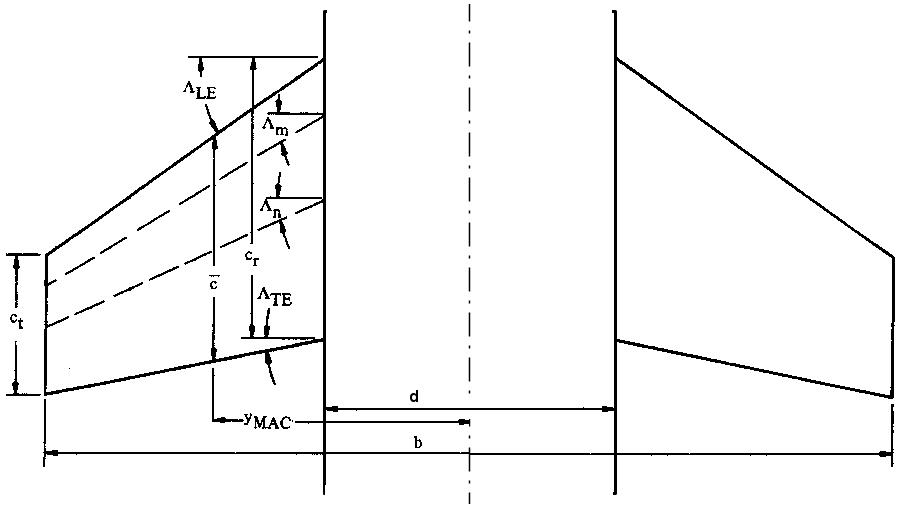


Figure 2.4: Foil parameters (Datcom 1978)

with length to diameter ratios between 5 and 7 are best for minimizing drag (Gertler 1950, Carmichael 1966).

From the simple perspective of drag reduction, a streamlined body form (see Figure 2.5) which promotes laminar flow within the boundary layer is the best choice. In laminar flow, fluid particles move in layers. Skin friction drag is lower than in a turbulent boundary layer where the fluid particles move more erratically resulting in higher shear stresses. To sustain laminar flow, a hull is designed such that the diameter increases gradually from the nose to create a favorable pressure gradient over the forward 60-70% of the hull (International Submarine Engineering Ltd 2004). In this area, the surface must be smooth and as hydrodynamically "clean" as possible. Forward-mounted hydroplanes cannot be fitted as they could destabilize the laminar flow. Consequently, all hydroplanes are mounted on a tailboom. Unfortunately, the unique shape of the laminar flow body does not readily permit lengthening (or shortening) the vehicle, thus limiting the possibility of modular expansion.

Figure 2.5: Laminar form hull

A simpler alternative to the laminar flow form is commonly called the "torpedo body". The torpedo body has a nose cap followed by a parallel mid-section and a tapered tail section. Since the torpedo body is not a laminar flow vehicle, drag should be expected to be greater than with the laminar flow hull. While lacking the hydrodynamic efficiency of the laminar flow hull, the torpedo body drag is relatively insensitive to manufacturing imperfections or damage that may occur. The packing density for payload and power source is also better in the torpedo hull than in the laminar flow body. Furthermore, the long parallel mid-section allows the vehicle length to be extended for future growth. The shape of a torpedo body, see Figure 2.6, may be divided into three sections (Myring 1976):

Nose The shape is given by the modified semi-elliptical radius distribution

$$r(x) = \frac{d}{2}\left(1 - \left(\frac{x-a}{a}\right)^2\right)^{\frac{1}{n}}, \quad 0 \le x \le a\,, \tag{2.4}$$

where d is the maximum body diameter, a is the nose length, and n is a parameter which determines the shape of the nose.

Midsection Here, the shape is simply given by a cylinder with constant radius;

$$r(x) = \frac{d}{2}, \quad a \le x \le b\,. \tag{2.5}$$

Tail The radius distribution is given by the cubic relationship

$$r(x) = \frac{d}{2} - \left(\frac{3\,d}{2\,(L-b)^2} - \frac{\tan\theta}{L-b}\right)(x-b)^2 + \left(\frac{d}{(L-b)^3} - \frac{\tan\theta}{(L-b)^2}\right)(x-b)^3\,, \quad b < x < L\,, \tag{2.6}$$

where 2θ is the included angle at the tip of the tail and L is the total length of the vehicle ($L > a + b$). Different tail shapes are produced, by varying θ.

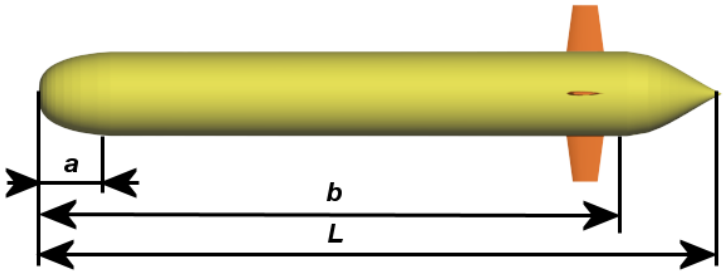


Figure 2.6: Torpedo shaped AUV

2.3 Hydrodynamic Computation, Background

The equations of motion for a submerged body involves the study of statics and dynamics. Static is concerned with the equilibrium of bodies at rest or moving with constant velocity, whereas dynamics is concerned with bodies having accelerated motion (Fossen 2002). Using the notation of Fossen, the dynamic equations of motion, for a submerged vehicle operating outside the wave affected zone, can conveniently be expressed as

$$(\boldsymbol{M}_{RB} + \boldsymbol{M}_A)\,\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\nu_r + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau}\,, \qquad (2.7)$$

where the subindexes RB and A refers to the rigid body and added mass respectively. The other terms are explained in Table 2.2.

Symbol	Description
$\boldsymbol{M}_{RB} + \boldsymbol{M}_A$	system inertia matrix
$\boldsymbol{C}(\,\cdot\,)$	Coriolis centripetal matrix
$\boldsymbol{D}(\,\cdot\,)$	damping matrix
$\boldsymbol{g}(\boldsymbol{\eta})$	vector of restoring forces and moments
$\boldsymbol{\tau}$	vector of control inputs
$\boldsymbol{\nu}$	body fixed velocity vector
$\boldsymbol{\nu}_r$	relative velocity vector
$\boldsymbol{\eta}$	position and orientation vector

Table 2.2: Rigid-body dynamics, nomenclature

The rigid body system inertia matrix will not be further explained in this report; the interested reader is instead advised to visit e.g Fossen (2002) for a thorough and detailed explanation. Further, theory for estimation of the nonlinear damping matrix, $\boldsymbol{D}(\,\cdot\,)$, and the nonlinear vector of control, $\boldsymbol{\tau}$, will not be presented in a way which is directly applicable to the notation used in (2.7). Instead, we will use an approach, in Chapter 5, where we calculate these hydrodynamic properties directly from known relations which govern the flow around simple shapes. Theory and procedures for estimating the other terms are given below.

2.3.1 Hydrostatics

The vehicle experiences hydrostatic forces and moments as a result of the combined effects of the vehicle weight (W) and buoyancy ($B = \rho\, g\, \nabla$). If we define the earth fixed buoyancy and gravity vectors as $\boldsymbol{f}_b^n = [0,\, 0,\, -B]^{\mathrm{T}}$ and $\boldsymbol{f}_g^n = [0,\, 0,\, W]^{\mathrm{T}}$ respectively. Then the body fixed restoring forces and moment vector is given by

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{bmatrix} \boldsymbol{R}_n^b(\boldsymbol{\Theta})\,\left(\boldsymbol{f}_b^n + \boldsymbol{f}_g^n\right) \\ \boldsymbol{S}(\boldsymbol{r}_g^b)\,\left(\boldsymbol{R}_n^b(\boldsymbol{\Theta})\boldsymbol{f}_g^n\right) + \boldsymbol{S}(\boldsymbol{r}_b^b)\,\left(\boldsymbol{R}_n^b(\boldsymbol{\Theta})\boldsymbol{f}_b^n\right) \end{bmatrix}\,, \qquad (2.8)$$

where $\boldsymbol{r}_g^b = [x_G,\, y_G,\, z_G]^{\mathrm{T}}$ and $\boldsymbol{r}_b^b = [x_B,\, y_B,\, z_B]^{\mathrm{T}}$ are the body-fixed location of the center of gravity (CG) and center of buoyancy (CB) respectively (relative

to a given point of reference). $\boldsymbol{S}(\,\cdot\,)$ and $\boldsymbol{R}_n^b(\boldsymbol{\Theta})$ are the vector cross product operator and the euler rotation matrix respectively (Fossen 2002), defined in Appendix A.3.

2.3.2 Coriolis

According to Fossen (2002) the system inertia matrix may be utilized to obtain a skew symmetric representation of the Coriolis centripetal matrix. If we write the 6×6 system inertia matrix as

$$\boldsymbol{M} = \boldsymbol{M}^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{M}_{11} & \boldsymbol{M}_{12} \\ \boldsymbol{M}_{21} & \boldsymbol{M}_{22} \end{bmatrix} > 0\,,$$

where $\boldsymbol{M}_{21} = \boldsymbol{M}_{12}^{\mathrm{T}}$. Then the Coriolis-centripetal matrix can be parameterized such that $\boldsymbol{C}(\boldsymbol{\nu}) = -\boldsymbol{C}^{\mathrm{T}}(\boldsymbol{\nu})$ by choosing

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} \boldsymbol{0}_{3\times 3} & -\boldsymbol{S}(\boldsymbol{M}_{11}\boldsymbol{\nu}_1 + \boldsymbol{M}_{12}\boldsymbol{\nu}_2) \\ -\boldsymbol{S}(\boldsymbol{M}_{11}\boldsymbol{\nu}_1 + \boldsymbol{M}_{12}\boldsymbol{\nu}_2) & -\boldsymbol{S}(\boldsymbol{M}_{21}\boldsymbol{\nu}_1 + \boldsymbol{M}_{22}\boldsymbol{\nu}_2) \end{bmatrix}, \tag{2.9}$$

where $\boldsymbol{S}(\,\cdot\,)$ is the cross product operator, defined by (A.4), and $\boldsymbol{\nu}_2 = [p,\, q,\, r]^{\mathrm{T}}$. If we consider the Coriolis contribution from the rigid body, then $\boldsymbol{\nu}_1 = [u,\, v,\, w]^{\mathrm{T}}$; else if we consider the contribution from the added mass matrix, then $\boldsymbol{\nu}_1 = [u_r,\, v_r,\, w_r]^{\mathrm{T}}$.

2.3.3 Added-Mass

If we restrict ourself to a typical AUV design, where the body has top-bottom and port-starboard symmetry, the vehicle added mass matrix may be written as

$$\boldsymbol{M}_A = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & Y_{\dot{r}} \\ 0 & 0 & Z_{\dot{w}} & 0 & Z_{\dot{q}} & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & M_{\dot{w}} & 0 & M_{\dot{q}} & 0 \\ 0 & N_{\dot{v}} & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix}. \tag{2.10}$$

where the various nonzero components are the partial derivative of a force or moment component X, Y, Z, K, M, or N with respect to a linear or angular acceleration $\dot{u}$, $\dot{v}$, $\dot{w}$, $\dot{p}$, $\dot{q}$ or $\dot{r}$ (Fossen 2002, SNAME 1950). It should be noted that the hydrodynamic system inertia matrix for an AUV will be positive definite, and can be approximated by $\boldsymbol{M}_A = \boldsymbol{M}_A^{\mathrm{T}}$.

In general, the added mass terms are a function of body shape, the direction of motion, and (to a lesser extent) flow parameter such as Reynolds number (White 1998). However, for a body of revolution it is possible to approximate added mass terms by a *strip theory* synthesis, in which the flow at each section is assumed to be locally two dimensional (Newman 1999). Strip theory can also be used for the added mass terms due to control surfaces, if the two dimensional geometry is approximated to a flat plate representing the foils. Unfortunately, strip theory fails to give any estimate of the axial added mass in surge.

Axial Added Mass

Inspired by (Triantafyllou & Hover 2003, Problem 11), we approximate the vehicle hull shape by an ellipsoid for which the major axis is half the vehicle length, L_{pp}, and the minor axis half the vehicle diameter, d. Blevins (1979) gives the following empirical formula for the axial added mass of an ellipsoid:

$$X_{\dot{u}} = -m_{11} = -\frac{4\varpi\rho\pi}{3}\left(\frac{d}{2}\right)^3 , \tag{2.11}$$

where ρ is the fluid density, and ϖ is an empirical parameter measured by Blevins (1979) and determined by the fineness ratio, f, of the vehicle as shown in Table 2.3 and Figure 2.7.

L_{pp}/d	ϖ	L_{pp}/d	ϖ
0.01	0.6348	1 .5	0.4557
0.1	0.6148	2 .0	0.4200
0.2	0.6016	2 .5	0.3908
0.4	0.5712	3 .0	0.3660
0.6	0.5447	5 .0	0.2956
0.8	0.5211	7 .0	0.2510
1.0	0.5000	10.0	0.2071

Table 2.3: Axial added mass parameter, ϖ (Blevins 1979)

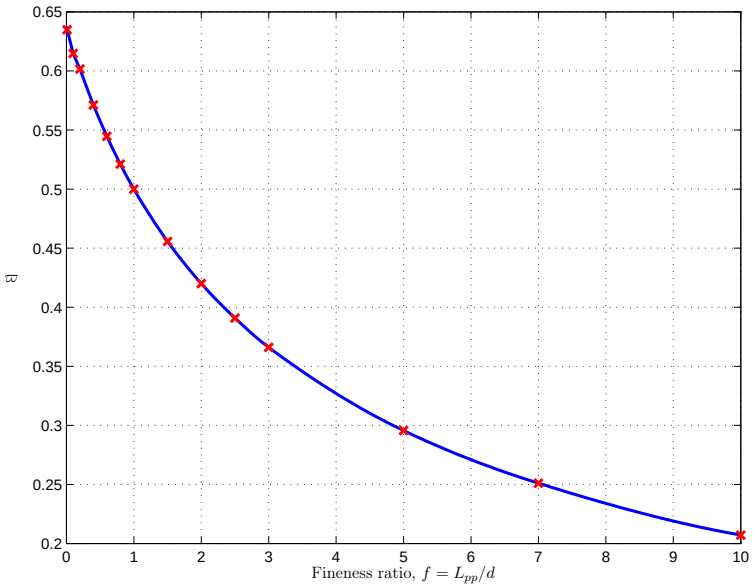


Figure 2.7: Axial added mass parameter (interpolated)

Crossflow Added Mass

The added mass for the body alone may be calculated with the aid of 2 dimensional strip theory. For a body of revolution, the added mass per unit length is according to Newman (1999) given as

$$m_{a,b}(x) = \frac{\pi\, d(x)^2}{4}\rho\,, \tag{2.12}$$

and the contribution from a foil section, mounted normal to the axis of reference, is given by by[1]

$$m_{a,f}(x) = \frac{\pi\,\rho}{4}\left(\frac{b(x)^2 - d(x)^2}{b(x)}\right)^2\,, \tag{2.13}$$

where $b(x)$ is the foil span as a function of axial position x.

Added inertia, roll

If we assume that the relative smooth sections of the vehicle hull do not generate any added mass in roll, and neglect the added mass generated by additional equipment fitted to the hull (sonar, transponders etc.), the two dimensional added mass in roll is considered to be the result of the control surfaces alone. From Newman (1999), the contribution from each section is given by

$$m_a(x) = \frac{\rho\, d^4}{16}\left(\frac{\left(2\,\alpha^2 - \alpha\,\sin 4\,\alpha + \frac{1}{2}\,\sin^2\,2\alpha\right)\cdot \csc^4\alpha}{\pi} - \frac{\pi}{2}\right)\,, \tag{2.14}$$

where $\sin\alpha = 2\,d\,b/\left(d^2 + b^2\right)$ and $\pi/2 < \alpha < \pi$; $\csc(\,\cdot\,) = 1/\sin(\,\cdot\,)$ is the cosecant function.

Coupling terms and added inertia

Even though it might-be possible to obtain an "exact" solution of these terms by integrating the appropriate added mass term(s) with the proper moment arm, it is usually easier to obtain an estimate of the solution by numerical integration; dividing the body into a finite number, N, of discrete panels and summing the

[1] (2.13) is a modified version of the expression given in (Newman 1999, Table 4.3). The contribution from the cylindrical section is removed, due to the fact that it has already been accounted for in (2.12).

contribution from each panel, yields an estimate of the total. Thus,

$$M_{\dot{w}} = Z_{\dot{q}} \simeq \sum_{i=1}^{N-1} (\hat{x} - x_m) \cdot \delta_x \, m^{xy}_{i(\hat{x})} \,, \tag{2.15a}$$

$$M_{\dot{q}} \simeq - \sum_{i=1}^{N-1} (\hat{x} - x_m)^2 \cdot \delta_x \, m^{xy}_{i(\hat{x})} \,, \tag{2.15b}$$

$$N_{\dot{v}} = Y_{\dot{r}} \simeq - \sum_{i=1}^{N-1} (\hat{x} - x_m) \cdot \delta_x \; m^{xz}_{i(\hat{x})} \,, \tag{2.15c}$$

$$N_{\dot{r}} \simeq - \sum_{i=1}^{N-1} (\hat{x} - x_m)^2 \cdot \delta_x \, m^{xz}_{i(\hat{x})} \,, \tag{2.15d}$$

where

$$\hat{x} = \frac{x_i + x_{i+1}}{2} \,, \qquad\qquad \delta_x = x_{i+1} - x_i \,,$$

and the different m_i terms are the the sum of (2.12) and (2.13)[2]. Note that in (2.15), we have written the equations in such a way that we can define the origin of $\hat{x}$ at the nose vertex (positive aft); the transformation into the body frame is ensured through x_m, the distance from the nose vertex to the body fixed origin (positive aft). It should also be noted that we have assumed that the body fixed reference point is along the axis of revolution.

[2]The dimension to be used (b and d), are the one that appears at $\hat{x}$, if we look at a cross-section of the body through the plane indicated by the superscript of m_i. If no foil is present, $b = d$.

Chapter 3

Lift and Drag

In this chapter, a method for calculating lift, drag and hydrodynamic derivatives for a typical AUV is presented. The theory and expressions are mainly based on empirical methods from *The USAF Stability and Control Datcom* (Datcom 1978), and references therein.

Datcom is a data compendium for determination of aircraft stability properties. It can deal with a broad scope of aircraft shapes, some of which bear a strong resemblance to typical AUV's. The major difference between aircrafts and underwater vehicles is the medium which they operate. Differences in operating medium which must be accounted for are principally those due to density and viscous effects. Since these, in general, are well identified in Datcom, it is particulary easy to account for them when applying the methodology for underwater vehicles. Essentially, density effects are removed by non-dimensionalization of the variables, compressibility effects are ignored and viscousity effects are accounted for through the Reynolds number at which the vehicle operates, i.e.

$$\mathrm{R}_n = \frac{U\,L}{\nu}$$

The kinematic viscosity, ν, for water is about one order of magnitude smaller than that of air, while the speed (U) of a typical AUV is in the order of two magnitudes smaller than typical a subsonic aircraft. In addition, the length (L) of an AUV is in the order of one magnitude smaller than a typical aircraft; thus, the Reynolds number for a typical AUV is comparable to that of an airplane in the lower subsonic range.

Datcom (1978) typically use the wing planform area as its reference area for the nondimensional coefficients, while most standard equations for marine vessels use the square of the vehicle body length (SNAME 1950). In this text, however, the equations has been rewritten into standard notation for submerged bodies.

The theory presented in this chapter only deals with the vertical plane, but it should be clear that same theory is applicable for the corresponding derivatives as a function of sideslip, β, in the horizontal plane.

3.1 Lift

Control of an AUV can be achieved by providing an incremental lift force on one or more of the AUV's lifting surfaces. Because the movable lifting surfaces are located at some distance from the center of gravity, the incremental lift force creates a moment about the center of gravity. Pitch and yaw control can be achieved by changing the lift on the control surfaces (elevator and rudder), and roll control can be achieved by deflecting the control surfaces in a differential manner.

3.1.1 Foil alone

Lift on a foil at angle of attack results from the distributed pressure over the surface of the foil. Most of the lift from a foil is derived from a region of low pressure on the upper surface near the leading edge. The magnitude and distribution of this pressure field is such that its integrated value over the foil surface results in a force vector which is nearly perpendicular to the free-stream direction (Hoerner 1985).

In a typical AUV design, it is common to use control surfaces with low-aspect ratio ($\mathcal{R}_e$). For these low aspect-ratio foils, experimental results shows that they have a relatively high stall angle of attack ($20-25°$) and a very "flat" stall (Jones 1952, Nahon 1996, Whicker & Fehlner 1958); that is, their lift coefficient stops rising but do not drop much past the stall point. It is also known that the characteristic of the lift curve slope, $C_{L_\alpha} = \frac{\partial C_L}{\partial \alpha}$, is essentially linear with angle of attack until stall occurs (Abbot & Doenhoff 1959).

If experimental data are available, it is normally recommended that these data are used to obtain the three dimensional lift-curve slope (Datcom 1978, Abbot & Doenhoff 1959); for low aspect-ratio foils, McCormick (1995) recommend the expression given by (3.1a). Whicker & Fehlner (1958) also provide an expression, (3.1b), which has proven to yield results that are very close to experimental data. Note that in (3.1b), only geometric properties are necessary for calculating the three dimensional lift-curve slope.

$$C_{L_\alpha} = \frac{\mathcal{R}_e}{\mathcal{R}_e + 2\frac{\mathcal{R}_e+4}{\mathcal{R}_e+2}} c_{l_\alpha} \tag{3.1a}$$

$$C_{L_\alpha} = \frac{S_e}{L_{pp}^2} \cdot \frac{2\,\pi\,\kappa\,\mathcal{R}_e}{2\kappa + \sqrt{\frac{\mathcal{R}_e^2}{\cos^4 \Lambda_{c/4}} + 4}}\,, \tag{3.1b}$$

In (3.1), c_{l_α} is the two-dimensional lift curve slope obtained from experimental results, i.e (Abbot & Doenhoff 1959), $\mathcal{R}_e$ is the aspect ratio of the exposed foil, S_e is the planform area of the exposed foil, $\Lambda_{c/4}$ is the sweep angle at one fourth of the chord, and $\kappa = 0.9$ is *"the section lift-curve slope corrected from experimental observations"* (Whicker & Fehlner 1958).

3.1.2 Body alone

The method presented in Datcom to estimate the lift curve slope for a body of revolution, is based on potential theory. Potential theory is limited to angles of attack near zero, where viscous effects are small. At higher angles of attack the viscous forces become increasingly important, and should be taken into account.

From Datcom (1978), the lift-curve slope for a body of revolution is given by

$$C_{L_\alpha} = \frac{2 \cdot (k_2 - k_1) \cdot S_0}{L_{pp}^2} \quad [1/\text{rad}]\,, \tag{3.2}$$

where $(k_2 - k_1)$ is the apparent mass factor developed by Munk (1934) and presented in Figure 3.1 as a function of body fineness ratio ($f = \frac{L_{pp}}{d}$). S_0 is the body cross-sectional area at the point (x_0) where the flow ceases to be potential. As shown in Figure 3.2, experimental results indicate that x_0 appears at the junction of the nose and the cylindrical midbody for a torpedo shaped vehicle; thus, $S_0 = \frac{\pi d^2}{4}$. For a streamlined vehicle, however, x_0 is a function of x_1, the body station where the slope of the curve of revolution, $\frac{\mathrm{d}r(x)}{\mathrm{d}x}$, first reaches its maximum negative value (Datcom 1978, Paster 1986). The relationship between x_0 and x_1 is given in Figure 3.3.

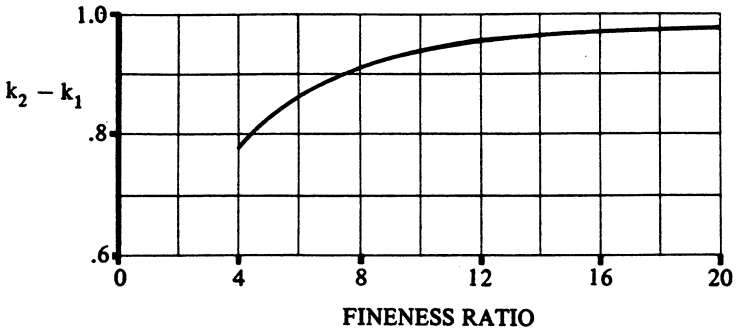


Figure 3.1: Apparent mass factor (Munk 1934)

3.1.3 Foil- body combination

When a lifting surface is added to a body, certain mutual interference effects may arise between the components. Datcom (1978) present a method, that builds on the methodology derived in Pitts, Nielsen & Kaattari (1957). The key idea is to compute interference factors between the lift surfaces and the body using slender-body theory. The first step in the procedure assumes that fins do not deflect, the effect of deflection is taken into account in Step 2.

Step 1

If we look at the foil-body combination as a unit (foil incidence is fixed relative to the body), and the angle of attack is varied for the foil-body combination as

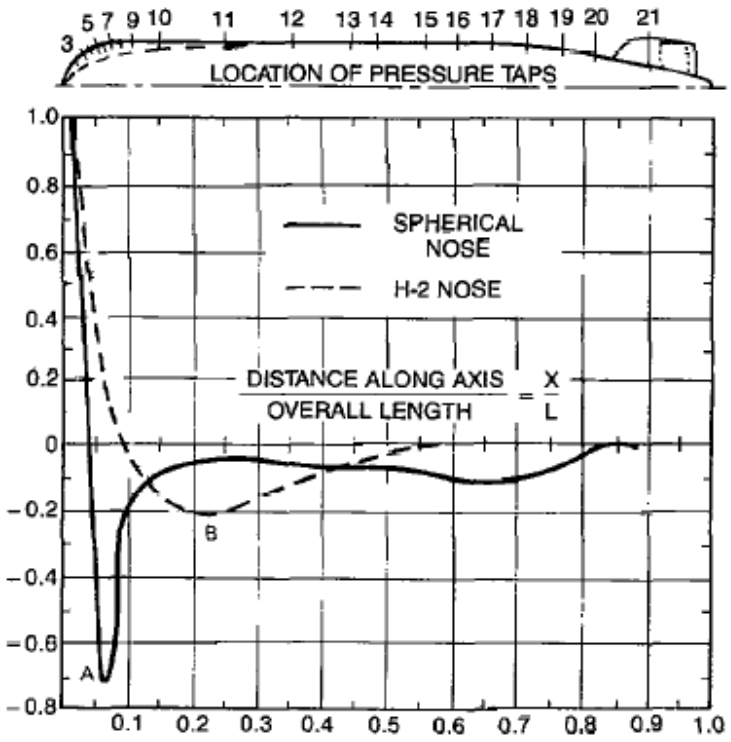


Figure 3.2: Pressure distribution of typical torpedo (Paster 1986)

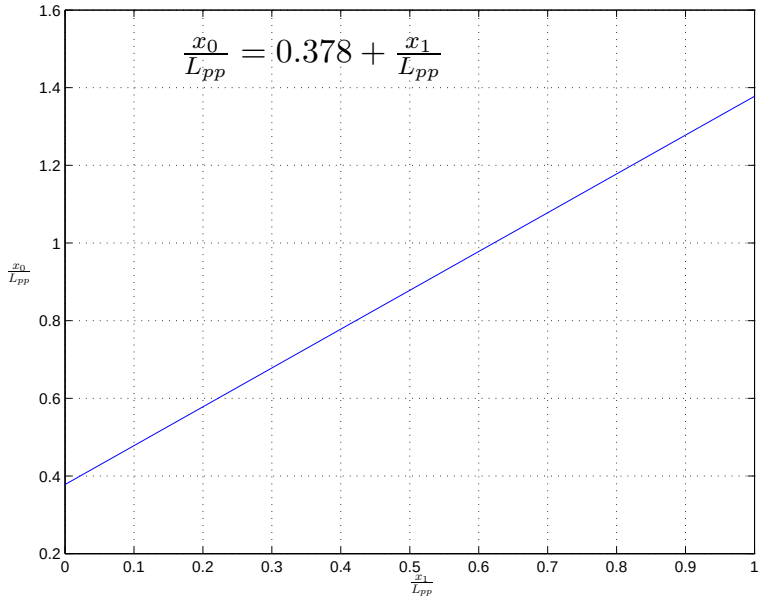


Figure 3.3: Body station where flow cease to be potential

a unit, the equation for the fin- body lift curve slope (based on L_{pp}^2) is given by

$$(C_{L_\alpha})_{FB} = (C_{L_\alpha})_B + (C_{L_\alpha})_{F(B)} + (C_{L_\alpha})_{B(F)} \tag{3.3}$$

where $(C_{L_\alpha})_B$ is the bare hull lift coefficient given by (3.2), and

$$(C_{L_\alpha})_{F(B)} = K_{F(B)} \cdot (C_{L_\alpha})_F \tag{3.4a}$$
$$(C_{L_\alpha})_{B(F)} = K_{B(F)} \cdot (C_{L_\alpha})_F \ . \tag{3.4b}$$

$(C_{L_\alpha})_F$ is the lift curve slope for the foil alone, given by (3.1), and the quantities $K_{F(B)}$ and $K_{B(F)}$ represents the ratios of the foil lift in the presence of body, and the body lift in presence of the foil, respectively, to the foil alone lift. So far, only a way of representing lift result has been represented. The solution of a problem requires a determination of each of these ratios.

Lift on foil in presence of body From slender body theory, the expression for $K_{F(B)}$ is according to Pitts et al. (1957) given by

$$\begin{aligned} K_{F(B)} = \frac{2}{\pi} \frac{\left(1+\varsigma^4\right)\left\{\frac{1}{2}\arctan\left(\frac{1}{2}\left(\vartheta-\varsigma\right)\right)+\frac{\pi}{4}\right\}}{\left(1-\varsigma\right)^2} \\ -\varsigma^2 \frac{(\vartheta-\varsigma)+2\,\arctan\varsigma}{\left(1-\varsigma\right)^2}\,, \end{aligned} \tag{3.5}$$

where ς is the diameter-span ratio (d/b), and $\vartheta = \varsigma^{-1}$. In the limiting case of $\varsigma = 0$ the combination is all foil and the value of $K_{F(B)} = 1$. As ς approaches unity, there is a very small exposed foil. For this small foil, the body is effectively a vertical reflection plane and the angle of attach is 2α (Pitts et al. 1957); this makes $K_{F(B)} = 2$.

Lift on body due to foil The slender body theory value of $K_{B(F)}$ is, according to Pitts et al. (1957), given by

$$K_{B(F)} = \frac{\left(1-\varsigma^2\right)^2}{\left(1-\varsigma\right)^2} - \frac{2\,\Theta}{\pi}\,, \tag{3.6}$$

where

$$\Theta = \frac{\left(1+\varsigma^4\right)\left(\frac{1}{2}\,\arctan\frac{1}{2}\left(\vartheta-\varsigma\right)+\frac{\pi}{4}\right)-\varsigma^2\left(\vartheta-\varsigma+2\,\arctan\varsigma\right)}{\left(1-\varsigma\right)^2}\,.$$

In the limiting case of $\varsigma = 0$ the combination is all foil and $K_{B(F)} = 0$. As ς approaches unity, there is a very small exposed foil. For this small foil the lift on the body due to the foil is the same as the lift on the foil itself. Thus, $K_{B(F)} = K_{F(B)} = 2$.

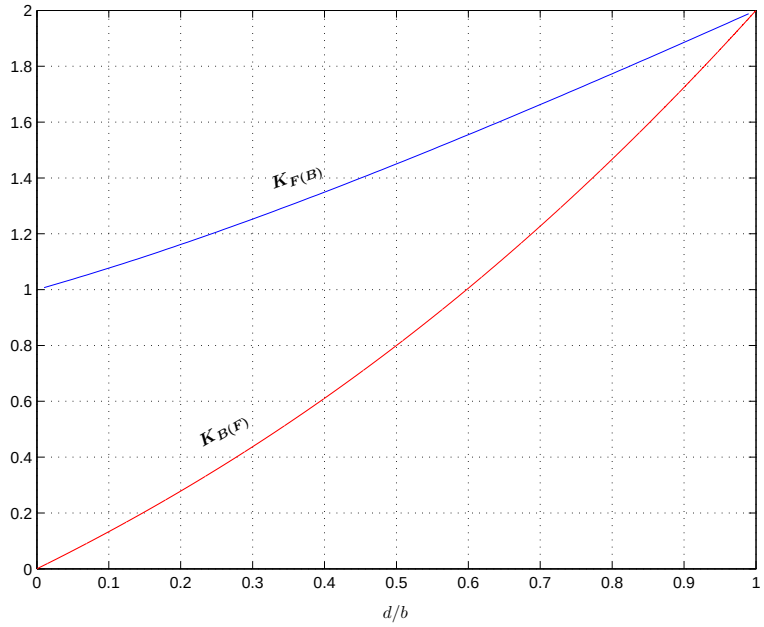


Figure 3.4: Lift ratios $K_{F(B)}$ and $K_{B(F)}$ (Slender body theory)

Step 2

If we look at the body fixed at zero angle of attack and the foil incidence varying, the equation is given by

$$(C_{L_\delta})_{FB} = \left[k_{F(B)} + k_{B(F)}\right] \cdot (C_{L_\alpha})_F \ , \tag{3.7}$$

where $k_{F(B)}$ and $k_{B(F)}$ are interference factors given by

$$\begin{aligned} \pi^2 \cdot k_{F(B)} = {} & \frac{\pi^2}{4} \frac{(\vartheta+1)^2}{\vartheta^2} + \frac{\pi \left(\vartheta^2+1\right)^2}{\vartheta^2 \left(\vartheta-1\right)^2} \arcsin \frac{\vartheta^2-1}{\vartheta^2+1} - \frac{2\,\pi\,(\vartheta+1)}{\vartheta\,(\vartheta-1)} \\ & + \frac{\left(\vartheta^2+1\right)^2}{\vartheta^2\,(\vartheta-1)^2} \arcsin^2 \frac{\vartheta^2-1}{\vartheta^2+1} - \frac{4\,(\vartheta+1)}{\vartheta\,(\vartheta-1)} \arcsin \frac{\vartheta^2-1}{\vartheta^2+1} \\ & + \frac{8}{(\vartheta-1)^2} \ln \frac{\vartheta^2+1}{2\,\vartheta} \ , \end{aligned} \tag{3.8a}$$

$$k_{B(F)} = K_{F(B)} - k_{F(B)} \ , \tag{3.8b}$$

where $K_{F(B)}$ is given by (3.6), and the value of $k_{F(B)}$ so obtained is presented in Figure 3.5.

3.2 Drag

Drag is the hydrodynamic force exerted on a body in the direction opposite to its velocity. The propulsion power requirement is proportional to the drag, times the velocity, divided by the propulsion efficiency. Drag is usually considered to

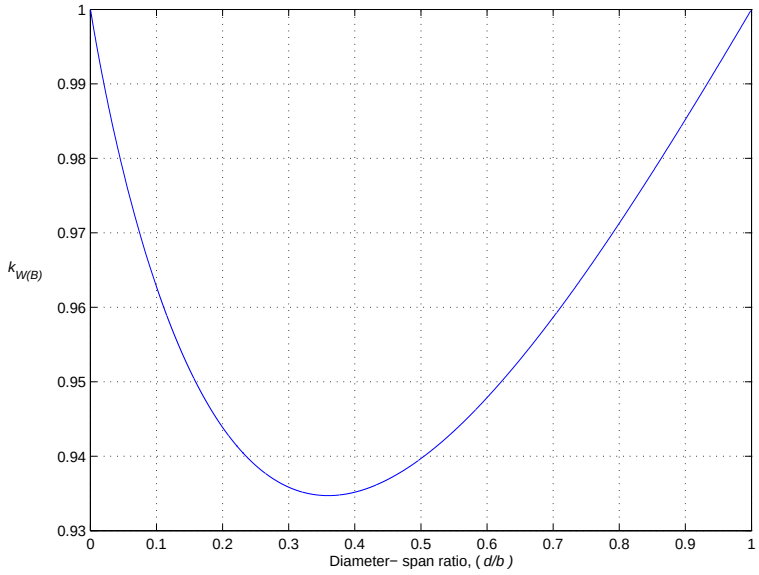


Figure 3.5: Lift ratio, $k_{F(B)}$, based on slender body theory

consist of two components (Paster 1986); friction drag, which is a function of speed and wetted area, and pressure drag (also called form drag), which is a function of shape and frontal area.

Friction drag is caused by shearing stress caused by the boundary layer, which arises from the resistance of the viscous fluid to the motion of the body passing through it. The amount of viscous resistance depends greatly on whether the flow is laminar or turbulent. The Reynolds number and the shape of the pressure disturbance determine whether the flow over the foil is laminar or turbulent (or a mixture of both).

According to the well known paradox of d'Alembert, the pressure drag is zero for a closed body immersed in an inviscid fluid. For a viscous fluid, however, there is a finite drag; resulting from an incomplete pressure recovery at the end of the body, which in turn is caused by the displacement of the boundary layer (see Figure 3.2). This drag is "small" for bodies with high fineness-ratios, f, but might become significant for blunt bodies (White 1998, Paster 1986).

3.2.1 Body alone

If we assume that there is a turbulent boundary layer condition over the entire surface, the zero-lift drag for an isolated body, based on L_{pp}^2, is given by

$$C_{D_0} = \left\{ \underbrace{C_f \left[1 + \frac{60\, d^3}{L_{pp}^3} + 0.0025 \frac{L_{pp}}{d} \right] \frac{4\, S_b}{\pi\, d^2}}_{(C_f)_b} + C_{D_b} \right\} \frac{\pi\, d^2}{4\, L_{pp}^2}\,, \tag{3.9}$$

where $(C_f)_b$ is the zero-lift drag of the body (exclusive base drag), C_f is the skin friction coefficient (explained below), L_{pp} is the length between perpendiculars, d is the maximum diameter of the body, S_b is the body wetted area (excluding the base area), and C_{D_b} is the base drag coefficient, given by

$$C_{D_b} = \frac{0.029}{\sqrt{(C_f)_b}} \cdot \left(\frac{d_b}{d} \right)^3\,, \tag{3.10}$$

where d_b is the base diameter. The term $0.0025 \frac{L_{pp}}{d}$ in (3.9) represent the pressure drag contribution.

An estimate of the skin friction coefficient, C_f, may be obtained with the aid of the "*Schoenherr Mean Line*"

$$\frac{0,242}{\sqrt{C_f}} = \log\left(\mathrm{R}_n\, C_f\right), \tag{3.11}$$

or the "*ITTC-1957*" line

$$C_f = \frac{0.075}{\left(\log\left(\mathrm{R}_n\right) - 2\right)^2} \tag{3.12}$$

(Journée & Massie 2001).

3.2.2 Foil alone

In Datcom (1978), the foil zero-lift drag coefficient, based on L_{pp}^2, is given by

$$C_{D_0} = C_f \left[1 + L \left(\frac{t}{c} \right) + 100 \left(\frac{t}{c} \right)^4 \right] R_{LS}\, \frac{S_{wf}}{L_{pp}^2}\,, \tag{3.13}$$

where:

- C_f is the turbulent flat plate coefficient as a function of the Reynolds number based on the mean hydrodynamic chord, $\bar{c}$.
- $\frac{t}{c}$ is the average streamwise thickness ratio of the foil.
- L is the hydrofoil thickness location parameter. ($L = 1.2$ for $(t/c)_{\max}$ located at $x_t \geq 0.3\, c$. $L = 2.0$ for $(t/c)_{\max} < 0.3\, c$).

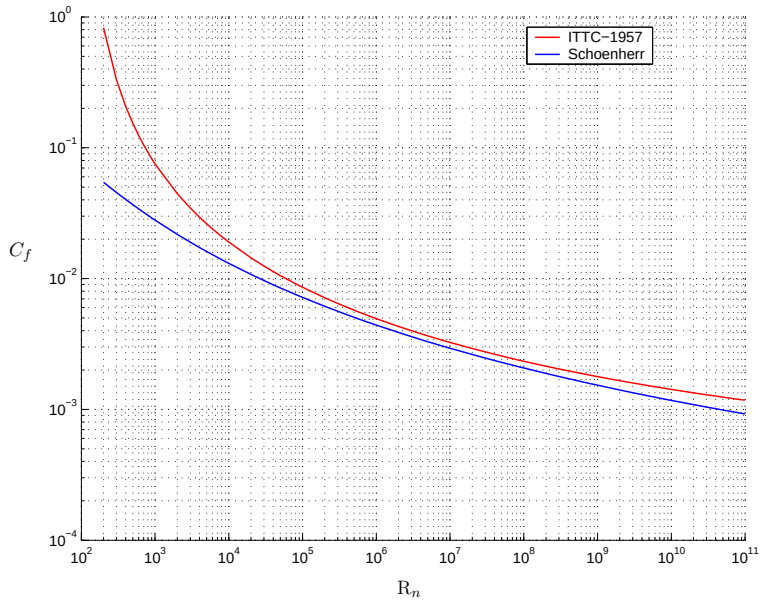


Figure 3.6: Friction drag

x_t is the chordwise position of maximum thickness.

S_{wf} is the wetted area of the foil.

R_{LS} is the lifting surface correction factor obtained from Figure 3.7, or estimated by the polynomial

$$R_{LS} = -0.993\,\theta^2 + 2.004\,\theta + 0.061\,,$$

where $\theta = \cos\Lambda_{(t/c)_{\max}}$, and $\Lambda_{(t/c)_{\max}}$ is the sweep angle at x_t.

3.2.3 Foil- body combination

In Datcom (1978), the zero-lift drag coefficient for the combination is determined by adding the drag coefficients of the exposed components and applying an interference correction factor to the skin friction and pressure-drag contributions. The component contributions of the foil and body are determined by the methods described above.

If we neglect the reduction in wetted area, caused by mounting the foil(s) to the body, the zero-lift drag coefficient of a foil-body combination, based on L_{pp}^2, is given by

$$(C_{D_0})_{FB} = \left\{ (C_{D_0})_F + (C_f)_b \, \frac{\pi\, d^2}{4\, L_{pp}^2} \right\} \cdot R_{FB} + C_{D_b} \cdot \frac{\pi\, d^2}{4\, L_{pp}^2}\,, \tag{3.14}$$

where $(C_{D_0})_F$ is the foil zero-lift drag coefficient given by (3.13), $(C_f)_b$ is the skin friction and pressure drag for the body alone; given by (3.9), and C_{D_b} is the

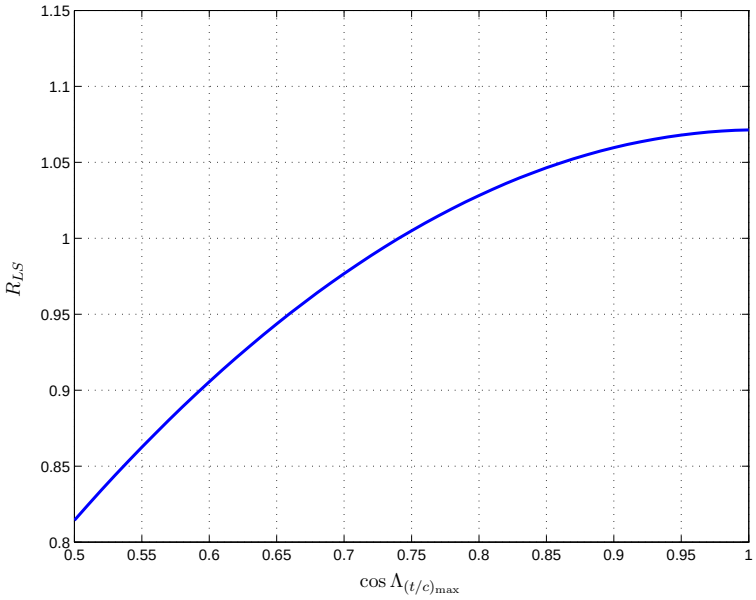


Figure 3.7: Lifting surface correlation factor

base drag coefficient given by (3.10). R_{FB} is a foil- body interference correlation factor. Unfortunately Datcom do not provide data, for R_{FB}, which is directly applicable for a typical AUV design; but if we interpolate the data provided (Datcom 1978, Figure 4.3.3.1-37), a numerical value $R_{FB} = 1.05$ seems to be reasonable.

3.3 Pitching moment, C_{m_α}

C_{m_α} is the change in the pitching moment due to change in angle of attack. This term determines the longitudinal stability of the vehicle, and must be negative in order to have a "*static stable*" vehicle (Blakelock 1991). A static stable vehicle is one that tends to return to its equilibrium condition after a disturbance has occurred. A negative C_{m_α} means that as as the angle of attack increases positively, the pitching moment becomes more negative, tending to decrease the angle of attack.

Body alone

The body pitching moment for angle of attack near zero (based on L_{pp}^3), is given by

$$\begin{aligned}(C_{m_\alpha})_B &= \frac{2\,(k_2-k_1)}{L_{pp}^3}\cdot\int_0^{x_0}\frac{\mathrm{d}S(x)}{\mathrm{d}x}\,(x_m-x)\;\mathrm{d}x \\ &\Updownarrow \quad \text{see (A.7)} \\ &= \frac{x_m\,C_{L_\alpha}}{L_{pp}} + 2\;(k_2-k_1)\,\frac{V_0-S_0\,x_0}{L_{pp}^3} \quad [1/\mathrm{rad}]\,,\end{aligned} \tag{3.15}$$

where x_m is the longitudinal distance from the nose of the body to the chosen point of reference, S_0 is the body cross-sectional area at x_0, and $V_0 = \int_0^{x_0}\frac{\pi}{4}\,d^2(x)\,\mathrm{d}x$. The other coefficients are defined in Section 3.1.2.

Foil in presence of body

The pitching moment for a foil in presence of the the body can be written as

$$(C_{m_\alpha})_{F(B)} = \left(\frac{x_{hc}}{L_{pp}}\right)_{F(B)}(C_{L_\alpha})_{F(B)}\;, \tag{3.16}$$

where $(C_{L_\alpha})_{F(B)}$ is given by (3.4a), and

$$\left(\frac{x_{hc}}{L_{pp}}\right)_{F(B)} = \frac{1}{L_{pp}}\,\left(x_m - x_{le} - \frac{2y_{hc}-d}{2}\tan\Lambda_{le} - \frac{\bar{c}}{4}\right) \tag{3.17}$$

is the non-dimensional position of the hydrodynamic center relative to x_m (Roskam 1979); the other parameters are as defined in Chapter 2.2.1.

Body in presence of foil

From Datcom (1978), the pitching moment for the foil in presence of body is given by

$$(C_{m_\alpha})_{B(F)} = \left(\frac{x_{hc}}{L_{pp}}\right)_{B(F)}(C_{L_\alpha})_{B(F)}\;, \tag{3.18}$$

where $(C_{L_\alpha})_{B(F)}$ is given by (3.4b), and $\left(\frac{x_{hc}}{L_{pp}}\right)_{B(F)}$ is the non-dimensional position of the hydrodynamic center relative to x_m. The procedure for estimating the hydrodynamic contribution due to lift carryover of the foil on the body, depends on the aspect ratio of the exposed foil, $\mathcal{AR}_e$, as described below.

Case $\mathcal{AR}_e \geq 4$. The hydrodynamic center contribution due to the lift carryover of the foil on the body is obtained from the equation

$$\left(\frac{x_{hc}}{L_{pp}}\right)_{B(F)} = \frac{x_m - x_{le}}{L_{pp}} - \frac{c_r}{L_{pp}}\underbrace{\left(\frac{1}{4} + \frac{b-d}{2\,c_r}\cdot\Xi\cdot\tan\Lambda_{c/4}\right)}_{x'_{\beta\mathcal{AR}_4}}, \tag{3.19}$$

where

$$\Xi = \frac{-\varsigma}{1-\varsigma} + \frac{\sqrt{1-2\,\varsigma}\,\ln\left(\frac{1-\varsigma}{\varsigma} + \frac{\sqrt{1-2\,\varsigma}}{\varsigma}\right) - (1-\varsigma) + \frac{\varsigma\,\pi}{2}}{\frac{\varsigma\,(1-\varsigma)}{\sqrt{1-2\,\varsigma}}\,\ln\left(\frac{1-\varsigma}{\varsigma} + \frac{\sqrt{1-2\,\varsigma}}{\varsigma}\right) + \frac{(1-\varsigma)^2}{\varsigma} - \frac{\pi\,(1-\varsigma)}{2}}\,,$$

and $\varsigma = d/b$ is the diameter span ratio; the other parameters are as defined in Chapter 2.2.1.

Case $\mathcal{AR}_e < 4$ The hydrodynamic center is obtained by interpolation[1] between $\beta\mathcal{AR}_e = 0$ and $\beta\mathcal{AR}_e = 4$. From slender body theory, the hydrodynamic center for $\beta\mathcal{AR}_e = 0$ (based on root chord length, c_r) is given by

$$x'_{\beta\mathcal{AR}_0} = \begin{cases} \frac{1}{2}\Gamma\,, & \text{if } 0 \le \Gamma < 1; \\ \frac{1}{2}\,, & \text{otherwise}\,. \end{cases}\,, \tag{3.20}$$

where

$$\Gamma = \frac{\mathcal{AR}_e\,(1+\lambda)}{4}\,\tan\Lambda_{le}\,,$$

λ is the taper ratio, and Λ_{le} is the leading edge sweep angle.

Following the procedure suggested in Datcom, the estimate of the hydrodynamic center is determined from a curve through the value for $\beta\mathcal{AR}_e = 0$ and tangent to the calculated line for $\beta\mathcal{AR}_e \ge 4$; thus,

$$\left(\frac{x_{hc}}{L_{pp}}\right)_{B(F)} = \frac{x_m - x_{le} - c_r \cdot \aleph_{(\beta\mathcal{AR}_e)}}{L_{pp}}\,, \tag{3.21}$$

where

$$\aleph_{(\beta\mathcal{AR}_e)} = \frac{x'_{\beta\mathcal{AR}_0} - x'_{\beta\mathcal{AR}_4}}{16}\,(\beta\mathcal{AR}_e - 4)^2 + x'_{\beta\mathcal{AR}_4}\,.$$

Pitching moment - foil body combination, $(C_{m_\alpha})_{FB}$

The curve slope for the foil-body combination is estimated by

$$\begin{aligned}(C_{m_\alpha})_{FB} &= \left(\frac{x_{hc}}{L_{pp}}\right)_{FB}(C_{L_\alpha})_{FB} \\ &= (C_{m_\alpha})_B + (C_{m_\alpha})_{F(B)} + (C_{m_\alpha})_{B(F)}\,,\end{aligned} \tag{3.22}$$

where $(C_{L_\alpha})_{FB}$ is the lift curve-slope for the foil-body combination, obtained from (3.3), and $\left(\frac{x_{hc}}{L_{pp}}\right)_{FB}$ is the non-dimensional position of the hydrodynamic center relative to the point of reference, x_m.

[1] In Datcom (1978), the procedure is based on the *Prandtl-Glauert* compressibility factor, $\beta = \sqrt{|1-M^2|}$, multiplied by the aspect ratio of the exposed foil, $\mathcal{AR}_e$; for all practical AUV applications, $\beta = 1$ (Nahon 1993).

3.4 Pitching moment, C_{m_δ}

C_{m_δ} is the change in pitching moment due to rudder deflection, and is obtained by

$$(C_{m_\delta})_{FB} = \left(\left(\frac{x_{hc}}{L_{pp}} \right)_{B(F)} k_{B(F)} + \left(\frac{x_{hc}}{L_{pp}} \right)_{F(B)} k_{F(B)} \right) (C_{L_\alpha})_F \,, \tag{3.23}$$

where $(C_{L_\alpha})_F$ is the lift curve slope for the foil alone, given by (3.1), $\left(\frac{x_{hc}}{L_{pp}}\right)_{B(F)}$ and $\left(\frac{x_{hc}}{L_{pp}}\right)_{F(B)}$ are the hydrodynamic centers defined above; $k_{B(F)}$ and $k_{F(B)}$ are the interference factor defined by (3.8).

3.5 Pitching derivative, C_{L_q}

When a foil-body combination rotates in pitch about a given center of gravity at an angular velocity, q, in a free-stream velocity v_∞, changes in local angle of attack that are proportional to the local vertical disturbance vector, w_r, are produced. As a result, an effective angle of attack increment for the complete combination is produced, resulting in a lift increment.

Foil alone

The equation for estimating the pitching derivative, based on L_{pp}^3, is given by

$$C_{L_q} = \frac{1}{L_{pp}} \left(\frac{\bar{c}}{2} + 2\,(x_{hc} - x_m) \right) C_{L_\alpha} \,, \tag{3.24}$$

where $\bar{c}$ is the mean hydrodynamic chord, $x_{hc} - x_m$ is the longitudinal distance from the hydrodynamic center of the foil to the point of reference (positive aft), and C_{L_α} is the foil lift curve slope given by (3.1).

Body alone

The method presented in Datcom for estimating the body contribution to C_{L_q}, is based on slender body theory. Based on L_{pp}^3, and referred to the center of rotation, the body contribution to C_{L_q} is given by

$$C_{L_q} = 2\,C_{L_\alpha} \left(1 - \frac{x_m}{L_{pp}} \right) \cdot \frac{S_{base}}{L_{pp}^2} \,, \tag{3.25}$$

where C_{L_α} is the body lift curve slope given by (3.2), S_{base} is the base area, and x_m is the longitudinal distance from the body nose vertex to the center of rotation, positive aft.

Foil-body combination

For a typical AUV design, the method presented in Datcom (1978) for estimating the pitching derivative for a foil body configuration, based on L^3_{pp}, is given by

$$\left(C_{L_q}\right)_{FB} = \left[K_{F(B)} + K_{B(F)}\right] \left(C_{L_q}\right)_F + \left(C_{L_q}\right)_B \,, \tag{3.26}$$

where

$\boldsymbol{K_{F(B)}}$ **and** $\boldsymbol{K_{B(F)}}$ are the appropriate foil-body interference factors obtained from (3.5) and (3.6) respectively.

$\left(\boldsymbol{C_{L_q}}\right)_{\boldsymbol{F}}$ is the contribution from the foil, obtained from (3.24).

$\left(\boldsymbol{C_{L_q}}\right)_{\boldsymbol{B}}$ is the contribution from the body to the pitching derivative, obtained from (3.25).

3.6 Pitching derivative, C_{m_q}

As a result of the lift increment, C_{L_q}, a pitching moment increment C_{m_q} is produced. The derivative is a measure of the pitching moment produced by rotational motion of the fluidframe about the spanwise axis, and is commonly referred as the pitch-damping derivative (Datcom 1978).

Foil alone

The semi-empirical method, presented in Datcom, for estimating the pitching increment for the foil alone (based on L^4_{pp}) is given by

$$C_{m_q} = -0.7\, c_{l_\alpha}\, \frac{\bar{c}^{\,2}}{L^4_{pp}}\, \Upsilon \cos\Lambda_{c/4}\,, \tag{3.27}$$

where

$$\Upsilon = \frac{\mathcal{AR}_e \left(\frac{\bar{x}}{2\,\bar{c}} + \frac{2\,\bar{x}^2}{\bar{c}^2}\right)}{\mathcal{AR}_e + 2\cos\Lambda_{c/4}} + \frac{1}{24}\left(\frac{\mathcal{AR}_e^3 \tan^2\Lambda_{c/4}}{\mathcal{AR}_e + 6\cos\Lambda_{c/4}}\right) + \frac{1}{8}\,,$$

and

$$\frac{\bar{x}}{\bar{c}} = \frac{x_{hc} - x_m}{\bar{c}}\,, \tag{3.28}$$

is the non-dimensional distance between the foil hydrodynamic center and the point of reference (based on mean hydrodynamic chord, $\bar{c}$). c_{l_α} is the two-dimensional lift curve slope, obtained from experimental results (e.g Abbot & Doenhoff (1959)); or by the relation given in (3.1a). It should be noted that (3.27) is strictly applicable for $\mathcal{AR}_e \in (1, 6)$. For aspect-ratios not in this range, the empirical factor, 0.7, should be modified according to the procedure recommended in (Datcom 1978, Section 7.1.1.2)

Body alone

The body contribution to C_{m_q} (based on L_{pp}^4) is given by

$$C_{m_q} = 2\,C_{m_\alpha} \cdot \frac{\left(1-\frac{x_m}{L_{pp}}\right)^2 \cdot S_{base} - \frac{\forall}{L_{pp}}\left(\frac{x_c - x_m}{L_{pp}}\right)}{\left(1-\frac{x_m}{L_{pp}}\right) \cdot S_{base} - \frac{\forall}{L_{pp}}}\,, \tag{3.29}$$

where C_{m_α} is given by (3.15), $\forall$ is the total body volume, S_b is the base area (area of the stern) and x_c is the longitudinal distance from the nose to the centroid of the volume:

$$x_c = \frac{1}{\forall}\int_0^{L_{pp}} x\,S(x)\,\mathrm{d}x\,.$$

Foil-body combination

For the foil-body configuration, the pitching derivative C_{m_q}, based on L_{pp}^4 and referred to the moment center, x_m, is given by

$$\left(C_{m_q}\right)_{FB} = \left(K_{F(B)} + K_{B(F)}\right) \cdot \left(C_{m_q}\right)_F + \left(C_{m_q}\right)_B\,, \tag{3.30}$$

where $\left(C_{m_q}\right)_F$ is given by (3.27), $\left(C_{m_q}\right)_B$ is given by (3.29), $K_{F(B)}$ and $K_{B(F)}$ are the interference factors obtained from (3.5) and (3.6) respectively.

3.7 Open loop transfer functions

To study the dynamics for an AUV in the vertical plane, it is sufficient to compute and analyze the transfer functions from stern plane deflection to pith and heave motions[2] (Barros et al. 2004). If we neglect the surge equation, and take the Laplace transform of α, θ, and δ, the nondimensional linearized models in the vertical plane (about steady forward motion at trimming speed) are given by

$$\left(\frac{(m'-Z'_{\dot w})\,L_{pp}}{U}\,s - Z'_\alpha\right)\alpha(s) - \left(\frac{(Z'_q+m'\,x'_G)\,L_{pp}^2}{U^2}\,s^2 + \frac{(Z'_q+m')\,L_{pp}}{U}\,s\right)\theta(s) = C_{L_\delta}\delta(s) \tag{3.31}$$

and

$$\left(\frac{(I'_{yy}-M'_{\dot q})\,L_{pp}^2}{U^2}\,s^2 - \frac{(C_{m_q}-m'\,x'_G)\,L_{pp}}{U}\,s - M'_\theta\right)\theta(s) - \left(\frac{(M'_{\dot w}+m'\,x'_G)\,L_{pp}}{U}\,s + C_{m_\alpha}\right)\alpha(s) = C_{m_\delta}\delta(s)\,, \tag{3.32}$$

where the prime denotes the nondimensional value of the parameters involved (see Table 3.1 for additional information). Finally, if we let $\dot z'(s) = s\,z'_0(s)$ denote the Laplace transform of the depth rate in non dimensional form, the linearized depth rate about trimming is given by

$$\dot z'(s) = \alpha(s) - \theta(s)\,. \tag{3.33}$$

The transfer functions $\frac{\alpha}{\delta}(s)$ and $\frac{\theta}{\delta}(s)$ are provided in Appendix A.5.

[2]Note that all equations and theory, and most of the text in this section, is adopted directly from the work of Barros et al. (2004).

Parameter	Description	Parameter	Description
$m' = \frac{m}{0.5\,\rho L_{pp}^3}$	Vehicle mass	$Z'_{\dot{w}} = \frac{Z_{\dot{w}}}{0.5\,\rho L_{pp}^3}$	Added mass
$Z_\alpha = -(C_{D_0} + C_{L_\alpha})$	Body fixed drag	$Z'_{\dot{q}} = \frac{Z_{\dot{q}}}{0.5\,\rho L_{pp}^4}$	Added mass cross term
$x'_G = \frac{x_G}{L_{pp}}$	Coordinate of CG	$Z'_q = -(C_{L_q} + X'_{\dot{u}})$	Pitch-der. + added mass
$z'_G = \frac{z_G}{L_{pp}}$	Coordinate of CG	$I'_{yy} = \frac{I_{yy}}{0.5\,\rho L_{pp}^5}$	Rigid body inertia
$M'_{\dot{q}} = \frac{M_{\dot{q}}}{0.5\,\rho L_{pp}^5}$	Added inertia	$M'_{\dot{w}} = \frac{M_{\dot{w}}}{0.5\,\rho L_{pp}^4}$	Added mass cross term
$M'_\theta = \frac{-m\,g\,z_G}{0.5\,\rho\,L_{pp}^3\,U^2}$	Pitching stability	$X'_{\dot{u}} = \frac{X_{\dot{u}}}{0.5\,\rho L_{pp}^3}$	Axial added mass

Table 3.1: Nondimensional parameters, open loop transfer function

3.8 Discussion

In this chapter, we have presented theory necessary to investigate the expected performance of a typical AUV in the vertical plane. The expressions obtained are mainly based on Datcom (1978), which systematically has proven to yield expressions for lift, drag and hydrodynamic derivatives, for the foil-body combination. For an AUV, which typically have a similar configuration of control surfaces in the horizontal and vertical plane, it should be apparent that the theory presented in this chapter also could be used in the horizontal plane, by calculating the corresponding derivatives as a function of sideslip, β.

From Figure 3.4, it is seen that the mutual interference effects between the lifting surfaces and the body may be significant; this is especially true for an AUV, where the lifting surfaces typically are small in relation to the body size. It is therefor important to include the lifting-surface/body combination as units rather than in isolation.

Regarding (3.1b), it should be noted that some authors use the "extended aspect ratio" to calculate the lift-curve slope. However, to the best of the author's knowledge, Whicker & Fehlner (1958) do refer to the exposed area in their original formulation.

Datcom (1978) also include a systematic procedure for taking into account the effect of trailing vortices from a front "wing", and how it influence the effectiveness of the control surfaces at the stern; however, due to the vehicle to be considered, MAYA, the theory presented has been limited to vehicles with one set of control surfaces (in each plane). Datcom, do not provide theory that is applicable for investigating the interference between the body and duct, and therefore it has not been investigated.

Chapter 4

Example

In this chapter, a numerical example for a torpedo shaped AUV is presented. The example is based on the work of Barros et al. (2004), and is restricted to motion in the vertical plane (small perturbations). Furthermore, the example is linearized around a velocity $U_o = \sqrt{u^2 + v^2 + w^2} = 1.5\text{m/s}$. The shape of the vehicle considered is identical to the vehicle which is planned to undergo towing tank experiments in the *Marine Cybernetics Laboratory*[1] during the fall of 2004.

4.1 Vehicle

Before we proceed with calculation of the hydrodynamic derivatives, we will define the geometry of the vehicle, and calculate various parameters needed in calculations.

Body

The geometry of the body is given by the parameters in Table 4.1. Inserting numerical values into (2.4), (2.5) and (2.6), yields

$$r(x) = \begin{cases} \frac{0.20}{2}\left(1 - \left(\frac{x-0.20}{0.20}\right)^2\right)^{\frac{1}{3.0}} & , 0 \leq x \leq 0.20 \\ \frac{0.20}{2} & , 0.20 < x \leq 1.40 \\ r_t(x) & , 1.40 < x \leq 1.640 \end{cases} , \tag{4.1}$$

where

$$\begin{aligned} r_t(x) =& \frac{0.20}{2} - \left(\frac{3 \cdot 0.20}{2 \cdot (1.640 - 1.40)^2} - \frac{\tan 25.0^\circ}{1.640 - 1.40}\right)(x - 1.40)^2 \\ &+ \left(\frac{0.20}{(1.640 - 1.40)^3} - \frac{\tan 25.0^\circ}{(1.640 - 1.40)^2}\right)(x - 1.40)^3 . \end{aligned}$$

[1] Marine Cybernetics Laboratory is a laboratory which is a joint facility between department of Engineering Cybernetics and department of Marine Technology at the Norwegian University of Science and Technology (NTNU), and Marintek (Fossen 2004*b*).

d	0.20m
L	1.640m
L_{pp}	1.640m
a	0.20m
b	1.40m
n	3.0
θ	25.0°

Table 4.1: Body parameters (MAYA)

Foil

The vehicle is equipped with a NACA 0012 foil section (mounted in the xy-plane), where the normalized cross-sectional curvature of the profile is given by (Abbot & Doenhoff 1959)

$$\hat{y}(\varrho) = \frac{\pm 0.12}{0.2}\left(0.2969\sqrt{\varrho} - 0.126\varrho - 0.3516\varrho^2 + 0.2843\varrho^3 - 0.1015\varrho^4\right) , \quad (4.2)$$

where $\varrho \in (0, 1)$. Numerical values for most of the parameters used in calculations are presented in Table 4.2.

Figure 4.1: Cross-section of a NACA 0012 profile

4.2 Hydrodynamic derivatives

4.3 Foil alone

C_{L_α}

Using equation (3.1), the three dimensional lift curve slope (based on L_{pp}^2) is calculated:

$$\begin{aligned} C_{L_\alpha} &= \frac{S_e}{L_{pp}^2} \cdot \frac{2\,\pi\,\mathcal{R}_e}{2 + \frac{1}{\kappa}\sqrt{\frac{\mathcal{R}_e^2}{\cos^2\Lambda_{c/4}} + 4\cos^2\Lambda_{c/4}}} \\ &= \frac{1.80 \cdot 10^{-2}}{1.640^2} \cdot \frac{2\,\pi \cdot 3.20}{2 + \frac{1}{0.90}\sqrt{\frac{3.20^2}{\cos^2\Lambda_{c/4}} + 4\cos^2\Lambda_{c/4}}} \\ &= 21.00 \cdot 10^{-3} \quad \left[\mathrm{rad}^{-1}\right] . \end{aligned} \quad (4.3)$$

Description	Symbol	Expression	Value
Body			
Displaced volume	$\forall$	$\pi \int_0^{L_{pp}} r_{(x)}^2 \, \mathrm{d}x$	$45.76 \cdot 10^{-3}\mathrm{m}^3$
Wetted area	S_b	$2\,\pi \int_0^{L_{pp}} r_{(x)} \, \mathrm{d}x$	$0.95\mathrm{m}^2$
Fineness ratio	f	$\frac{L_{pp}}{d}$	8.20
Center of buoyancy	x_c	$\frac{\pi}{\forall} \int_0^{L_{pp}} x\, r_{(x)}^2 \, \mathrm{d}x$	0.781m
	x_0	a	0.20m
	S_0	$\frac{\pi\, d^2}{4}$	$3.14 \cdot 10^{-2}\mathrm{m}^2$
	V_0	$\int_0^{x_0} \pi\, r_{(x)}^2 \, \mathrm{d}x$	$4.64 \cdot 10^{-3}\mathrm{m}^3$
Base diameter	d_b	$2 \cdot r_{(L_{pp})}$	0.0m
Base area	S_e	$\frac{\pi d_b^2}{4}$	$0.0\mathrm{m}^2$
Foil			
Foil span	b		0.44m
Root chord	c_r		0.09m
Tip chord	c_t		0.06m
Leading edge	x_{le}		1.255m
Taper ratio	λ	$\frac{c_t}{c_r}$	0.67
Exposed foil area	S_e	$\frac{(b-d)\, c_r\, (1+\lambda)}{2}$	$1.80 \cdot 10^{-2}\mathrm{m}^2$
Wetted area, foil	S_{wf}	$S_e \cdot \int_0^1 \sqrt{\hat{y}(\varrho)^2 + \varrho^2} \, \mathrm{d}\varrho$	$3.67 \cdot 10^{-2}\mathrm{m}^2$
Aspect ratio	AR_e	$\frac{(b-d)^2}{S_e}$	3.20
Sweep angle at $c/2$	$\Lambda_{c/2}$		0.0°
Sweep angle at $c/4$	$\Lambda_{c/4}$	$\arctan \frac{4 \cdot 0.25}{\mathit{AR}_e} \cdot \frac{1-\lambda}{1+\lambda}$	3.58°
Sweep angle, leading edge	Λ_{le}	$\arctan \frac{4 \cdot 0.5}{\mathit{AR}_e} \cdot \frac{1-\lambda}{1+\lambda}$	7.13°
Average thickness ratio	$\frac{t}{c}$	$2 \cdot \int_0^1 \hat{y}(\varrho) \, \mathrm{d}\varrho$	0.082

Table 4.2: MAYA particulars

C_{L_q}

In order to calculate the pitching derivative (3.24), the hydrodynamic center of the foil configuration need to be determined. It is known that the cross-sectional hydrodynamic center is located at $c/4$ for a NACA-0012 configuration (Abbot & Doenhoff 1959, Datcom 1978). Furthermore, the spanwise location of the mean hydrodynamic chord, y_{hc}, is equivalent to the spanwise location of the centroid of area, which can be expressed as (Roskam 1979)

$$\begin{aligned} y_{hc} &= \frac{b-d}{2} + \frac{(1+2\,\lambda)(b-d)}{6\,(1+\lambda)} \\ &= 0.156\text{m}\,. \end{aligned} \tag{4.4}$$

Thus, it is seen from Figure 4.2 that the hydrodynamic center of the foil alone is given by

$$\begin{aligned} x_{hc,f} &= x_{le} + \frac{2\,y_{hc}-d}{2}\tan\Lambda_{le} + \frac{\bar{c}}{4} \\ &= 1.255\text{m} + 0.007\text{m} + 0.019\text{m} \\ &= 1.281\text{m}\,, \end{aligned} \tag{4.5}$$

where $\bar{c} = 0.076$m is the chord-length at y_{hc}.

Using (3.24), the pitching derivative for the foil alone (based on L_{pp}^3) is calculated:

$$\begin{aligned} C_{L_q} &= \frac{1}{L_{pp}}\left(\frac{\bar{c}}{2} + 2\,\left(x_{hc,f} - x_m\right)\right) C_{L_\alpha} \\ &= \frac{1}{1.640}\left(\frac{0.076}{2} + 2\cdot(1.281-0.781)\right)\cdot 21.00\cdot 10^{-3} \\ &= 13.30\cdot 10^{-3} \quad \left[\text{rad}^{-1}\right]\,. \end{aligned} \tag{4.6}$$

C_{m_q}

Noting that the aspect-ratio of the exposed foil, AR_e, is in the range where (3.24) is applicable, a numerical estimate for C_{m_q} (based on L_{pp}^4) for the foil alone is obtained by

$$\begin{aligned} C_{m_q} &= -0.7\,c_{l_\alpha}\,\frac{S_e\cdot\bar{c}^{\,2}}{L_{pp}^4}\,\Upsilon\cos\Lambda_{c/4} \\ &= -0.7\cdot 6.30\cdot\frac{1.80\cdot 10^{-2}\cdot 0.076^2}{1.640^4}\cdot 55.6\cdot\cos\Lambda_{c/4} \\ &= -3.52\cdot 10^{-3} \quad \left[\text{rad}^{-1}\right]\,, \end{aligned} \tag{4.7}$$

where $\Upsilon = 55.6$ (see Chapter 3.6).

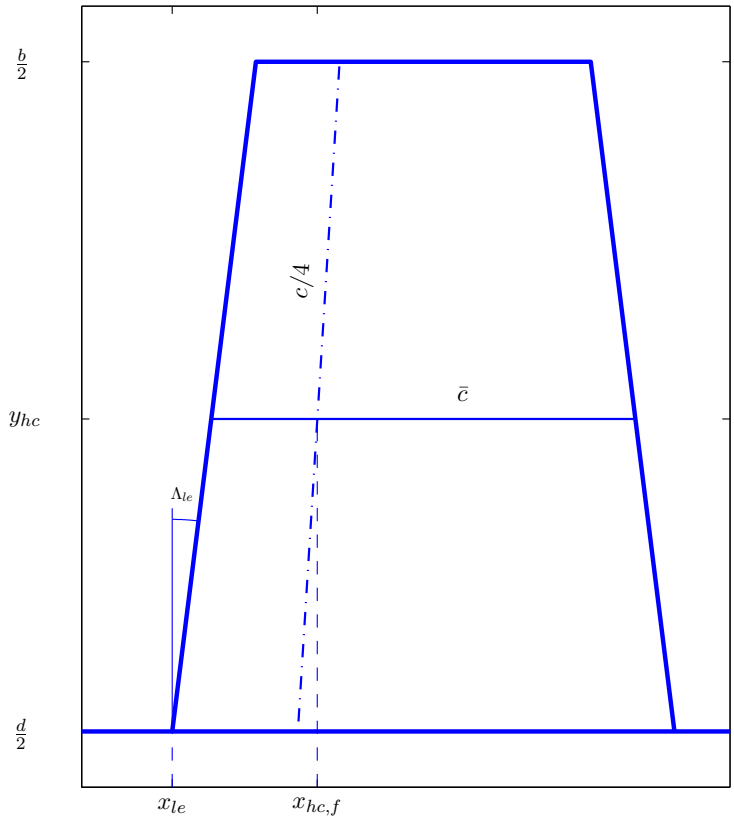


Figure 4.2: Hydrodynamic center foil

C_{D_0}

The average thickness ratio, t/c, of the NACA-0012 foil section is simply the area of (4.2):

$$\frac{t}{c} = 2 \cdot \int_0^1 \hat{y}(\varrho)\, \mathrm{d}\varrho \simeq 0.082\,. \tag{4.8}$$

Solving the equation $\frac{\mathrm{d}\hat{y}(\varrho)}{\mathrm{d}\varrho} = 0$, with respect to ϱ, yields the chordwise position of maximum thickness, $x_t \simeq 0.300$, which determines the numerical value of the hydrofoil thickness location parameter, $L = 1.2$. The lifting surface correction factor $R_{LS} = 1.07$ is obtained from Figure 3.7 (for $\cos 2.86^\circ \simeq 1.00$). The turbulent flat plate coefficient, $C_f = 0.009$, is obtained from Figure 3.6 for a Reynolds number, $\mathrm{R}_n = U\bar{c}/\nu = 7.31 \cdot 10^4$, based on the mean hydrodynamic chord, $\bar{c} = 0.076$m, advance speed $U = U_o = 1.5$m/s, and a kinematic viscosity $\nu = 1.56 \cdot 10^{-6}$ (water with 5% salinity at 5°C).

Inserting numerical values into (3.13), yields:

$$\begin{aligned} C_{D_0} &= C_f \left[1 + L\left(\frac{t}{c}\right) + 100\left(\frac{t}{c}\right)^4\right] R_{LS} \frac{S_{wf}}{L_{pp}^2} \\ &= 0.009 \cdot \left[1 + 1.2 \cdot 0.082 + 100 \cdot 0.082^4\right] \cdot 1.07 \cdot \frac{3.67 \cdot 10^{-2}}{1.640^2} \\ &= 14.75 \cdot 10^{-5} \quad \left[\text{rad}^{-1}\right] . \end{aligned} \tag{4.9}$$

4.3.1 Summary

Parameter	Value	k	Description
C_{L_α}	$21.00 \cdot 10^{-3}$	2	Lift curve slope
C_{L_q}	$13.30 \cdot 10^{-3}$	3	Pitching derivative
C_{m_q}	$-3.52 \cdot 10^{-3}$	4	Pitching derivative
C_{D_0}	$14.75 \cdot 10^{-5}$	2	Zero lift drag

Table 4.3: Foil alone, hydrodynamic coefficients (based on L_{pp}^k)

4.4 Body alone

C_{L_α}

For a fineness ratio, $f = L_{pp}/d = 8.20$, the apparent mass factor, $k_2 - k_1 = 0.91$, is obtained from Figure 3.1. As discussed in Chapter 3.1.2, the cross-sectional area at the point where the flow cease to be potential, for a torpedo shaped vehicle, is given by

$$S_0 = \frac{\pi d^2}{4} \simeq 3.14 \cdot 10^{-2} \text{m}^2 . \tag{4.10}$$

Then, using (3.2), the lift curve slope (based on L_{pp}^2) for the body alone is calculated:

$$\begin{aligned} C_{L_\alpha} &= \frac{2 \cdot (k_2 - k_1) \cdot S_0}{L_{pp}^2} \\ &= \frac{2 \cdot 0.91 \cdot 3.14 \cdot 10^{-2}}{1.640^2} \\ &= 21.33 \cdot 10^{-3} \quad \left[\text{rad}^{-1}\right] . \end{aligned} \tag{4.11}$$

C_{L_q} and C_{m_q}

As can be seen from (4.1), the base area of the vehicle, S_{base}, is zero; thus, from (3.25) and (3.29) it follows that the contribution for the body alone to C_{L_q} and C_{m_q} is zero.

C_{D_0}

From Figure 3.6, for a Reynolds number, $\mathrm{R}_n = UL_{pp}/\nu = 1.58 \cdot 10^6$, based on $L_{pp} = 1.640$m, advance speed $U = U_o = 1.5$m/s, and a kinematic viscosity $\nu = 1.56 \cdot 10^{-6}$ (water with 5% salinity at 5°C), the turbulent flat plate coefficient, $C_f = 4.256 \cdot 10^{-3}$, is obtained. Because the base diameter (d_b) is zero, there is no contribution from the "base" to the total body drag ($C_{D_b} = 0$). From Table 4.2 we obtain the wetted area of the body, $S_b = 0.95\mathrm{m}^2$, and referring to (3.9), the zero-lift drag of the body is

$$\begin{aligned}\left(C_{D_f}\right)_b &= C_f \left[1 + \frac{60\,d^3}{L_{pp}^3} + 0.0025\frac{L_{pp}}{d}\right] \frac{4\,S_b}{\pi\,d^2} \\ &= 4.256 \cdot 10^{-3} \cdot \left[1 + \frac{60 \cdot 0.20^3}{1.640^3} + 0.0025 \cdot \frac{1.640}{0.20}\right] \cdot \frac{4 \cdot 0.95}{\pi \cdot 0.20^2} \\ &= 14.52 \cdot 10^{-2} \quad \left[\mathrm{rad}^{-1}\right] .\end{aligned} \tag{4.12}$$

Thus, inserting numerical values into (3.9), yields an estimate of the zero-lift drag coefficient (based on L_{pp}^2):

$$\begin{aligned}C_{D_0} &= \left\{\left(C_{D_f}\right)_b + C_{D_b}\right\} \cdot \frac{\pi\,d^2}{4\,L_{pp}^2} \\ &= 1.70 \cdot 10^{-3} \quad \left[\mathrm{rad}^{-1}\right] .\end{aligned} \tag{4.13}$$

4.4.1 Summary

Parameter	Value	k	Description
C_{L_α}	$21.33 \cdot 10^{-3}$	2	Lift curve slope
C_{m_α}	$9.47 \cdot 10^{-3}$	3	Pitching moment
C_{L_q}	0.0	3	Pitching derivative
C_{m_q}	0.0	4	Pitching derivative
C_{D_0}	$1.70 \cdot 10^{-3}$	2	Zero lift drag

Table 4.4: Body alone, hydrodynamic coefficients (based on L_{pp}^k)

4.5 Foil body combination

Before we proceed with the calculation of the combination, we define

$$\begin{aligned}\left(C_{L_\alpha}\right)_{F(B)} &= K_{F(B)}\left(C_{L_\alpha}\right)_F \\ &= 1.40 \cdot 21.00 \cdot 10^{-3} \\ &= 29.5 \cdot 10^{-3}\end{aligned} \tag{4.14a}$$

$$\begin{aligned}\left(C_{L_\alpha}\right)_{B(F)} &= K_{B(F)}\left(C_{L_\alpha}\right)_F \\ &= 0.71 \cdot 21.00 \cdot 10^{-3} \\ &= 14.9 \cdot 10^{-3}\,,\end{aligned} \tag{4.14b}$$

where the interference factors are obtained from Figure 3.4, with numerical values given in Table 4.5.

Parameter	Value	Source
$K_{F(B)}$	1.40	Figure 3.4
$K_{B(F)}$	0.71	Figure 3.4
$k_{F(B)}$	0.94	Figure 3.5
$k_{B(F)}$	0.47	Equation (3.8b)

Table 4.5: Interference factors ($\varsigma = d/b = 0.45$)

C_{L_α}

From (3.3), the lift curve slope for the combination is

$$\begin{aligned}(C_{L_\alpha})_{FB} &= (C_{L_\alpha})_B + (C_{L_\alpha})_{F(B)} + (C_{L_\alpha})_{B(F)} \\ &= 21.33 \cdot 10^{-3} + 29.5 \cdot 10^{-3} + 14.9 \cdot 10^{-3} \\ &= 65.76 \cdot 10^{-3} \quad \left[\text{rad}^{-1}\right] .\end{aligned} \tag{4.15}$$

C_{L_δ}

From (3.7), the lift due to foil deflection is

$$\begin{aligned}(C_{L_\delta})_{FB} &= \left[k_{F(B)} + k_{B(F)}\right] \cdot (C_{L_\alpha})_F \\ &= [0.94 + 0.47] \cdot 21.00 \cdot 10^{-3} \\ &= 29.48 \cdot 10^{-3} \quad \left[\text{rad}^{-1}\right] ,\end{aligned} \tag{4.16}$$

where the interference factors are obtained from Table 4.5.

C_{m_α}

Body alone Inserting numerical values into (3.15), yields an estimate (based on L_{pp}^3) of the body pitching moment:

$$\begin{aligned}(C_{m_\alpha})_B &= \frac{x_m\, C_{L_\alpha}}{L_{pp}} + 2\,(k_2 - k_1)\,\frac{V_0 - S_0\, x_0}{L_{pp}^3} \\ &= \frac{0.781 \cdot 21.33 \cdot 10^{-3}}{1.640} + 2 \cdot 0.91 \cdot \frac{4.64 \cdot 10^{-3} - 3.14 \cdot 10^{-2} \cdot 0.20}{1.640^3} \\ &= 9.47 \cdot 10^{-3} \quad \left[\text{rad}^{-1}\right] .\end{aligned} \tag{4.17}$$

Foil in presence of body Inserting numerical values into (3.17), an estimate of the hydrodynamic center of the foil in presence of the body is given by

$$\begin{aligned}\left(\frac{x_{hc}}{L_{pp}}\right)_{F(B)} &= \frac{1}{L_{pp}}\left(x_m - x_{le} - \frac{2\, y_{hc} - d}{2}\tan\Lambda_{le} - \frac{\bar{c}}{4}\right) \\ &= \frac{1}{1.640}\left(0.781 - 1.255 - \frac{2 \cdot 0.156 - 0.20}{2}\tan 7.13^\circ - \frac{0.076}{4}\right) \\ &= -0.305\,.\end{aligned} \tag{4.18}$$

Using (3.16), the body pitching moment contribution is estimated as

$$\begin{aligned}(C_{m_\alpha})_{F(B)} &= (C_{L_\alpha})_{F(B)} \left(\frac{x_{hc}}{L_{pp}}\right)_{F(B)} \\ &= 29.5 \cdot 10^{-3} \cdot -0.305 \\ &= -9.00 \cdot 10^{-3}\,.\end{aligned} \tag{4.19}$$

Body in presence of foil In order to calculate the hydrodynamic center of the body in presence of foil, we use the interpolation procedure described in Chapter 3.3, and plot the result in Figure 4.3 as a function of $\beta \mathcal{AR}_e$. Then, for $\beta \mathcal{AR}_e = 3.20$, $\left(\frac{x_{hc}}{L_{pp}}\right)_{B(F)} = -0.304$.

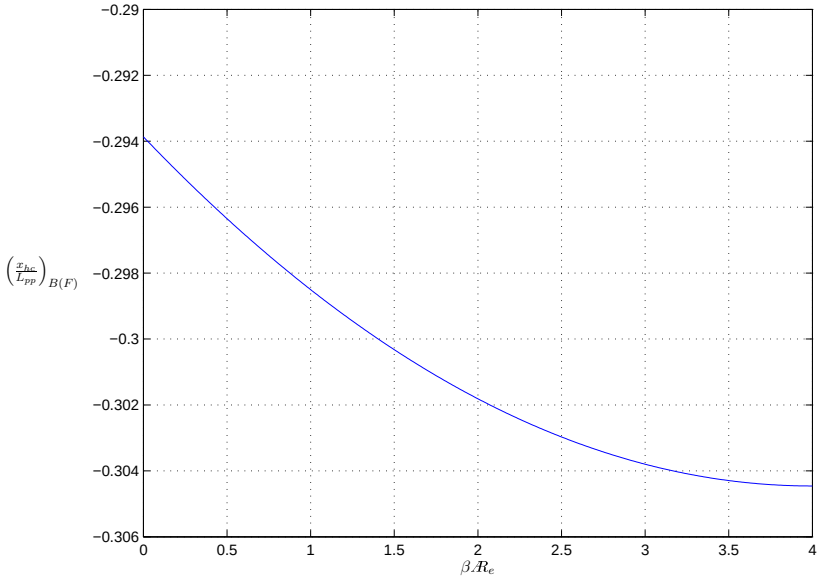


Figure 4.3: Plot of $\left(\frac{x_{hc}}{L_{pp}}\right)_{B(F)}$ versus $\beta \mathcal{AR}_e$

Using (3.22), the total pitching moment for the combination is estimated as

$$\begin{aligned}(C_{m_\alpha})_{FB} &= (C_{m_\alpha})_B + (C_{m_\alpha})_{F(B)} + (C_{m_\alpha})_{B(F)} \\ &= 9.47 \cdot 10^{-3} - 9.00 \cdot 10^{-3} - 4.54 \cdot 10^{-3} \\ &= -4.07 \cdot 10^{-3} \quad \left[\text{rad}^{-1}\right]\,.\end{aligned} \tag{4.20}$$

C_{m_δ}

Using (3.23), the pitching moment due to rudder deflection for the combination is obtained by

$$\begin{aligned}(C_{m_\delta})_{FB} &= \left(\left(\frac{x_{hc}}{L_{pp}}\right)_{B(F)} k_{B(F)} + \left(\frac{x_{hc}}{L_{pp}}\right)_{F(B)} k_{F(B)} \right) (C_{L_\alpha})_F \\ &= (-0.304 \cdot 0.47 - 0.305 \cdot 0.94) \cdot 21.00 \cdot 10^{-3} \\ &= -8.98 \cdot 10^{-3} \quad \left[\text{rad}^{-1}\right] .\end{aligned} \tag{4.21}$$

C_{L_q}

The total lift moment increment for the combination, is calculated by inserting numerical values into (3.26); thus

$$\begin{aligned}\left(C_{L_q}\right)_{FB} &= \left[K_{F(B)} + K_{B(F)}\right] \left(C_{L_q}\right)_F + \left(C_{L_q}\right)_B \\ &= [1.40 + 0.71] \cdot 13.30 \cdot 10^{-3} + 0 \\ &= 28.14 \cdot 10^{-3} \quad \left[\text{rad}^{-1}\right] .\end{aligned} \tag{4.22}$$

C_{m_q}

Inserting numerical values into (3.26), yields

$$\begin{aligned}\left(C_{m_q}\right)_{FB} &= \left(K_{F(B)} + K_{B(F)}\right) \cdot \left(C_{m_q}\right)_F + \left(C_{m_q}\right)_B \\ &= (1.40 + 0.71) \cdot -3.52 \cdot 10^{-3} + 0 \\ &= -7.44 \cdot 10^{-3} \quad \left[\text{rad}^{-1}\right] .\end{aligned} \tag{4.23}$$

C_{D_0}

The total skin friction drag for the combination, is estimated by inserting numerical values into (3.14); thus

$$\begin{aligned}(C_{D_0})_{FB} &= \left\{ (C_{D_0})_F + (C_f)_b \, \frac{\pi\, d^2}{4\, L_{pp}^2} \right\} \cdot R_{FB} + C_{D_b} \cdot \frac{\pi\, d^2}{4\, L_{pp}^2} \\ &= \left\{ 14.75 \cdot 10^{-5} + 14.52 \cdot 10^{-2} \cdot \frac{\pi \cdot 0.20^2}{4 \cdot 1.640^2} \right\} \cdot 1.05 + 0 \cdot \frac{\pi \cdot 0.20^2}{4 \cdot 1.640^2} \\ &= 1.94 \cdot 10^{-3} \quad \left[\text{rad}^{-1}\right] .\end{aligned} \tag{4.24}$$

4.5.1 Summary

Parameter	Value	k	Description
C_{L_α}	$65.76 \cdot 10^{-3}$	2	Lift curve slope
C_{L_δ}	$29.48 \cdot 10^{-3}$	2	
C_{m_α}	$-4.07 \cdot 10^{-3}$	3	
C_{m_δ}	$-8.98 \cdot 10^{-3}$	3	
C_{L_q}	$28.14 \cdot 10^{-3}$	3	Pitching derivative
C_{m_q}	$-7.44 \cdot 10^{-3}$	4	Pitching derivative
C_{D_0}	$1.94 \cdot 10^{-3}$	2	Zero lift drag

Table 4.6: Hydrodynamic coefficients, foil body combination, (based on L_{pp}^k)

4.6 Added mass

Taking into account that we are only looking into the vertical plane, the added mass matrix is written as

$$\boldsymbol{M}_A = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Z_{\dot{w}} & -Z_{\dot{q}} \\ 0 & -M_{\dot{w}} & -M_{\dot{q}} \end{bmatrix}, \tag{4.25}$$

where the nonzero coefficients are estimated below. Using the assumption that $\boldsymbol{M}_A = \boldsymbol{M}_A^{\mathrm{T}}$, indicate that $M_{\dot{w}} = Z_{\dot{q}}$.

As discussed in Chapter 2.3.3, an estimate of the added mass in surge might-be obtained by fitting an ellipsoid within the hull. From Figure 2.7, for a fineness ratio $f = 8.20$, we obtain $\varpi = 0.23$. Then from (2.11), the axial added mass (surge) is estimated by

$$\begin{aligned} -X_{\dot{u}} &= \frac{4\varpi\rho\pi}{3}\left(\frac{d}{2}\right)^3 \\ &= \frac{4 \cdot 0.23 \cdot 1025 \cdot \pi}{3}\left(\frac{0.20}{2}\right)^3 \\ &= 0.99\mathrm{kg}\,. \end{aligned}$$

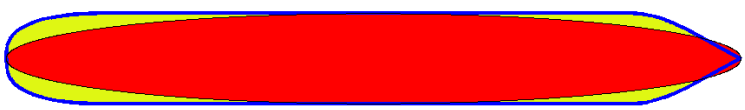


Figure 4.4: Ellipsoid fitted within the hull

An estimate of the parameter $Z_{\dot{w}}$ might be obtained by integrating (2.13) and (2.13) over the length of the vehicle. However, due to the fact that is very difficult to obtain an explicit solution of the integral of (4.1), we will use estimates obtained from hde (see Chapter 5 for more details) as a solution to the integral

$$\begin{aligned} -Z_{\dot{w}} &= \int_0^{L_{pp}} m_{a,b}(x)\,\mathrm{d}x + \int_{L_{pp}}^{L_{pp}+c_r} m_{a,f}(x)\,\mathrm{d}x \\ &\simeq 53.87\mathrm{kg}\,. \end{aligned}$$

In the same way as for $Z_{\dot{w}}$, we use the estimate from hde as the solution to the other terms. The estimates are then

$$\begin{aligned} -Z_{\dot{q}} = -M_{\dot{w}} &= \sum_{i=1}^{N-1} (\hat{x} - x_m)\cdot \delta_x\, m^{xy}_{i(\hat{x})} \\ &\simeq 4.06\mathrm{kg\,m}\,, \\ -M_{\dot{q}} &= \sum_{i=1}^{N-1} (\hat{x} - x_m)^2\cdot \delta_x\, m^{xy}_{i(\hat{x})} \\ &\simeq 10.07\mathrm{kg\,m^2}\,, \end{aligned}$$

where the coefficients are as defined in Chapter 2.3.3.

4.7 Open loop simulation

If we assume a neutrally buoyant vehicle, with $m = 46.90$kg, and a uniform density over the volume[2]. Then, if we non-dimensionalize the coefficients according to Table 3.1, we obtain the non-dimensional coefficients in Table 4.7. Inserting numerical values into the transfer functions (A.8) and (A.9) yields

$$\frac{\alpha}{\delta}(s) = \frac{1.2864\cdot s^2 + 5.135\cdot s + 0.88884}{1.7768\cdot s^3 + 8.8697\cdot s^2 + 10.1292\cdot s + 2.041}$$

and

$$\frac{\theta}{\delta}(s) = \frac{-7.8856\cdot s + 5.135}{1.7768\cdot s^3 + 8.8697\cdot s^2 + 10.1292\cdot s + 2.041}\,.$$

The expected performance of the vehicle in the vertical plane is shown in Figure 4.5 and 4.6, where the results are obtained from a step response simulation, with $\delta = 1^\circ$, at $t' = 5$.

4.8 Discussion

In this chapter, we have analyzed the expected performance of the MAYA AUV. The example is based on the work presented in Barros et al. (2004), but here we have avoided the simplifications by using the transfer functions developed in the previous chapter. From Figure 4.6, it is seen that the negative C_{m_α} stabilize the vehicles angle of attack.

[2] except for a point mass, necessary to obtain a meta-centric stable vehicle

Parameter	Description	Parameter	Description
$m' = 20.747 \cdot 10^{-3}$	Vehicle mass	$Z'_{\dot{w}} = -23.828 \cdot 10^{-3}$	Added mass
$Z_\alpha = -67.692 \cdot 10^{-3}$	Body fixed drag	$Z'_{\dot{q}} = -1.094 \cdot 10^{-3}$	Added mass cross term
$x'_G = 0$	Coordinate of CG	$Z'_q = -27.705 \cdot 10^{-3}$	Pitch-der. + added mass
$z'_G = 12.195 \cdot 10^{-3}$	Coordinate of CG	$I'_{yy} = 1.388 \cdot 10^{-3}$	Rigid body inertia
$M'_{\dot{q}} = -1.657 \cdot 10^{-3}$	Added inertia	$M'_{\dot{w}} = -1.094 \cdot 10^{-3}$	Added mass cross term
$M'_\theta = -1.809 \cdot 10^{-3}$	Pitching stability	$X'_{\dot{u}} = -0.438 \cdot 10^{-3}$	Axial added mass

Table 4.7: Nondimensional parameters, open loop transfer function

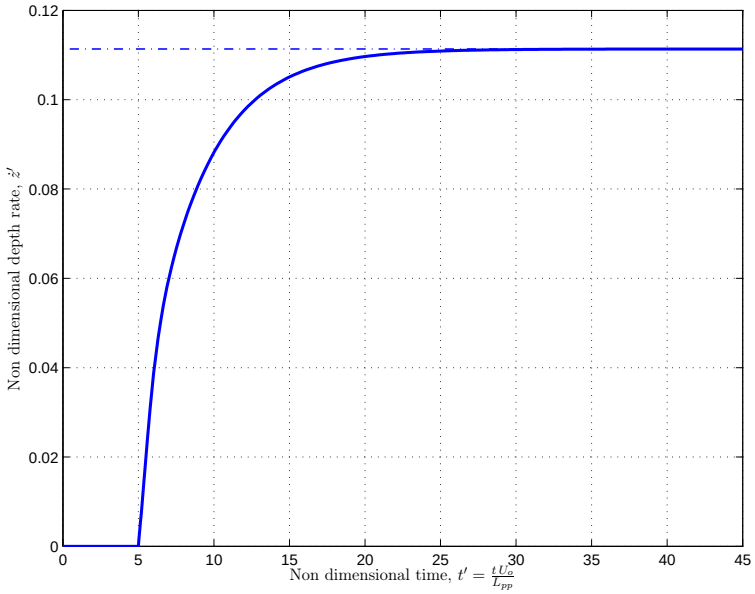


Figure 4.5: Open loop simulation, depth rate

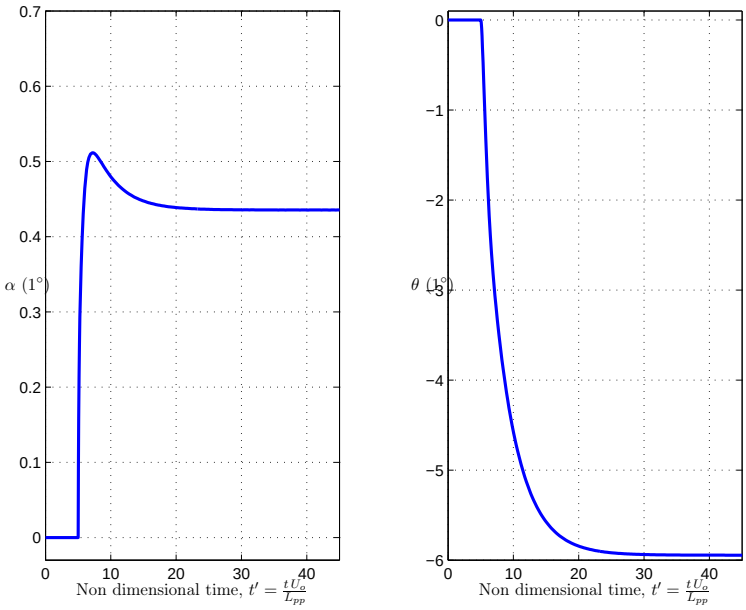


Figure 4.6: Open loop simulation, pitch and angle of attack

Chapter 5

Hydrodynamic software

5.1 MATLAB Toolbox- hde

As a part of this work, a MATLAB toolbox for hydrodynamic estimation of a body of revolution has been developed. The application estimates the coefficients discussed in Chapter 3, for both the horizontal and vertical plane. In addition, it also estimates properties such as added mass and rigid body dynamics (optional).

5.1.1 Running the application

Prior to running the application, the toolbox need to be loaded into your MATLAB search path. This is accomplished by running the *hdeinstall.m* script from your MATLAB Command Window (provided in Appendix D). Then, by typing *hde* in the MATLAB Command Window, a startup menu identical to Figure 5.1 appears.

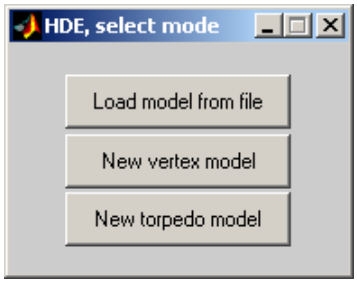


Figure 5.1: hde- start menu

Load model from Open and edit a previous hde session.

New vertex model An interface where the user can place vertexes, which determine the curve of revolution by cubic hermite interpolation. By pushing the *Create Geometry* button, a body of "any shape" can be drawn.

New torpedo model An interface where the shape is determined by the Myring (1976) parameters outlined in Section 2.2.2.

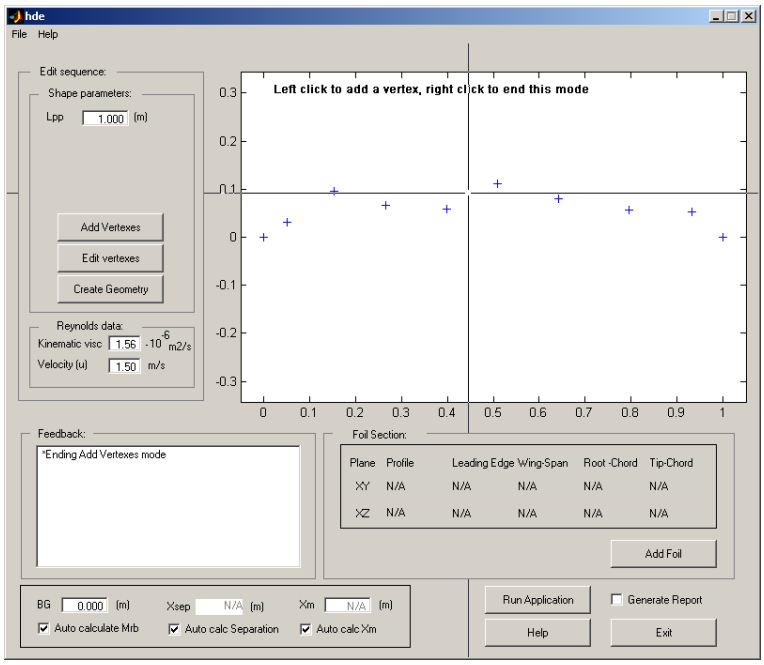


Figure 5.2: hde- Creating a vertex model

Adding a foil section

After the shape of the body has been specified and the *Create Geometry* button has been pressed, foil sections are added by pushing the *Add Foil* button. Then a dialog box, identical to Figure 5.3 appear, where all variables except for *Density* are as defined in Chapter 2.2.1. Density is simply the density of the material used in the foil sections (hde assumes aluminium by default).

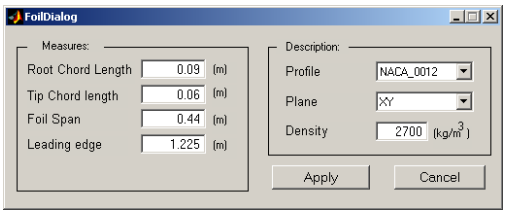


Figure 5.3: hde- Foil dialog

Specifying parameters

Referring to Figure 5.2 and 5.4, the other user defined options are as specified in Table 5.1.

TAG	Description
Auto calc Mrb	If checked, hde automatically estimates the rigid body system inertia matrix, $\boldsymbol{M}_{RB}$, assuming a neutrally buoyant vehicle. Deselecting this checkbox makes it possible for the user to specify the rigid body dynamics.
BG	Desired metacentric height (assuming *Auto calc Mrb* is checked)
Auto calc Separation	If checked, hde estimates the point where the flow cease to be potential according to the theory presented in Section 3.1.2
Xsep	Point where flow cease to be potential (relative to nose vertex)
Kinematic visc	Kinematic viscosity, ν, used in calculations
Velocity (u)	Velocity used in estimation of linearized derivatives
Add Vertexes	Used to place vertexes which determine the shape of a vertex model
Edit Vertexes	Opens a dialog box, where the user can edit the coordinates of vertexes
Lpp	Length between the perpendiculars, L_{pp}
L	Myring parameter as outlined in Section 2.2.2
Ln	Length of nose section, a
Lt	Distance from nose vertex to the start of the tail, b
D	Diameter of body, d
theta	Half of the included tail angle, θ
n	Parameter which determines the shape of the nose
Run Application	Estimate lift, drag, added mass etc.
Help	Executes the hde toolbox help functions
Generate Report	Automatically generate a report of the estimated parameters and derivatives (requires MikTex (2003) to be installed)

Table 5.1: hde- user defined parameters

5.1.2 Viewing the output

There are two different ways to view and analyze the output from hde, in MATLAB and in an automatically generated report.

MATLAB

The information collected and generated by the application is contained within structures. After running the application a datafile called *output.mat* is stored in the active directory. If the user type *load output* in the MATLAB Command Window, two different structures, *Foil* and *Body*, are loaded into the workspace. These structures contains all the estimates and information collected through the session.

The report

Assuming MikTex (2003) and a *pdf* viewer are installed on your system, hde is capable of presenting the estimates in an automatically generate pdf file. This feature was created by preparing a Latex text-file from MATLAB, where links to

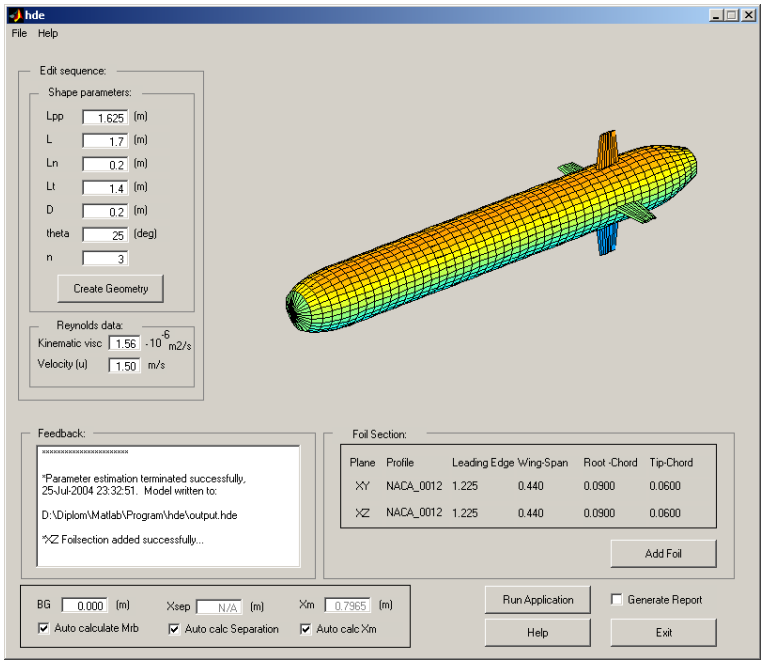


Figure 5.4: hde- Torpedo model

numerical values are written as tags[1]. In the latex source code, the file created by MATLAB is imported, and the parameters, addressed by theirs tags, are dynamically updated during compilation.

5.2 Simulation environment

In order to make it possible to simulate and examine different designs, a nonlinear MATLAB Simulink interface, *hdesim.mdl*, which builds on the work reported in Nahon (1996), has been implemented. The interface, import an hde model, and calculates the hydrodynamic forces directly from known relations (lift and drag) which govern the flow around the body and foil sections (hereafter referred to as reference points). A summary of the procedure is given below.

1. At each iteration, the velocity at each reference point is calculated according to the velocity kinematics

$$V_i = \sqrt{(u + p\,z_i - r\,y_i)^2 + (v + r\,x_i - p\,z_i)^2 + (w + p\,y_i - q\,x_i)^2}\,,$$

   where x_i, y_i and z_i are the body fixed coordinates of the hydrodynamic center for reference point i.

2. The angle of attack and sideslip angles are calculated at each reference

[1] using the \newcommand{\TAG}{<*value*>} feature in Latex.

point according to

$$\alpha_i = \arctan\left(\frac{w + p\,y_i - q\,x_i}{u + q\,z_i - r\,y_i}\right), \qquad \beta_i = \arcsin\left(\frac{v + r\,x_i - p\,z_i}{V_i}\right).$$

3. The lift coefficients are calculated at each reference point according to the procedures described in Chapter 3. If necessary, the lift is limited by the flat stall characteristic described in Section 3.1.1.

4. The drag coefficients are calculated at each control point according to the procedures described in Chapter 3. In addition, lift induced drag from the control surfaces are added according to the procedure described in McCormick (1994). Thus,

$$C_{D_i} = C_{D_{0,i}} + \frac{C_{L,i}^2}{\pi\,\text{Æ}_i\,e},$$

where C_{D_0} is the parasite drag, and e is the Oswald efficiency factor (implemented as $e = 0.87$). To avoid drag flatten when the planes operates past the stall point, the lift coefficient used in the equation above, is the one obtained obtained prior to incorporating stall effects.

5. Each component force is dimensionalized by the dynamic pressure and reference area ($\frac{1}{2}\,\rho\,V_i^2\,L_{pp}^2$), and transformed into the body axes.

6. The forces and moments acting on the vehicle are summed together, making it possible to obtain an estimate of the acceleration vector, $\dot{\boldsymbol{\nu}}$.

7. Integrating $\dot{\boldsymbol{\nu}}$ in Simulink, yields an estimate of the velocity vector $\boldsymbol{\nu}$, which in turn is transformed into $\dot{\boldsymbol{\eta}}$ through the transformation

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu},$$

where $\boldsymbol{J}(\,\cdot\,)$ is the transformation matrix defined in Appendix A.3.

8. Integrating $\dot{\boldsymbol{\eta}}$ yields an estimate of the inertial position vector, $\boldsymbol{\eta}$.

5.2.1 hde Simulink interface

The Simulink interface, shown in Figure 5.5, may be used for real time simulation of hde models. In order to use the model, there are a few matters the user should be aware of. First, in order to load a model into the *hdeSim* environment, the user need to push the *Load HDE model* button; which will open a dialog where the user can specify a saved hde model. Second, the vehicle used in simulation need control surfaces in both the xy and xz plane in order to have a stable vehicle, which is necessary if the user want reasonable results from simulation. Third, the simulation environment is not expected to work very well at velocities close to zero, or at large sideslip/ angle of attacks (α, $\beta \geq 50°$), because of limitation in theory used in implementation.

Referring to Figure 5.5, the block labelled *Oceanic disturbances* is simply a first order Gauss-Markov Process, which can be used in simulation (optional) for testing the robustness of controllers. The subsystem *AUV*, shown in Figure 5.6, is used as an interface against the simulation procedure described above. This subsystem contains some features from the *Marine GNC-toolbox* (Fossen 2003).

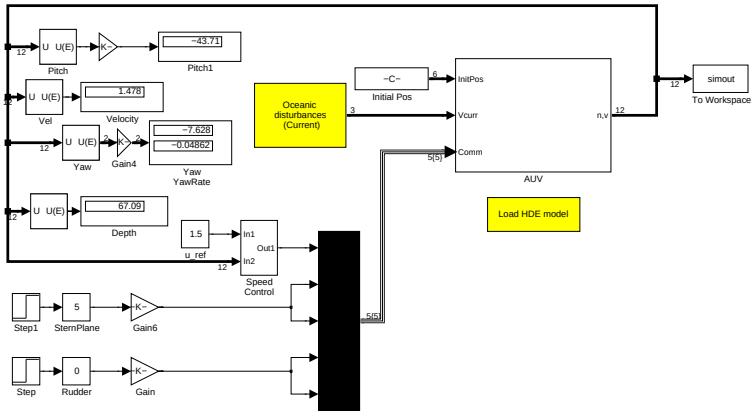


Figure 5.5: hde Simulink interface, hdeSim

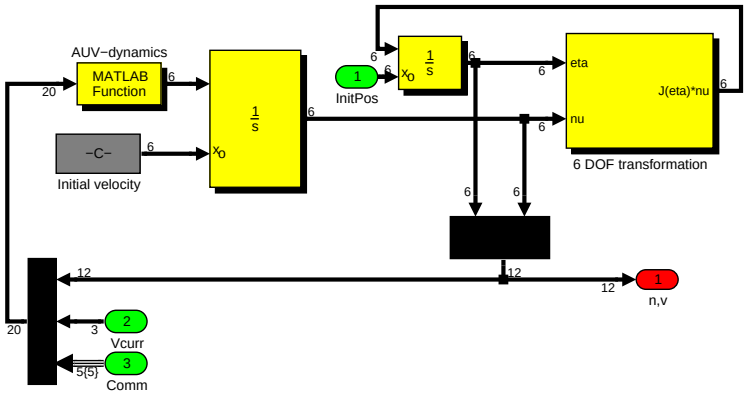


Figure 5.6: hde, AUV model

5.2.2 Open loop simulations

In order to verify the procedure described above, we construct an hde model of the REMUS AUV, which is a Myring torpedo profile (see Figure 5.8), and compare the simulated results with experimental results reported in Prestero (2001). The shape and particulars of the vehicle are approximately given by the parameters in Table 5.2.

Body		Foils		Mass and inertia	
a	0.191m	x_{le}	1.207m	m	30.48kg
b	0.845m	c_r	0.089m	z_{cg}	0.02m
L	1.386m	c_t	0.059m	I_{xx}	0.177kgm^2
L_{pp}	1.349m	b	0.262m	I_{yy}	3.45kgm^2
n	2			I_{zz}	3.45kgm^2
θ	25°				
d	0.191m				

Table 5.2: REMUS particulars

Prestero (2001) report a yaw-rate of approximately $-10°/s$ when the REMUS AUV is exposed to a rudder step input, $\delta_r \simeq 4°$ at a surge velocity, u, of approximately 1.5m/s. From simulation in the hde environment, we obtain Figure 5.7 and an estimated yaw rate of approximately $-9°/s$.

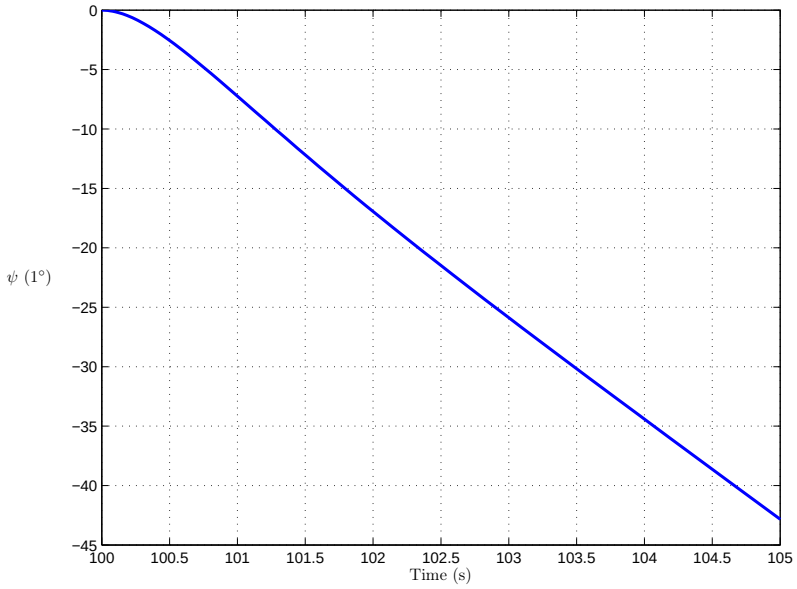


Figure 5.7: Simulation of yaw step response, $\delta_r = 4°$

Prestero (2001), also include some other results from open loop experiments. However, because it has been difficult to compare these results with simulation, they are not included. The difficulties are mainly due to the fact that the author suspect that reported data of either the "weight and buoyancy" and/or inertias are not in agreement with the vehicle. The suspicion is based on Figure 7.8 in Prestero (2001), where we observe that the vehicle keeps a fixed depth when the pitch angle is zero and the stern planes are deflected approximately 4°. Looking at the physics, it indicate that the center of gravity probably is located aft the center of buoyancy.

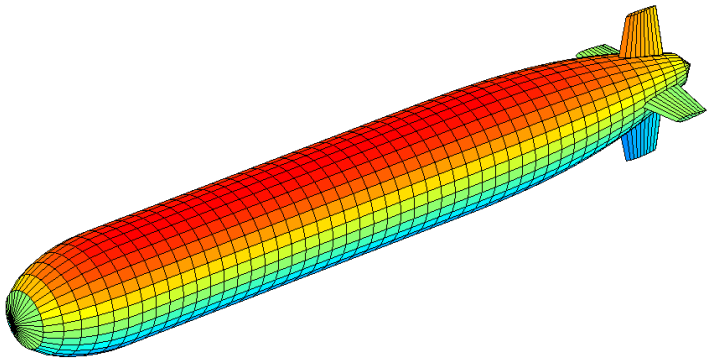


Figure 5.8: hde- model, REMUS

An open loop simulation of the vehicle discussed in Chapter 4, has also been carried out. The motivation for this particular simulation is simply to verify the results from simulation vs. the linearized transfer function. As in Figure 4.5, the results from simulation, Figure 5.9, are nondimensionalized. As we can see, the steady state nondimensional depth rate, $\dot{z}'$, is approximately 0.55 when the stern planes are deflected 5°.

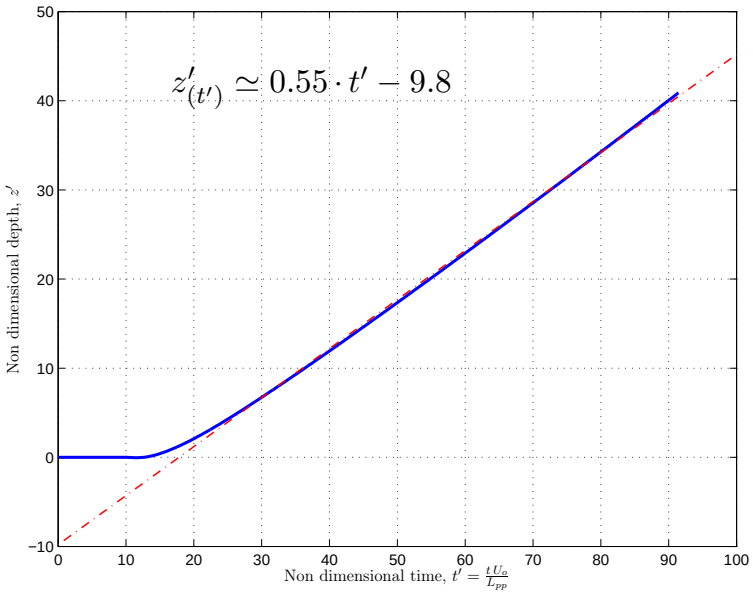


Figure 5.9: hde open loop simulation, step $\delta_e = 5°$ at $t' = 5$

5.3 WAMIT

As a part of this work, a number of WAMIT runs has been executed. The body which has been examined is identical to the one discussed in Chapter 4. As

seen in Figure 5.10, several user defined files need to be prepared in order to run WAMIT. All files, except for the MultiSurf CAD model (see Figure 5.11) used to describe the geometric input to WAMIT, are provided in Appendix C. The different runs, except for the one used to compare the output from hde with WAMIT in the next section, has been executed in an environment similar to the basin at the Marine Cybernetics Laboratory (Fossen 2004*b*). In the user prepared input-files, the vehicle has been submerged 0.75m and the water-depth has been specified to be 1.5m. The origin of the coordinate systems has been specified to be at the center of buoyancy (along the axis of revolution, 0.781m aft the nose vertex). The length used for nondimensionalizing is $L = L_{pp} = 1.64$m. All output files are provided in Appendix C.

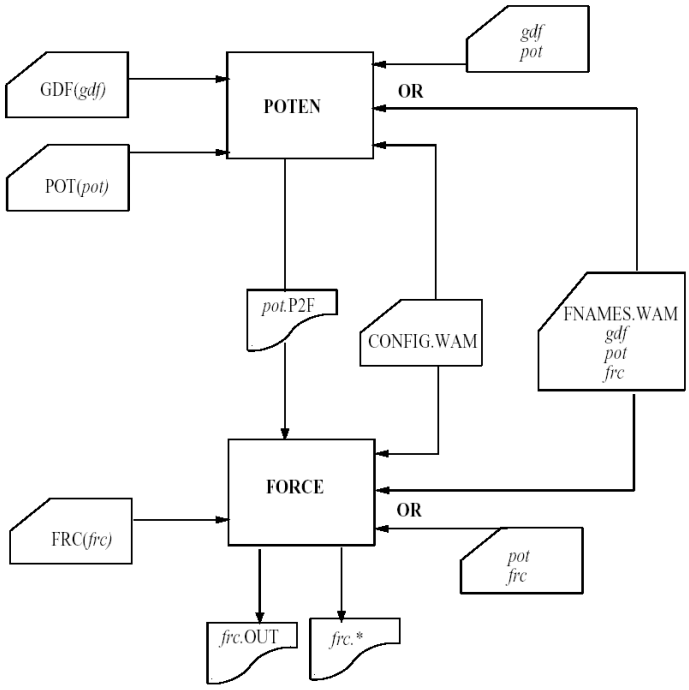


Figure 5.10: Flow chart of WAMIT with their associated input and output files (WAMIT-Inc 2003)

5.4 Comparison

In the following, we compare the added mass calculations in hde against WAMIT and experimental results. A discussion of the result is presented in Section 5.5.

5.4.1 Added mass

In order to verify the results from the hde application, we compare the output with experimental data for the REMUS AUV (see Figure 5.8), reported in

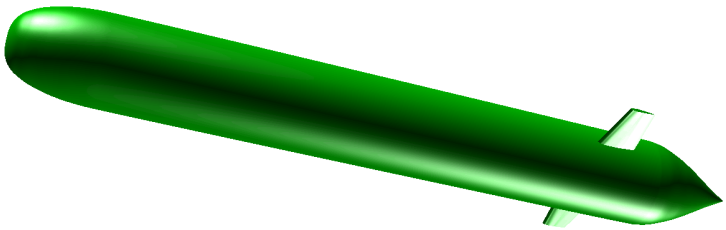


Figure 5.11: MultiSurf CAD model, MAYA

(Prestero 2001, Appendix C), and a torpedo shaped AUV designed and tested at the Queensland University of Technology (Ridley et al. 2003). The results are presented in Table 5.3 and 5.4.

	$X_{\dot{u}}$	$Y_{\dot{v}}$	$Y_{\dot{r}}$	$Z_{\dot{w}}$	$Z_{\dot{q}}$	$K_{\dot{p}}$	$M_{\dot{w}}$	$M_{\dot{q}}$	$N_{\dot{v}}$	$N_{\dot{r}}$
hde	−0.94	−35.31	2.14	−35.31	−2.14	−0.02	−2.14	−4.67	2.14	−4.67
Prestero (2001)	−0.93	−35.50	1.93	−35.50	−1.93	−0.07	−1.93	−4.88	1.93	−4.88
difference	0.6%	−0.5%	10.8%	−0.5%	10.9%	−76.9%	10.9%	−4.4%	10.8%	−4.4%

Table 5.3: REMUS added mass, hde estimates vs. experimental results

	$X_{\dot{u}}$	$Y_{\dot{v}}$	$Y_{\dot{r}}$	$Z_{\dot{w}}$	$Z_{\dot{q}}$	$K_{\dot{p}}$	$M_{\dot{w}}$	$M_{\dot{q}}$	$N_{\dot{v}}$	$N_{\dot{r}}$
hde	−0.41	−26.86	2.48	−26.86	−2.48	−0.04	−2.48	−4.23	2.48	−4.23
Ridley et al. (2003)	0.42	−27.20	−1.83	−27.20	1.83	−0.04	1.83	−4.34	−1.83	−4.34
Ridley et al. (2003)[2]	−0.42	−27.20	1.83	−27.20	−1.83	−0.04	−1.83	−4.34	1.83	−4.34
difference	−1.7%	−1.2%	35.4%	−1.2%	35.4%	6.0%	35.4%	−2.6%	35.4%	−2.6%

Table 5.4: Added mass, hde estimates vs. Ridley et al. (2003)

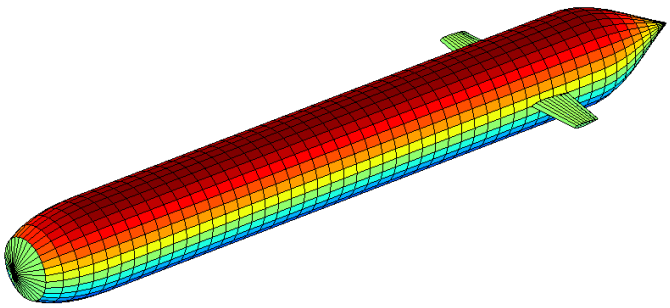


Figure 5.12: hde model used to obtain Table 5.5

In order to test the output from WAMIT, we also compare the output from

[2]To the best of the authors knowledge, it looks like there are some incorrect signs in the original text.

hde with WAMIT; the results are presented in Table 5.5.

	$X_{\dot{u}}$	$Y_{\dot{v}}$	$Z_{\dot{w}}$	$Z_{\dot{q}}$	$K_{\dot{p}}$	$M_{\dot{w}}$	$M_{\dot{q}}$	$N_{\dot{r}}$
hde	-1.00	-46.90	-53.87	-4.31	-0.06	-4.31	-10.28	-8.40
WAMIT	-2.14	-41.59	-43.86	-0.85	-0.02	-0.85	-6.85	-6.18
difference	-53.2%	12.8%	22.8%	405.0%	212.8%	404.9%	50.1%	36.0%

Table 5.5: MAYA added mass, hde estimates vs. WAMIT

5.4.2 Drag

So far, we have tested and verified hde's capability to estimate lift with open loop simulations; however there is one more property we need to address; namely drag. In the following, we try to compare the results from hde with experimental data reported in Ridley et al. (2003) and Prestero (2001). The results are provided in Table 5.6, and will be discussed in more detail at the end of the chapter.

	REMUS	Unnamed
hde	$F = 5.20$N	$X_{\lvert u\rvert u} = -2.49$kg/m
Prestero (2001)	$F = 4.01$N	
Ridley et al. (2003)		$X_{\lvert u\rvert u} = -3.11$kg/m

Table 5.6: Drag comparison, hde vs. experimental data

5.5 Discussion

In this chapter, we have presented the hde toolbox developed as a part of this work. With the aid of open loop simulations, we have compared the hde simulator against experimental data; and it has been seen that the results are indeed comparable.

The added mass estimates from hde are comparable to experimental results reported in Ridley et al. (2003) and Prestero (2001); at least if we "correct" the signs of some of the results reported in Ridley et al. (2003). However, it is seen that the estimates from hde of the cross-terms, $Y_{\dot{r}}$, $Z_{\dot{q}}$, $M_{\dot{w}}$ and $N_{\dot{v}}$, are overestimated compared to the experimental results. The added mass in roll, $K_{\dot{p}}$, is in very good agreement with the experimental results reported in Ridley et al. (2003), but is underestimated by almost 80% in Prestero (2001). One possible reason for this "huge" deviation is thought to be that the REMUS vehicle was equipped with some additional equipment during towing tank experiments (LBL-, SSS-, and ADCP transducers).

The comparison of added mass estimated by hde and WAMIT are a bit disappointing; at least if we take the relative good agreement between experimental

results and hde as a guideline. It seems like WAMIT fail to estimate the contribution from the low aspect-ratio foils involved in a typical AUV design. During the first test runs with WAMIT, the default panel size of 0.2 <units> was used, and we could hardly observe any contribution from the foils at all. However, in the results presented above and in Appendix C, the *panel_size* is reduced to 0.02 <units>; which is close to the minimum panel size WAMIT allow for this particular vehicle (WAMIT-Inc 2003, WAMIT can "only" subdivide a body into 2048 subdomains). An observation, made by the author, is that reduction of panel_size "improved" the output until we reached a panel size of approximately 0.08 <units>; further reduction did not yield any noticeable difference in the output.

Regarding the comparison of drag in Table 5.6, it is hard to give any qualitative evaluation of the results. We observe that hde overestimate the drag of the REMUS AUV and underestimate the drag compared to the result reported in Ridley et al. (2003). The latter is expected, because the vehicle used in experiment was equipped with a thruster/duct; which will add some drag to the total (recall that hde do not consider thruster issues at all). The former, on the other hand, is more difficult to explain. In the work of Prestero (2001), the objective was to separate the drag contribution from the AUV and the various equipment installed. It seems like Prestero (2001) has decided that the drag contribution from the hull equals empirical data provided in textbooks, in order to simplify further experiments where he estimate the drag contribution for equipment mounted on the body; therefor, one should probably be careful when using the data provided for drag as experimental results.

Chapter 6

Conclusion

In this work, we have successfully used the USAF Stability and Control Datcom (1978), to obtain hydrodynamic stability derivatives in the vertical plane. The equations has been rewritten so that they all refer to a common reference area (L_{pp}), which is standard for submerged vehicles (SNAME 1950). It is hoped that this work can be used as a foundation and reference, when modelling vehicles such as an AUV. It is believed that the work presented in Chapter 3, have the potential of reducing miscalculations and pitfalls due to conversation between all the reference areas normally used in aero- and hydrodynamic computation.

The theory presented in Chapter 3, has successfully been used in Chapter 4 to obtain a linearized model of the MAYA AUV in the vertical plane. In Chapter 5, the same theory was used to develop a MATLAB toolbox, hde, which is thought to be the main contribution from this work. The application, is capable of estimating hydrodynamic parameters such as static and dynamic derivatives (in both the vertical and horizontal plane), lift and moment curves for control surfaces, and added mass in six degrees of freedom. Further, a nonlinear MATLAB Simulink interface, which use the models constructed in hde, has been implemented. The simulator, described in Chapter 5.2, retains all the nonlinearities inherent in the coupled dynamics equations of motion, as well as those inherent to the hydrodynamic relations which govern the forces acting on the hull and control planes (Nahon 1996). It is hoped that hde, and its simulation environment, can be used by both students and professionals to analyze vehicles dynamics and/or for testing of controllers.

The output from hde has been compared with experimental data reported in Ridley et al. (2003) and Prestero (2001). Simulations in hde indicate that properties such as lift and turning rates are comparable with experimental data. Further, hde estimates of the added mass terms are indeed comparable with those reported from the authors mentioned above. The drag forces has also been investigated, but it has not succeeded to make a qualitative comparison of this particular term.

From WAMIT, an investigation of added mass of an typical AUV has been

carried out. Based on the results, it is concluded that WAMIT fail to accurately estimate the added mass contribution from the low aspect-ratio foils involved in a typical AUV design.

6.1 Recommendations for future work

A great amount of work has been carried out; but there are some improvements listed below that can be made to the hde environment.

In Chapter 3, we assumed that there was a turbulent boundary layer condition over the entire surface of the body. Even though this assumption may be adequate for a torpedo shaped vehicle operating at cruising speed, it indicates that hde will not favor a streamlined vehicle. It is therefore recommended that theory that distinguishes between the different vehicles is implemented. It would also be advantageous if we could extend the theory, by taking into account thrusters dynamic, and duct/body interaction.

The report generator as implemented could be extended to present the expected dynamic of the vehicle in form of figures showing the step response of the vehicle.

The notation used for parameters in the horizontal plane is not in accordance with standard notation in the current version, and should be improved.

The current version of hde, only supports vehicles with one set of control surfaces in each plane. If the theory is extended to take into account *downwash*, we could easily extend hde to estimate the expected performance for a vehicle with two control surfaces in one plane.

Further, the current version does not really estimate the body fixed damping matrix, in the form presented in (2.7). Even though it is thought that acceptable estimates of this particular matrix can be obtained, by performing standard maneuvering tests (Fossen 2002, Triantafyllou & Hover 2003) in the hde environment, and measure the response in Simulink, it would be favorable if we could extend the theory in hde to directly give an estimate of the matrix.

References

Abbot, I. H. & Doenhoff, A. E. (1959), *Theory of wing sections*, Dover books on physics and chemistry, McGraw-Hill Book Company.

Barros, E. A., Pascóal, A. & de Sa, E. (2004), AUV dynamics: Modelling and parameter estimation using analytical, semi-empirical, and CFD methods, *in* 'Proc. IFAC Conference on Control Applications in Marine Systems', Ancona, Italy. To appear.

Blakelock, J. H. (1991), *Automatic Control of Aircraft and Missiles*, 2 edn, John Wiley and Sons, Inc.

Blevins, R. D. (1979), *Formulas for natural frequency and mode shape*, Krieger Publishing.

Carmichael, B. H. (1966), 'Underwater drag reduction through optimum shape', *AIAA Second Propulsion Joint Specialist Conference* .

Datcom (1978), *USAF Stability and Control Datcom*, McDonnell Douglas Corporation, Wright-Patterson Airforce Base, Ohio. Hoak, D E & Finck, R D.

Fossen, T. I. (2002), *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*, Marine Cybernetics, Trondheim, Norway.

Fossen, T. I. (2003), 'Matlab gnc toolbox'. Internet. [Accessed September 27, 2003] <http://www.marinecybernetics.com/software.htm>.

Fossen, T. I. (2004*a*), 'Lecture notes'. Internet. [Accessed June 22, 2004] <http://www.itk.ntnu.no/fag/gnc/LectureNotes/Lecture.pdf>.

Fossen, T. I. (2004*b*), 'Marine cybernetics laboratory'. Internet. [Accessed June 18, 2004] <http://www.itk.ntnu.no/marinkyb/MCLab/>.

Gertler, M. (1950), Resistance experiments on a systematic series of streamlined bodies of revolution - for application to the design of high-speed submarines, Technical Report C-297, Navy Department- The David W. Taylor Model Basin.

Goheen, K. R. (1991), 'Modelling methods for underwater robotic vehicle dynamics', *Journal of Robotic Systems* **8**(3), 295–317.

Hoerner, S. F. (1985), *Fluid dynamic lift: Practical Information on Aerodynamic and Hydrodynamic lift*, 2 edn, Published by author.

Humphreys, D. E. (1981), Dynamics and hydrodynamics of ocean vehicles, *in* 'Proc. MTS/IEEE OCEANS', pp. 88–91.

International Submarine Engineering Ltd (2004), 'Ise web based designinfo'. Internet. [Accessed April 2, 2004] <http://www.ise.bc.ca/>.

Jones, G. W. (1952), Investigation of the effects of variation in the reynolds number between $0.4 \cdot 10^6$ and $3 \cdot 10^6$ on the low-speed aerodynamic characteristics of three low-aspect-ratio symmetrical wings with rectangular plan forms, Technical Report RM L52G18, NACA, Langley Aeronautical Laboratory. Internet. [Accessed June 16, 2004] <http://naca.larc.nasa.gov/digidoc/report/rm/18/NACA-RM-L52G18.PDF>.

Journée, J. M. J. & Massie, W. W. (2001), 'Offshore hydromechanics'. Internet. [Accessed June 16, 2004] <www.ocp.tudelft.nl/mt/journee/Files/Lectures/OffshoreHydromechanics.pdf>.

Maeda, H. & Tatsuta, S. (1989), Prediction method of hydrodynamic stability derivatives of an autonomous non-tethered submerged vehicle, Vol. 6, The Hague, Netherlands, pp. 105–114.

McCormick, B. W. (1994), *Aerodynamics, aeronautics, and flight mechanics*, 2 edn, Wiley Text Books.

McCormick, B. W. (1995), *Aerodynamics, Aeronautics, and Flight Mechanics*, 2 edn, John Wiley and Sons, Inc.

MikTex (2003), 'Tex implementation for the windows operating system'. Internet. [Accessed August 15, 2003] <http://www.miktex.org/>.

Munk, M. M. (1934), *Aerodynamic Theory*, Vol. 5, Dover Publ. Inc, chapter Aerodynamics of Airships. Reprinted several times.

Myring, D. F. (1976), 'A theoretical study of body drag in subcritical axisymmetric flow', *Aeronautical Quarterly* **27**(3), 186–194.

Nahon, M. (1993), Determination of Undersea Vehicle Hydrodynamic Derivatives Using the USAF Datcom, *in* 'Proc. MTS/IEEE OCEANS', Vol. 2, pp. 283–288.

Nahon, M. (1996), A simplified dynamics model for autonomous underwater vehicles, *in* 'Proc. Symposium on Autonomous Underwater Vehicle Technology', Vol. 2, IEEE, pp. 373–379.

Nelson, R. C. (1997), McGraw-Hill Higher Education.

Newman, J. N. (1999), *Marine Hydrodynamics*, 9 edn, MIT, Cambridge, Massachusetts.

Paster, D. L. (1986), Importance of hydrodynamic considerations for underwater vehicle design, *in* 'Proc. MTS/IEEE OCEANS', Vol. 18, pp. 1413–1422.

Pepijn, W. J., Johansen, T. A., Sørensen, A. J., Flanagan, C. & Toal, D. (2004), Neural network augmented identification of underwater vehicle models, *in* 'Proc. IFAC Conference on Control Applications in Marine Systems', Ancona, Italy. To appear.

Pereira, J. & Duncan, A. (2000), System identification of underwater vehicles, *in* 'Proceedings of the 2000 International Symposium on Underwater Technology', IEEE, pp. 419–424.

Pitts, W. C., Nielsen, J. N. & Kaattari, G. E. (1957), Lift and center of pressure of wing-body-tail combinations at subsonic, transonic, and supersonic speeds, Technical Report TN 1307, NACA. Internet. [Accessed March 15, 2004] <http://naca.larc.nasa.gov/reports/1957/naca-report-1307/naca-report-1307.pdf>.

Prestero, T. (2001), Verification of a six-degree of freedom simulation model for the remus autonomous underwater vehicle, Master's thesis, MIT.

Ridley, P., Fontan, J. & Corke, P. (2003), Submarine Dynamic Modeling, *in* 'Proc. Australasian Conference on Robotics & Automation', Brisbane, Australia.

Roskam, J. (1979), *Airplane flight dynamics and automatic flight controls*, number 1 *in* 'Airplane flight dynamics', Roskam Aviation and Engineering Corporation.

SNAME (1950), 'Nomenaclature for treating the motion of a submerged body through a fluid', Technical and Research Bulletin No. 1-5.

Triantafyllou, M. S. & Hover, F. S. (2003), 'Maneuvering and control of marine vehicles'. Internet. [Accessed November 20, 2003] <http://ocw.mit.edu/NR/rdonlyres/Ocean-Engineering/13-49Maneuvering-and-Control-of-Surface-and-Underwater-VehiclesFall2000/9902D412-8BF9-4401-874B-850F2FC6267A/0/all.pdf>.

WAMIT-Inc (2003), 'Wamit user manual'. Internet. [Accessed December 17, 2003] <www.wamit.com>.

Whicker, L. F. & Fehlner, L. F. (1958), Free-stream characteristics of a family of low-aspect-ratio, all movable control surfaces for application to ship design, Technical Report 933, Navy Department- The David W. Taylor Model Basin.

White, F. M. (1998), *Fluid Mechanics*, 4 edn, McGraw-Hill Book Company.

Appendices

Appendix A

Nomenclature

A.1 Notation

First Column	Second Column	Third Column
b	m	Foil span
$\bar{c}$	m	Mean hydrodynamic center; chord length
C_{D_0}	-	Parasite drag
CB	m	Center of buoyancy
CG	m	Center of gravity
C_{L_α}	rad^{-1}	Lift curve slope
C_{L_q}	rad^{-1}	Pitching moment, curve-slope
C_{L_δ}	rad^{-1}	Control surface, lift curve-slope
C_{m_α}	rad^{-1}	Pitching moment, curve-slope
C_{m_δ}	rad^{-1}	Control surface moment curve-slope
C_{m_q}	rad^{-1}	Pitching moment, curve-slope
$(C_{L_\alpha})_B$	rad^{-1}	Lift curve slope, body alone
$(C_{L_\alpha})_{F(B)}$	rad^{-1}	Lift curve slope, foil in presence of body
$(C_{L_\alpha})_{B(F)}$	rad^{-1}	Lift curve slope, body in presence of foil
$(C_{L_\alpha})_{FB}$	rad^{-1}	Lift curve slope, foil-body configuration
d	m	Diameter of body of revolution
d_b	m	Base diameter, body
f	-	Fineness ratio
$\boldsymbol{f}_b^n$	-	Buoyancy vector decomposed in NED
$\boldsymbol{f}_g^n$	-	Gravity vector decomposed in NED
$k_{B(F)}$	-	Correlation factor
$k_{F(B)}$	-	Correlation factor
$K_{F(B)}$	-	Correlation factor
$K_{B(F)}$	-	Correlation factor
ν	m^2/s	Kinematic viscousity
L_{pp}	m	Length between perpendiculars
x_c	m	Centroid of the volume (Center of buoyancy)
x_{hc}	m	Hydrodynamic center
$\forall$	m^3	Displaced volume, body

Continued on next page

First Column	Second Column	Third Column
R_n	-	Reynolds number
$\boldsymbol{R}_n^b(\cdot)$	-	Rotation matrix NED=>Body
$\boldsymbol{R}_b^n(\cdot)$	-	Rotation matrix Body=>NED
$\boldsymbol{S}(\cdot)$	-	Cross-product operator
S_{base}	m^2	Base area body
S_e	m^2	Exposed foil area
S_b	m^2	Wetted area body
S_{wf}	m^2	Wetted area foil
x_m	m	Reference point
x_c	m	Centroid volume (Center of buoyancy)
x_{hc}	m	Hydrodynamic center
ν	m^2/s	Kinematic viscosity
ς	-	d/b, diameter-span ratio
ϑ	-	b/d, span-diameter ratio
ϑ	-	b/d, span-diameter ratio

Table A.1: Nomenclature

A.2 Abbreviations

AUV	Autonom Underwater Vehicle
CAD	Computer Aided Drawing
CA	Center of Added mass
CB	Center of Buoyancy
CG	Center of Gravity
DOF	Degree Of Freedom
ECEF	Earth center earth fixed
NED	North East Down
hde	HydroDynamic Estimation toolbox

A.3 Matematical definitions and notation

Position and orientation vector, $\boldsymbol{\eta}$

The position and orientation vector is defined in ECEF. However, in this text flat earth navigation has been assumed, which suggest that the position vector can be decomposed in NED relative to some predefined reference point. Hence, the position vector is defined as:

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^n \\ \boldsymbol{\Theta} \end{bmatrix} \tag{A.1}$$

where $\boldsymbol{p}^n = [n,\, e,\, d]^{\mathrm{T}}$ is the relative position in north, east and down direction respectively. $\boldsymbol{\Theta} = [\phi,\, \theta,\, \psi]^{\mathrm{T}}$ is a vector of Euler angles.

Generalized velocity vector, $\boldsymbol{\nu}$

The generalized velocity vector utilized in this text, is decomposed in the body frame:

$$\boldsymbol{\nu} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^{\mathrm{T}} \tag{A.2}$$

where u, v, w are the body fixed linear velocities and p, q, r are the body fixed angular velocities respectively.

Definition of Euler rotation matrix

The Euler rotation matrix transform a vector from one coordinate system into another. It can be defined by the multiplication of three elementary rotation matrices corresponding to for example the vehicles roll ϕ, pitch θ and yaw ψ angles.

$$\begin{aligned} \boldsymbol{R}_b^n(\boldsymbol{\Theta}) &= \boldsymbol{R}_n^b(\boldsymbol{\Theta})^{-1} = \boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{z,\phi} \\ &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \\ &= \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \end{aligned} \tag{A.3}$$

The Euler rotation matrix is orthogonal and therefor its inverse is equal to its transpose.

Definition of the Cross product operator

The cross product operator is defined by $\boldsymbol{\lambda} \times \boldsymbol{a} := \boldsymbol{S}(\boldsymbol{\lambda})\boldsymbol{a}$, where

$$\boldsymbol{S}(\boldsymbol{\lambda}) = -\boldsymbol{S}^{\mathrm{T}}(\boldsymbol{\lambda}) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix}, \tag{A.4}$$

and $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3]^{\mathrm{T}}$.

6DOF transformation matrix, $\boldsymbol{J}(\boldsymbol{\eta})$

The 6DOF transformation matrix, transforms the body-fixed linear and angular velocities into earth fixed (NED) velocities ($\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu}$).

$$\boldsymbol{J}(\boldsymbol{\eta}) = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{\Theta}) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}(\boldsymbol{\eta}) \end{bmatrix} \tag{A.5}$$

where $\boldsymbol{T}(\boldsymbol{\eta})$ is the angular velocity transformation matrix:

$$\boldsymbol{T}(\boldsymbol{\eta}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \tag{A.6}$$

A.4 Derivation of body-pitching moment curve slope

$$
\begin{aligned}
C_{m_\alpha} &= \frac{2\cdot(k_2-k_1)}{L_{pp}^3}\cdot\int_0^{x_0}\frac{\mathrm{d}S(x)}{\mathrm{d}x}\,(x_m-x)\;\mathrm{d}x \quad (1/\mathrm{rad}),\\
&= \frac{2\cdot(k_2-k_1)}{L_{pp}^3}\cdot\left(x_m\int_{S_{(0)}}^{S_{(x_0)}}\mathrm{d}S(x)-\int_0^{x_0}x\,\mathrm{d}S(x)\right)\\
&= x_m\,Cla\,S(x)\,|_{x=0}^{x_0}-2\,(k_2-k_1)\underbrace{\left(x\,S(x)\,|_{x=0}^{x_0}-\int_0^{x_0}S(x)\,\mathrm{d}x\right)}_{\text{Chain rule}}\\
&= \frac{x_m\,C_{L_\alpha}}{L_{pp}}+2\,(k_2-k_1)\,\frac{V_0-S_0\,x_0}{L_{pp}^3} \quad [1/\mathrm{rad}]\,.
\end{aligned}
\tag{A.7}
$$

A.5 Open loop transfer functions

As you can imagine, it is hard to type the equations below. In order to avoid mistyping the author used MATLAB as a tool for "typing" the equations. The reader should be aware that if multiple operators are present, they are not mistyped (i.e "$--a=+a$", "$-+a=-a$"). All the parameters in the equation are as defined in Table 3.1.

$$\frac{\alpha}{\delta}(s) = \frac{\left(\frac{(I'_{yy}-M'_{\dot{q}})L_{pp}^2}{U^2}\cdot C_{L_{\dot{\delta}}}+C_{m_{\dot{\delta}}}\cdot\left(\frac{(Z'_{\dot{q}}+m'x'_G)L_{pp}^2}{U^2}\right)\right)\cdot s^2+\left(C_{m_{\dot{\delta}}}\cdot\left(\frac{(Z'_{\dot{q}}+m')L_{pp}}{U}\right)-\left(\frac{(Cmq-m'x'_G)L_{pp}}{U}\right)\cdot(C_{L_{\dot{\delta}}})\right)\cdot s-C_{L_{\dot{\delta}}}\cdot M'_\theta}{\left(\frac{(I'_{yy}-M'_{\dot{q}})L_{pp}^2}{U^2}\cdot\left(\frac{(m'-Z'_{\dot{w}})L_{pp}}{U}\right)-\left(\frac{(M'_{\dot{w}}+m'x'_G)L_{pp}}{U}\right)\cdot\left(\frac{(Z'_{\dot{q}}+m'x'_G)L_{pp}^2}{U^2}\right)\right)\cdot s^3+\left(C_{m_\alpha}\cdot\left(\frac{(Z'_{\dot{q}}+m'x'_G)L_{pp}^2}{U^2}\right)-Z'_\alpha\cdot\frac{(I'_{yy}-M'_{\dot{q}})L_{pp}^2}{U^2}-\left(\frac{(Cmq-m'x'_G)L_{pp}}{U}\right)\cdot\left(\frac{(m'-Z'_{\dot{w}})L_{pp}}{U}\right)-\left(\frac{(M'_{\dot{w}}+m'x'_G)L_{pp}}{U}\right)\cdot\left(\frac{(Z'_{\dot{q}}+m')L_{pp}}{U}\right)\right)\cdot s^2+\left(C_{m_\alpha}\cdot\left(\frac{(Z'_{\dot{q}}+m')L_{pp}}{U}\right)+\left(\frac{(Cmq-m'x'_G)L_{pp}}{U}\right)\cdot Z'_\alpha-M'_\theta\cdot\left(\frac{(m'-Z'_{\dot{w}})L_{pp}}{U}\right)\right)\cdot s+Z'_\alpha\cdot M'_\theta} \tag{A.8}$$

$$\frac{\theta}{\delta} = \frac{\left(\left(\frac{(m'-Z'_{\dot{w}})L_{pp}}{U}\right)\cdot C_{m_{\dot{\delta}}}+C_{L_{\dot{\delta}}}\cdot\left(\frac{(M'_{\dot{w}}+m'x'_G)L_{pp}}{U}\right)\right)\cdot s-C_{L_{\dot{\delta}}}\cdot C_{m_\alpha}-Z'_\alpha\cdot C_{m_{\dot{\delta}}}}{\frac{(I'_{yy}-M'_{\dot{q}})L_{pp}^2}{U^2}\cdot\left(\frac{(m'-Z'_{\dot{w}})L_{pp}}{U}\right)\cdot s^3-\left(\frac{(M'_{\dot{w}}+m'x'_G)L_{pp}}{U}\right)\cdot\left(\frac{(Z'_{\dot{q}}+m'x'_G)L_{pp}^2}{U^2}\right)\cdot s^3+\left(C_{m_\alpha}\cdot\left(\frac{(Z'_{\dot{q}}+m'x'_G)L_{pp}^2}{U^2}\right)-\frac{(I'_{yy}-M'_{\dot{q}})L_{pp}^2}{U^2}\cdot Z'_\alpha-\left(\frac{(Cmq-m'x'_G)L_{pp}}{U}\right)\cdot\left(\frac{(m'-Z'_{\dot{w}})L_{pp}}{U}\right)\cdot s^2-\left(\frac{(M'_{\dot{w}}+m'x'_G)L_{pp}}{U}\right)\cdot\left(\frac{(Z'_{\dot{q}}+m')L_{pp}}{U}\right)\right)\cdot s^2+\left(\left(\frac{(Cmq-m'x'_G)L_{pp}}{U}\right)\cdot Z'_\alpha-M'_\theta\cdot\left(\frac{(m'-Z'_{\dot{w}})L_{pp}}{U}\right)+C_{m_\alpha}\cdot\left(\frac{(Z'_{\dot{q}}+m')L_{pp}}{U}\right)\right)\cdot s+M'_\theta\cdot Z'_\alpha} \tag{A.9}$$

Appendix B

MATLABSource code

Below, are the different functions developed during this thesis. It is hoped that the source code is documented and structured in such a way that it is self-explanatory. The expressions and theory used are mainly those presented in Chapter 1, Chapter 2.3 and Chapter 5.

B.1 hde-toolbox

B.1.1 hdeSimulinkModel.m

```
function nuDot = hdeSimulinkModel(eta, nu, curr, Uc, varargin)
%
%   nudot = hdeSimulinkModel(eta, nu, curr, Uc, varargin) calculates the
%   acceleration of the hde model.
%
% Input:
%       - eta   6x1 position vector
%       - nu    6x1 velocity vector
%       - curr  3x1 disturbance vector (velocities decomposed in NED)
%       - Uc    (1+2*n)x1 Control input , where n is the number of control
%               planes.
%       - Use varargin if you want to open a new hde model within the
%         simulator environment
%
%
% Author:   Håvard Bø
% Date:     summer 2004
% Revisions:

%fid = fopen('temp.txt','w');
persistent FIRSTRUN HC MRB MA VISC LREF Lpp D WA dbDIVd
persistent DRAGNORM RHO CLA K W B XG XB CD
% Uc = [Propeller, axy_n, axy_p, axz_n, axz_p]
% References:
%   Fossen, T. I  (2003)  Matlab GNC Toolbox. [Internet]
%           <http://www.marinecybernetics.com/software.htm>
%   Fossen, T. I  (2002)  "Marine Control Systems: Guidance, Navigation and
%           Control of Ships, Rigs and Underwater Vehicles". Marine
%           Cybernetics, Trondheim Norway.
%   Jones, G. W (1952). "Investigation of the effects of variation in the
%           Reynolds number between 0.4*10^6 and 3*10^6 on the low-speed
%           aerodynamic characteristics of three low-aspect-ratio symmetrical
%           wings with rectangular plan forms". NACA, Langley Aeronautical Laboratory
%           Reportnr: RM L52G18. [Internet, June 16 - 2004]
%           <http://naca.larc.nasa.gov/digidoc/report/rm/18/NACA-RM-L52G18.PDF>
```

```
%   Nahon, M (1996). "A Simplified Dynamics Model for Autonomous Underwater
%           Vehicles", Proc. Symposium on Autonomous Underwater Vehicle
%           Technology VOL 2, pp 373-379. (IEEE).
%   Pitts, W C & Nielsen, J N & Kaattari, G E. (1957). "Lift and center of pressure of
%           wing-body-tail combinations at subsonic, transonic, and supersonic speeds.
%           NACA, technical report: TN 1307.  [Internet, March 15 - 2004
%           <http://naca.larc.nasa.gov/reports/1957/naca-report-1307/naca-r
%           eport-1307.pdf>

    if ~isempty(varargin)
        FIRSTRUN = [];
        [fileName, filePath] = uigetfile({'*.hde','HDE model'},'Load an HDE model from file');
        unzip([filePath,fileName],tempdir)
        fileNa = strrep(fileName,'.hde','');
        load([tempdir,[fileNa,'.mat']]);
        clear fileNa

    end
    % Executes first time the function is called
    if isempty(FIRSTRUN)
        g = 9.81;
        RHO = Body.Param.Rho;       % Fluid density
        MRB = Body.Mrb; % Rigid body system inertia matrix
        MA = Body.Ma;   % Added mass
        W = MRB(1,1)*g;   % Weight of the vehicle (used in restoring forces)
        B = RHO*g*Body.Volume; % Buoyancy of body (used in restoring forces)
        XG = [MRB(2,6)/MRB(1,1), 0 , MRB(5,1)/MRB(1,1)]'; % Fossen, T (2002, eq:3.55)
        XB = [Body.Data.Xm - Body.Xc, 0,0]'; %
        Lpp = Body.Param.Lpp;       % Length between perpendiculars
        LREF = Lpp;                 % Insert first reference length (used in drag calc)
        D = Body.Param.d;           % Diameter of body (reference)
        WA = Body.WA;               % Body, wetted Area
        VISC = Body.Data.KinViscousity;  % Kinematic viscousity
        dbDIVd = Body.Sim.dbDIVd; % Used to calculate basedrag
        DRAGNORM = Body.Sim.dragNormal; % Precalc data used to calc drag
        CD = Body.Sim.cdDIVcf; %Precalc data used to calc drag
        HC = Body.Sim.pos; % Hydrodynamic center of body alone
        CLA = 0.5*Body.derivatives.Cla*[1;1]; %Body lift curve slope
        K.Kfb = []; K.Kbf = []; K.kbf = []; K.kfb = []; % Init struct
        for k=1:length(Foil)
            B = B + RHO*g*Foil(k).Volume; % Displaced volume foil
            % Insert hydrodynamic center of the control surfaces
            HC = [HC,Foil(k).Sim.neg, Foil(k).Sim.pos];
            % Insert foilSection lift curve slopes
            % Calculations in HDE is based on both foils => divide by 2
            newCLA = 0.5*[Foil(k).Cla, Foil(k).Cla  ];

            if lower(Foil(k).Plane) == 'xy';
                CLA = [CLA, [newCLA;0, 0]]; % Only angle of attack affect...
            elseif lower(Foil(k).Plane) == 'xz';
                CLA = [CLA, [0, 0;newCLA]]; % Only sideslip angle affect...
            else
                error('Plane is not defined...');
            end
            clear newCLA
            % Insert characteristic lengths used in Rn calculations
            LREF = [LREF, Foil(k).Cmac, Foil(k).Cmac];
            CD = [CD, Foil(k).Sim.cdDIVcf/2, Foil(k).Sim.cdDIVcf/2];
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % Interference factors      %
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            K.Kfb = [K.Kfb, Body.FoilBody(k).Kfb, Body.FoilBody(k).Kbf];
            K.Kbf = [K.Kbf, Body.FoilBody(k).Kbf, Body.FoilBody(k).Kbf];
            K.kfb = [K.kfb, Body.FoilBody(k).kfb, Body.FoilBody(k).kfb];
            K.kbf = [K.kbf, Body.FoilBody(k).kbf, Body.FoilBody(k).kbf];
        end % For
```

```
%         warning('hdeSimulinkModel.m override the calculated Buoyancy (Bouyancy = Weight)')
          B = W;
    FIRSTRUN = 1;
    return
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   Extract Euler angles   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
phi = eta(4);                %
theta = eta(5);              %
psi = eta(6);                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1) Transform the disturbance vector into body frame
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nur = [(Rzyx(phi,theta,psi))^-1*curr; zeros(3,1)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2) Calculate relative velocity (6DOF)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Vr = nu+ nur;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3) Calculate the velocity and angle of attack /sideslip
%    at each reference point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [Vi, ALPHA, BETA] = refPointVel(Vr, HC) ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 4) Calculate lift at each reference point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lift due to angle of attack/ Sideslip (Step1)+
% control surface deflection (Step2)
[Step1, Step2, LiftIndDrag] = calcLift(CLA, ALPHA, BETA, Uc,K);
 CL =  ([1;1]*0.5*RHO*Lpp^2*(abs(Vi.^2))).*Step1;
 CLdelta = ([1;1]*0.5*RHO*Lpp^2*(abs(Vi.^2))).*Step2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 5) Calculate drag at each reference point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Cf = calcCf(VISC, LREF, Vi); % Schoenherr flat-plate coeff
% Dragcoeff corresponding to the each ref. point
Drag = 0.5*RHO*Lpp^2*(abs(Vi).*Vi).*...
        calcDrag(Cf,CD,dbDIVd,DRAGNORM, Uc,LiftIndDrag);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 6) Transform the forces and moments into body axes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fprop = Uc(1);  % Already body fixed (along the x axis)
FcFrame = [Drag;...
        CL(2,:) + CLdelta(2,:);...
        CL(1,:) + CLdelta(1,:)];
Fhydro = calcForceMoment(ALPHA, BETA, FcFrame, HC);
Fhydro(1) = Fhydro(1) - Fprop;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 7) Calculate the corresponding acceleration vector

% m2c(diag(diag(MA)), Vr)*Vr + m2c(diag(diag(MRB)), nu)*nu +
%m2c(MA, Vr)*Vr + m2c(MRB, nu)*nu
%Vrm2c(MA, Vr)*Vr +

nuDot = - (MRB+MA)^-1*(m2c(MRB, nu)*nu +m2c(MRB, Vr)*Vr...
```

```
      + gvect(W, B, theta, phi, XG, XB) + Fhydro);
% clc
% [ALPHA; BETA]*180/pi
%  [(m2c(MRB, nu)*nu)';
%  (m2c(MRB, Vr)*Vr)';
%  gvect(W, B, theta, phi, XG, XB)';
%  Fhydro';
% ]
%  pause(0.04)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%       Private functions used in calculations       %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Rotation matrix- Source Fossen, T I (2003)
function R = Rzyx(phi,theta,psi)
% R = Rzyx(phi,theta,psi) computes the Euler angle
% rotation matrix R in SO(3) using the zyx convention
%
% Author:   Thor I. Fossen
% Date:     14th June 2001
% Revisions:
    cphi = cos(phi);
    sphi = sin(phi);
    cth  = cos(theta);
    sth  = sin(theta);
    cpsi = cos(psi);
    spsi = sin(psi);
    R = [...
            cpsi*cth  -spsi*cphi+cpsi*sth*sphi  spsi*sphi+cpsi*cphi*sth
        spsi*cth  cpsi*cphi+sphi*sth*spsi   -cpsi*sphi+sth*spsi*cphi
        -sth      cth*sphi                  cth*cphi ];

% Coriolis matrix- Source Fossen, T I (2003)
function C = m2c(M,nu)
% C = M2C(M) computes the 6x6 Coriolis-centripetal matrix C from the
% the 6x6 system inertia matrix M=M'>0 and the 6x1 velocity vector nu
%
% Author:    Thor I. Fossen
% Date:      14th June 2001
% Revisions: 26th June 2002,  M21 = M12 is corrected to M12'
    nu1 = nu(1:3);
    nu2 = nu(4:6);
    M11 = M(1:3,1:3);
    M12 = M(1:3,4:6);
    M21 = M12';
    M22 = M(4:6,4:6);
    dt_dnu1 = M11*nu1 + M12*nu2;
    dt_dnu2 = M21*nu1 + M22*nu2;
    C = [  zeros(3,3)      -Smtrx(dt_dnu1)
        -Smtrx(dt_dnu1)  -Smtrx(dt_dnu2) ];


% Restoring forces - Source: Fossen, T (2003)
function g = gvect(W,B,theta,phi,r_g,r_b)
% g = GVECT(W,B,theta,phi,r_g,r_b) computes the 6x1 vector of restoring
% forces about an arbitrarily point O for a submerged body. For floating
% vessels, see Gmtrx.m
%
% Inputs:  W, B: weight and buoyancy
%          phi,theta: roll and pitch angles
%          r_g = [x_g y_g z_g]: location of CG with respect to O
%          r_b = [x_b y_b z_b]: location of CB with respect to O
%
% Author:   Thor I. Fossen
% Date:     14th June 2001
% Revisions:
```

```
sth  = sin(theta); cth  = cos(theta);
sphi = sin(phi);   cphi = cos(phi);

g = [...
   (W-B)*sth
  -(W-B)*cth*sphi
  -(W-B)*cth*cphi
  -(r_g(2)*W-r_b(2)*B)*cth*cphi + (r_g(3)*W-r_b(3)*B)*cth*sphi
   (r_g(3)*W-r_b(3)*B)*sth      + (r_g(1)*W-r_b(1)*B)*cth*cphi
  -(r_g(1)*W-r_b(1)*B)*cth*sphi + (r_g(2)*W-r_b(2)*B)*sth      ];

% Function used to calculate the velocity at a given reference point, and
% effective angle of attack/ sideslip    ref:  Nahon (1996)
function [Vi,ALPHAi,BETAi] = refPointVel(Vr, pos)
    xi = pos(1,:);
    yi = pos(2,:);
    zi = pos(3,:);
    u = Vr(1);
    v = Vr(2);
    w = Vr(3);
    p = Vr(4);
    q = Vr(5);
    r = Vr(6);
    Vi = sqrt(...   % Ref: Nahon (1996)
        (u + p*zi - r*yi).^2 +...
        (v + r*xi - p*zi).^2 +...
        (w + p*yi - q*xi).^2 );

    % Calculate effective angle of attack
    num = (w+p*yi-q*xi);
    den = (u+q*zi-r*yi);
    warning off MATLAB:divideByZero;
    %
    ALPHAi = atan( num./den );
    ALPHAi(find(isnan(ALPHAi))) = 0; % If not defined
    %
    % Calculate effective sideslip angle
    num = v+r*xi-p*zi;
    BETAi = asin( num./Vi );
    BETAi(find(isnan(BETAi))) = 0; % If zero velocity
    warning on MATLAB:divideByZero;

function CDbase = calcCDbase(CD_marked, dBase, dmax);
    CDbase = 0.029*((dBase/dmax)^3)/CD_marked;

% Function used to calculate Schoenherr flat-plate coeff
function [Cf,varargout] = calcCf(visc, L, U)
% The procedure below as  first designed was very timeconsuming.
% Later on, it has been modified by using a polynomal approximation of
% the Schoenherr mean line ().
    Rn = abs(U).*L/visc;
    Rn(find(Rn<300)) = 300;
    Rn(find(Rn>1e12)) = 1e12;
    varargout = Rn;
    %Rn = log(Rn);
    % A close approximation developed by the author
    %Cf = exp(- 0.00025759*Rn.^3 + 0.019483*Rn.^2 - 0.59059*Rn - 0.31927);
     Cf = (0.075)./(log10(Rn)-2).^2;
    % Old solution => Time consuming
%     if Rn< 1e4; Rn=1e4; end;
%     varargout = Rn;
%     Cf = [];
%     for i = 1:length(Rn)
%         Cf = [Cf, str2double(char(solve(['0.242/sqrt(Cf) = log10(',num2str(Rn(i)),'*Cf)'])))];
%     end
```

```
%

% Function used to calculate Schoenherr flat-plate coeff

function Drags = calcDrag(Cf,CD,dbDIVd,DRAGNORM, u, LiftIndDrag);
    RFB = 1.05; % According to Datcom (1978, Figure 4.3.3.1-37)
    temp = Cf.*CD; % Parasite drags

    temp(:,2:size(temp,2)) = temp(:,2:size(temp,2))*RFB +LiftIndDrag;
    temp(1) = (temp(1)*RFB + 0.029/sqrt(temp(1)) * (dbDIVd)^3)*...
        DRAGNORM; % from based on d^2 => Lpp^2
    Drags = temp;


function [Step1, Step2, LiftIndDrag] = calcLift(CLA, ALPHA, BETA, Uc, K);
    MAXcontrolInput = 30*pi/180;
    MAXangle = 25*pi/180; % According to Jones, G. W (1952)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate lift due to angle of attack/ Sideslip
% Step-1 in my Msc Report
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    angles = [ALPHA;BETA];
    body = angles(:,1); % Angle of attack/ Sideslip body
    angles = angles(:,[2:size(angles,2)]); % Extract Control surfaces
    indexStep1 = find(abs(angles) > MAXangle);
    angles(indexStep1) = sign(angles(indexStep1))*MAXangle;
    % Compensate for foil in presence of body; body in presece of foil
    % Pitts et.al (1957)
    angles = ([1;1]*(K.Kbf+K.Kbf)).*angles;
    temp = [body,angles];
     % Nondim lift in body frame for each section
    Step1 = temp.*CLA;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate lift due to deflection of control surfaces
% (Step-2 in my Msc Report)
% and add the lift induced drag...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    angles = [ALPHA;BETA];
    angles = angles(:,[2:size(angles,2)]); % Extract Control surfaces
    CLAcs =  CLA(:,[2:size(CLA,2)]); % Extract Control surfaces
    ControlInput = Uc(2:length(Uc));
    % Check for correct input%%%%%%%%%%%%%%%%%%%%%%%%
    temp = find(abs(ControlInput)>MAXcontrolInput);           %
    ControlInput(temp) = sign(ControlInput(temp))*MAXcontrolInput;
    if ~isempty(temp);
        warning(['Consider saturation on your control surfaces, DeltaCom > ',MAXcontrolInput]);
    end%
    clear temp;                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    indexStep2 = find(CLAcs); % Find indexes corresponding to Uc...
    % which are greater than MAXangle
    effAngles = 0*angles; % Initialize martix
for i=1:length(indexStep2)
    [ned, sid] = ind2sub(size(CLAcs), indexStep2(i));
     effAngles(ned,sid) = effAngles(ned,sid) + ControlInput(sid);
    if find(abs(angles(indexStep2(i))+ControlInput(sid)) > MAXangle);
        if find(indexStep1 == indexStep2(i))
            % Deflection do not produce additional lift,
            % which means that we can set the deflection to zero...
            ControlInput(sid) = 0;
        else
            % The effective lift has reached "saturation",
            % which means that we change the effective deflection
        ControlInput(sid) = sign(ControlInput(sid))*...
            (abs(ControlInput(sid)) - ((abs(angles(indexStep2(i))+ControlInput(sid))) - MAXangle));
        end % If
```

```
    end % If
end % For
    % Step-2 including "Foil body interference factors" (Pitts et.al, 1957)
    Step2 = ([1;1]*[(ControlInput').*(K.Kfb+K.Kbf+K.kfb+K.kbf)]).*CLAcs;
    Step2 = [zeros(size(Step2,1),1), Step2];

    LiftIndDrag = ([1;1]*[(ControlInput').*(K.kfb+K.kbf)] +...
        (angles).*([1;1]*(K.Kbf+K.Kfb))  ...
        ).*CLAcs;
    OEF = 0.87; % Oswald efficiency factor (Nahon,1996 & )
    LiftIndDrag = sum(LiftIndDrag,1);
    LiftIndDrag = 1/(pi*3.2*OEF)*LiftIndDrag.^2;



function Fb = calcForceMoment(alpha, beta, Fc, Xhc)
Ftemp = [];
Mtemp = [];
    for i = 1:size(alpha,2);
        Rya =[cos(alpha(i)) 0 sin(alpha(i));...
            0   1   0;...
         -sin(alpha(i))  0  cos(alpha(i))];
        Rzb = [cos(beta(i))  sin(beta(i))   0;...
           -sin(beta(i))   cos(beta(i))  0;...
           0    0   1];
        Fi = Rya'*Rzb'*Fc(:,i);
        Ftemp = [Ftemp, Fi ];
        Mtemp = [Mtemp, Smtrx(Xhc(:,i))*Fi];
    end
    Fb = sum([Ftemp;Mtemp],2);


function S = Smtrx(r)
% S = SMTRX(r) computes the 3x3 vector cross product matrix S=-S'
% such that rxt = S(r)t is true for all 3x1 vectors r and t
%
% Author:   Thor I. Fossen
% Date:     14th June 2001
% Revisions:

S = [    0  -r(3)   r(2)
      r(3)     0   -r(1)
     -r(2)   r(1)     0 ];
```

B.1.2 HDE.m

```
function varargout = hde(varargin)
% HDE - Toolbox for estimation of hydrodynamic parameters for a submerged body
% of revolution
%
% HDE('')       -   Open a dialogbox where the user can choose between the different
%                   modes
% HDE('load')   -   Open a dialog box where the user can specify a
%                   previously saved session
% HDE('torpedo') -  Start a new session where the shape of the vehicle is
%                   given by torpedo polynomials
% HDE('vertex') -   Start a new session where the shape of the vehicle is
%                   constructed by cubic hermite interpolation, between
%                   vertexes specified by the user
% HDE(filename.hde) -   Open the HDE model {filename.hde}
%
% Author:    Håvard Bø
% Date:      May 2003
% Revisions:
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
    gui_State = struct('gui_Name',   upper( mfilename), ...
                       'gui_Singleton',  gui_Singleton, ...
                       'gui_OpeningFcn', @hde_OpeningFcn, ...
                       'gui_OutputFcn',  @hde_OutputFcn, ...
                       'gui_LayoutFcn',  [] , ...
                       'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



function hde_OpeningFcn(hObject, eventdata, handles, varargin)
global FOIL BODY
if     isempty(varargin)
    % Display a dialogbox, where the user can choose to
    % load a model from file, or start a new one
    buttonHeight = 2.3;  buttonLength = 25; ButtonSpace = 0.2;
    fig_dim = [4*buttonHeight + 3*ButtonSpace, buttonLength*1.5];
    midFig = (fig_dim(2)/2)-buttonLength/2;

    % Define placement of buttons
    LoadButtonPos =     [midFig, 2.5*buttonHeight + 2*ButtonSpace  , buttonLength, buttonHeight];
    VertexButtonPos =   [midFig, 1.5*buttonHeight + ButtonSpace    , buttonLength, buttonHeight];
    TorpedoButtonPos =  [midFig, buttonHeight/2      , buttonLength, buttonHeight];

    temp = get(handles.figure1,'position');
    arr_fig = figure('unit','characters','NumberTitle','off','Menubar','none','resize','off',...
        'position', [temp(1) temp(2) fig_dim(2) fig_dim(1)]);
    clear temp

    Load_but = uicontrol('Style','pushbutton','unit','characters','String','Load model from file',...
        'position',LoadButtonPos, 'callback','uiresume','tag','Load');
    Vertex_but = uicontrol('Style','pushbutton','unit','characters','String','New vertex model',...
        'position',VertexButtonPos, 'callback','uiresume','tag','vertex');
    Torpedo_but = uicontrol('Style','pushbutton','unit','characters','String','New torpedo model',...
        'position',TorpedoButtonPos, 'callback','uiresume','tag','torpedo');
    set(arr_fig, 'Name', 'HDE, select mode')
    uiwait;
    but = gco;
    try
        cmd = lower(get(but,'tag'));
        close
        hde(cmd)
    catch
        disp('Ending hde...')
        close
    end

    % The user want a body where the shape is given by torpedo polynomials
elseif  strcmp(lower(char(varargin(1))), 'torpedo')
    % Set default input
    BODY.Profile = 'torpedo';
    BODY.Param.a = str2double(get(handles.a_Input, 'String'));
    BODY.Param.b = str2double(get(handles.b_Input, 'String'));
    BODY.Param.d = str2double(get(handles.D_Input, 'String'));
```

```
    BODY.Param.n = str2double(get(handles.n_Input, 'String'));
    BODY.Param.theta = str2double(get(handles.theta_Input, 'String'));
    BODY.Param.Lpp = str2double(get(handles.Lpp_Input, 'String'));
    BODY.Param.Lpoint = str2double(get(handles.Lpoint_Input, 'String'));
    BODY.Data.Xm = [];
    BODY.Data.AutoXm = 1;

    set(handles.Xm_Input,'Enable','off')
    set(handles.AutCalcXm,'value',1);
    set(handles.SepPoint_Input,'enable','off')

    set(handles.addVertex_PushButton,'Visible','off')
    set(handles.editVertex_PushButton,'Visible','off')
    set(handles.BG_Text,'visible','on')
    set(handles.BG_Input,'visible','on')
    set(handles.RigidBody_PushButton,'visible','off')
    set(handles.MrbAuto_Checkbox,'value',1)
    set(handles.SepPointAuto_Checkbox,'value',1)
    set(handles.SepPoint_Input,'enable','off')

    plotStartFig(hObject, eventdata, handles)

    % The user want to specify the body with the aid of vertexes
elseif strcmp(lower(char(varargin(1))), 'vertex')
    set(handles.Lpoint_Input,'Visible','off')
    set(handles.a_Input,'Visible','off')
    set(handles.b_Input,'Visible','off')
    set(handles.D_Input,'Visible','off')
    set(handles.theta_Input,'Visible','off')
    set(handles.n_Input,'Visible','off')
    set(handles.torpedoDescription(:), 'Visible', 'off')
    set(handles.torpedoDimensions(:), 'Visible', 'off')
    set(handles.BG_Text,'visible','on')
    set(handles.BG_Input,'visible','on')
    set(handles.RigidBody_PushButton,'visible','off')
    set(handles.MrbAuto_Checkbox,'value',1)
    set(handles.SepPointAuto_Checkbox,'value',1)
    set(handles.SepPoint_Input,'style','text')
    set(handles.Xm_Input,'Enable','off')
    set(handles.AutCalcXm,'value',1);
    set(handles.SepPoint_Input,'enable','off')

    BODY.Data.Xm = [];
    BODY.Data.AutoXm = 1;

    BODY.Profile = 'vertex';
    BODY.Param.Lpp = 1;
    BODY.Param.x = [0 BODY.Param.Lpp];
    BODY.Param.r = [0 0];
    set(handles.Lpp_Input, 'string', sprintf('%1.3f',BODY.Param.Lpp))
    plotVertex(hObject, eventdata, handles)

elseif strcmp(lower(char(varargin(1))), 'load')
    load_model_Callback(hObject, eventdata, handles)
else
    try  % load model, filename given by varargin(1)
        load_model_Callback(hObject, eventdata, handles, char(varargin(1)))
    catch % Function call is not valid
        errorMsg = sprintf('Function call %s is not reckognized as a valid input\n', char(varargin));
        errorMsg = sprintf('%sAvailable inputs are: \n',errorMsg);
        errorMsg = sprintf('%s- filename.hde\n',errorMsg);
        errorMsg = sprintf('%s- torpedo \n',errorMsg);
        errorMsg = sprintf('%s- vertex \n',errorMsg);
        errorMsg = sprintf('%s- load\n',errorMsg);
        error(sprintf('%s',errorMsg));
    end % try
```

```
end % if


% Choose default command line output for hde

function varargout = hde_OutputFcn(hObject, eventdata, handles)
    disp(sprintf('%s',eventdata));
%varargout{1} = handles.output;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% create functions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function SepPoint_Input_CreateFcn(hObject, eventdata, handles)
    global BODY
    if ispc; set(hObject,'BackgroundColor','white');
    else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
    end
    set(hObject,'value',1)
    BODY.Data.AutoSep = 1;
    BODY.Data.sepPoint = [];


function Lpp_Input_CreateFcn(hObject, eventdata, handles)
if ispc set(hObject,'BackgroundColor','white');
else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes during object creation, after setting all properties.
function velocityInput_CreateFcn(hObject, eventdata, handles)
global BODY
if ispc; set(hObject,'BackgroundColor','white');
else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

BODY.Data.velocity = str2double(get(hObject, 'string'));


function Lpoint_Input_CreateFcn(hObject, eventdata, handles)
if ispc set(hObject,'BackgroundColor','white');
else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during object creation, after setting all properties.
function a_Input_CreateFcn(hObject, eventdata, handles)
    if ispc; set(hObject,'BackgroundColor','white');
    else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function b_Input_CreateFcn(hObject, eventdata, handles)
    if ispc;    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function D_Input_CreateFcn(hObject, eventdata, handles)
    if ispc ; set(hObject,'BackgroundColor','white');
    else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
    end

function theta_Input_CreateFcn(hObject, eventdata, handles)

function n_Input_CreateFcn(hObject, eventdata, handles)
    if ispc; set(hObject,'BackgroundColor','white');
```

```
    else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
    end


function KinViscousity_Input_CreateFcn(hObject, eventdata, handles)
    global BODY
    if ispc; set(hObject,'BackgroundColor','white');
    else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
    end
    BODY.Data.KinViscousity = str2double(get(hObject, 'string'))*10^-6;

function AUVfigure_CreateFcn(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function FeedBack_Text_CreateFcn(hObject, eventdata, handles)
    outText(1:10,1:150) = ' ';
    set(hObject,'String', outText);

if ispc; set(hObject,'BackgroundColor','white');
else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function Xm_Input_CreateFcn(hObject, eventdata, handles)
    global BODY
    if ispc; set(hObject,'BackgroundColor','white');
    else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
    end



function BG_Input_CreateFcn(hObject, eventdata, handles)
global BODY
if ispc; set(hObject,'BackgroundColor','white');
else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
BODY.Data.BG = str2double(get(hObject, 'string'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%   End of CreateFcn
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%   Start of Callback functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function a_Input_Callback(hObject, eventdata, handles)
global BODY
    temp = str2double(get(hObject,'String'));% returns contents of a_Input as a double
    oldA = BODY.Param.a ;
    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg = 'Incorrect input, Ln is NaN';
        set(hObject, 'string', sprintf('%1.4f',oldA))
    elseif temp < 0
        errMsg = 'Incorrect input, Ln < 0';
        set(hObject, 'string', sprintf('%1.4f',oldA))
    elseif temp > BODY.Param.b
        errMsg = 'Incorrect input, Ln > Lt';
        set(hObject, 'string', sprintf('%1.4f',oldA))
    else
        BODY.Param.a = temp;
        set(hObject, 'string', sprintf('%1.4f',BODY.Param.a))
    end


function Lpp_Input_Callback(hObject, eventdata, handles)
    global BODY FOIL
    temp = str2double(get(hObject, 'String'));
```

```
    oldLpp = BODY.Param.Lpp;

    if isnan(temp)
        errMsg = 'Incorrect input, Lpp = NaN';
        set(hObject, 'String', sprintf('%1.4f',oldLpp));
    else
        switch lower(BODY.Profile)
            case 'torpedo'
                if temp > BODY.Param.Lpoint
                BODY.Param.Lpoint = temp;
                BODY.Param.Lpp = temp;
                set(handles.Lpoint_Input,'String',sprintf('%1.4f',BODY.Param.Lpoint));
                set(hObject,'String',sprintf('%1.4f',BODY.Param.Lpp));
                elseif temp < BODY.Param.b
                errMsg = 'Incorrect input, Lpp < Lt';
                set(hObject,'String',sprintf('%1.4f',BODY.Param.Lpp));
                else
                BODY.Param.Lpp = temp;
                set(hObject,'String',sprintf('%1.4f',BODY.Param.Lpp));
                end % if temp>Lpoint
            case 'vertex'
                if temp <= 0
                errMsg = 'Incorrect input, Lpp <= 0';
                set(hObject, 'String', sprintf('%1.4f',BODY.Param.Lpp));
                else
                BODY.Param.Lpp = temp;
                set(hObject, 'String', sprintf('%1.4f',BODY.Param.Lpp));
                BODY.Param.x = temp/oldLpp * BODY.Param.x; % Rescale
                BODY.Param.r = temp/oldLpp * BODY.Param.r; % Rescale
                errMsg = 'Rescaling body';
                BODY = bodyCalc(BODY);
                rescaleFoil(hObject, eventdata, handles, temp/oldLpp);
                plotShape(hObject, eventdata, handles);
                try plotFoil(hObject, eventdata, handles, handles.AUVfigure);
                catch
                errMsg = sprintf('%s\nDid not succeed to rescale foil section(s)',errMsg);
                end % try
                end % Check of input for vertex body
            otherwise
                errMsg = sprintf('%s\nDid not recognize body profile',errMsg);
        end % switch
    end %if

    try writeToFeedback(hObject, eventdata, handles, errMsg);end
% save changes in handle
    guidata(hObject, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%     End of Lpp_Input Callback
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function b_Input_Callback(hObject, eventdata, handles)
global BODY
    temp = str2double(get(hObject,'String'));% returns contents of b_Input as a double
    oldB = BODY.Param.b;
    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg = 'Incorrect input, Lt is NaN';
        set(hObject, 'string', sprintf('%1.4f',oldB))
    elseif temp < BODY.Param.a
        errMsg = 'Incorrect input, Lt < Ln';
        set(hObject, 'string', sprintf('%1.4f',oldB))
    elseif temp > BODY.Param.Lpp
        errMsg = 'Incorrect input, Lt > Lpp';
        set(hObject, 'string', sprintf('%1.4f',oldB))
    else
        BODY.Param.b = temp;
```

```
        set(hObject, 'string', sprintf('%1.4f',BODY.Param.b))
    end
try writeToFeedback(hObject, eventdata, handles, errMsg); end


function D_Input_Callback(hObject, eventdata, handles)
    global BODY
    temp = str2double(get(hObject,'string'));

    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg = 'Incorrect input, D is NaN';
        set(hObject, 'string', sprintf('%1.4f',BODY.Param.d));
    elseif temp <=0
        errMsg = 'Incorrect input, D must be a positive number';
        set(hObject, 'string', sprintf('%1.4f',BODY.Param.d))
    else
        BODY.Param.d = temp;
        set(hObject, 'string', sprintf('%1.4f',BODY.Param.d))
    end
    try
        writeToFeedback(hObject, eventdata, handles, errMsg)
    end


function theta_Input_Callback(hObject, eventdata, handles)
    global BODY
    temp = str2double(get(hObject,'string'));

    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg = 'Incorrect input, theta is NaN';
        set(hObject, 'string', sprintf('%2.2f',BODY.Param.theta));
    elseif temp <0 || temp > 50
        errMsg = 'Incorrect input, theta shpuld be in range (0,50)';
        set(hObject, 'string', sprintf('%2.2f',BODY.Param.theta))
    else
        BODY.Param.theta = temp;
        set(hObject, 'string', sprintf('%2.2f',BODY.Param.theta))
    end
    try
        writeToFeedback(hObject, eventdata, handles, errMsg)
    end


function n_Input_Callback(hObject, eventdata, handles)
global BODY
    temp = str2double(get(hObject,'String'));% returns contents of n_Input as a double
    oldN = BODY.Param.n;
    N_min = 0;
    N_max = 5;

    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg = 'Incorrect input, n is NaN';
        set(hObject, 'string', sprintf('%1.2f',BODY.Param.n))
    elseif temp < N_min || temp > N_max
        errMsg = sprintf('Incorrect input, should be in range %g<n<%g',N_min,N_max);
        set(hObject, 'string', sprintf('%1.2f',BODY.Param.n))
    else
        BODY.Param.n = temp;
        set(hObject, 'string', sprintf('%1.2f',BODY.Param.n))
    end
try writeToFeedback(hObject, eventdata, handles, errMsg); end



function Lpoint_Input_Callback(hObject, eventdata, handles)
global BODY
```

```
    oldL = BODY.Param.Lpoint;
    temp = str2double(get(hObject,'string'));

    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg = 'Incorrect input, L is NaN';
        set(hObject, 'string', oldL)
    elseif temp < 0
        errMsg = 'Incorrect input, L should be a positive number';
        set(hObject, 'string', sprintf('%1.4f',oldL))
    elseif temp < BODY.Param.a
        BODY.Param.a = temp;
        BODY.Param.b = temp;
        BODY.Param.Lpp = temp;
        BODY.Param.Lpoint = temp;
        set(handles.a_Input,'String',sprintf('%1.4f',BODY.Param.a));
        set(handles.b_Input,'String',sprintf('%1.4f',BODY.Param.b));
        set(handles.Lpp_Input,'String',sprintf('%1.4f',BODY.Param.Lpp));
        set(hObject,'String',sprintf('%1.4f',BODY.Param.Lpoint));
    elseif temp < BODY.Param.b
        BODY.Param.b = temp;
        BODY.Param.Lpp = temp;
        BODY.Param.Lpoint = temp;
        set(handles.b_Input,'String',sprintf('%1.4f',BODY.Param.b));
        set(handles.Lpp_Input,'String',sprintf('%1.4f',BODY.Param.Lpp));
        set(hObject,'String',sprintf('%1.4f',BODY.Param.Lpoint));
    elseif temp < BODY.Param.Lpp
        BODY.Param.Lpp = temp;
        BODY.Param.Lpoint = temp;
        set(handles.Lpp_Input,'String',sprintf('%1.4f',BODY.Param.Lpp));
        set(hObject,'String',sprintf('%1.4f',BODY.Param.Lpoint));
    else
        BODY.Param.Lpoint = temp;
        set(hObject,'String',sprintf('%1.4f',BODY.Param.Lpoint));
    end
try writeToFeedback(hObject, eventdata, handles, errMsg); end


function KinViscousity_Input_Callback(hObject, eventdata, handles)
    global BODY
    max = 2;
    min = 0.5;
    temp = str2double(get(hObject, 'string'));
    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg= 'Kinematic viscosity is NaN';
        set(hObject, 'String',sprintf('%1.4f',BODY.Data.KinViscousity))
    elseif temp<min || temp > max
        errMsg = ['Incorrect input, Kinematic viscosity should be in range (', min,',' max ')*10^-6'];
        set(hObject, 'String',sprintf('%1.4f',BODY.Data.KinViscousity))
    else
        BODY.Data.KinViscousity = temp*10^-6;
        set(hObject, 'String',sprintf('%1.2f',temp))
    end
try writeToFeedback(hObject, eventdata, handles, errMsg); end


function velocityInput_Callback(hObject, eventdata, handles)
global BODY
    temp = str2double(get(hObject, 'string'));
    if isnan(temp)
        temp = BODY.Data.velocity;
    elseif temp<=0
        errMsg('Velocity must be a positive number');
        temp = BODY.Data.velocity;
        writeToFeedback(hObject, eventdata, handles, errMsg)
    end
    BODY.Data.velocity = temp(1);
```

```
    set(hObject, 'string',sprintf('%1.3f',BODY.Data.velocity));


function FeedBack_Text_Callback(hObject, eventdata, handles)


% --- Executes on button press in AddWing_PushButton.
function AddWing_PushButton_Callback(hObject, eventdata, handles)
    global BODY FOIL

    if ~isfield(BODY, 'shape')
        textString = sprintf('Body must be generated before a foil can be added\n');
        textString = sprintf('%s=> Geometry is not specified\n',textString);
        return
    end
    H = FOILDIALOG;
    if ~validFoil_Input(hObject, eventdata, handles, H);
        textString = 'Incorrect input, foilsection not added';
        writeToFeedback(hObject, eventdata, handles, textString);
        return
    end

    if isempty(FOIL);
        FOIL = foilalone(foilCalc(H, BODY), 0, BODY);   % First foil-section?
        textString = sprintf('%s Foilsection added successfully...', H.Plane);
    else
        NewFoil = foilalone(foilCalc(H, BODY), 0, BODY);
        for i =1:length(FOIL)
            if strcmp(lower(FOIL(i).Plane),lower(H.Plane))
                FOIL(i) = NewFoil;
                textString = sprintf('%s Foilsection replaced successfully...', H.Plane);
                break
            elseif i == length(FOIL)
                FOIL(i+1) = NewFoil;
                textString = sprintf('%s Foilsection added successfully...', H.Plane);
            else
                textString = 'Something is wrong in AddWing_PushButton_Callback';
            end % if FOIL(i).Plane == H.Plane
        end % for
    end % if length(FOIL)==0;
    eval(sprintf('set(handles.%s_profileText,''string'',''%s'')',H.Plane,H.Profile));
    eval(sprintf('set(handles.%s_LeText,''string'',''%1.3f'')',H.Plane,H.Xo));
    eval(sprintf('set(handles.%s_SpanText,''string'',''%1.3f'')',H.Plane,H.Span));
    eval(sprintf('set(handles.%s_CrText,''string'',''%1.4f'')',H.Plane,H.Cr));
    eval(sprintf('set(handles.%s_CtText,''string'',''%1.4f'')',H.Plane,H.Ct));
    plotShape(hObject, eventdata, handles);
    plotFoil(hObject, eventdata, handles, handles.AUVfigure);
    writeToFeedback(hObject, eventdata, handles, textString);



% --- Executes on button press in RunApplication_PushButton.
function RunApplication_PushButton_Callback(hObject, eventdata, handles)
global BODY FOIL
BODY.Param.Rho = 1025; % Water density
if length(BODY)==0 || length(FOIL)==0
    errMsg = sprintf('Body and foil(s) must be present before running application\n');
    writeToFeedback(hObject, eventdata, handles, errMsg);
else
    allright = 1;

    try % To sort the control surface planes
        planes = [];
        for k = 1:length(FOIL)
            planes = [planes;FOIL(k).Plane];
```

```
    end
    planes = cellstr(char(planes));
    [planes, planeOrder] = sort(planes);
    FOIL = FOIL(planeOrder);
    clear planes planeOrder
catch
    allright = 0;
    errMsg = sprintf('Could not sort control planes\n%s',lasterr);
    writeToFeedback(hObject, eventdata, handles, errMsg);
end


Rho = BODY.Param.Rho;

try BODY = calcvolumSurface(bodyCalc(BODY));
    if get(handles.AutCalcXm,'value');
        BODY.Data.Xm = BODY.Xc;
        set(handles.Xm_Input,'string', sprintf('%1.4f',BODY.Data.Xm))
        %BODY.Data = calcvolumSurface(bodyCalc(BODY));
    end
    Xm = BODY.Data.Xm;
catch
    allright = 0;
    errMsg = sprintf('Error in calcvolumSurface\n%s',lasterr);
    writeToFeedback(hObject, eventdata, handles, errMsg);
end

try FOIL = foilcalc(FOIL,BODY);
catch
    allright = 0;
    errMsg = sprintf('Error in foilCalc\n%s',lasterr);
    writeToFeedback(hObject, eventdata, handles, errMsg);
end

if get(handles.MrbAuto_Checkbox,'value')    %if autoCalc Rigid body
    BG = str2double(get(handles.BG_Input,'string'));
    try
        BODY.Mrb = mrigidbody(BODY, FOIL, BG, Xm, Rho);
        BODY.Data.AutoMrb = 1;
    catch
        allright = 0;
        errMsg = sprintf('Error in mRigidBody\n%s',lasterr);
        writeToFeedback(hObject, eventdata, handles, errMsg);
    end
else BODY.Data.AutoMrb = 0;
end

try BODY.Ma = addedmass(BODY,FOIL, Xm, Rho);
catch
    allright = 0;
    errMsg = sprintf('Error in addedMass\n%s',lasterr);
    writeToFeedback(hObject, eventdata, handles, errMsg);
end

try
    BODY.nonDim.Mrb = massNonDim(BODY.Mrb, BODY.Param.Lpp, Rho); % NondimMass
    BODY.nonDim.Ma = massNonDim(BODY.Ma,  BODY.Param.Lpp, Rho); % NondimMass
catch
    allright = 0;
    errMsg = sprintf('Error in massNonDim\n%s',lasterr);
    writeToFeedback(hObject, eventdata, handles, errMsg);
end

try
    FOIL = foilalone(FOIL,Xm, BODY);
catch
```

```
        allright = 0;
        errMsg = sprintf('Error in foilalone.m \n%s',lasterr);
        writeToFeedback(hObject, eventdata, handles, errMsg);
    end

    try
        BODY = bodyalone(BODY,Xm);
    catch
        allright = 0;
        errMsg = sprintf('Error in bodyalone.m\n%s',lasterr);
        writeToFeedback(hObject, eventdata, handles, errMsg);
    end

    try
        BODY.CDoINFO = 'Drag, foil body combination (based on Lpp^2)';
        [BODY.FoilBody, BODY.CDo] = foilBodyComb(FOIL,BODY,Xm);
    catch
        allright = 0;
        errMsg = sprintf('Error in foilBodyComb.m\n%s',lasterr);
        writeToFeedback(hObject, eventdata, handles, errMsg);
    end

    try
        if BODY.Data.AutoSep
            SepPointAuto_Checkbox_Callback(hObject, eventdata, handles, 'Hei');
        end
    catch
        allright = 0;
        errMsg = sprintf('Simon says, something is wrong',lasterr);
        writeToFeedback(hObject, eventdata, handles, errMsg);
    end


%     try
%         BODY.D_Info = 'Damping matrix';
%         [BODY.D,BODY.nonDim.D]  = dampingTerms(BODY,FOIL);
%     catch
%         allright = 0;
%         errMsg = sprintf('Error in dampingTerms\n%s',lasterr);
%         writeToFeedback(hObject, eventdata, handles, errMsg);
%     end
%
%     try
%         BODY.T_Info = 'Rudder actuator configuration matrix';
%         [BODY.T, BODY.nonDim.T] = rudderTerms(BODY,FOIL);
%     catch
%         allright = 0;
%         errMsg = sprintf('Error in rudderTerms\n%s',lasterr);
%         writeToFeedback(hObject, eventdata, handles, errMsg);
%     end

    if allright
        BODY.Info = sprintf('Parameter estimation terminated successfully, %s',datestr(clock,0));
        textString = sprintf('%s',BODY.Info);
        textString = sprintf('%s.  Model written to:\n',textString);
        pathName = what;
        pathName = pathName.path ;
        fileName = 'output';
        fileName = [pathName filesep fileName];
        textString = sprintf('%s%s.hde',textString, fileName);
        plot3D(hObject, eventdata, handles)
        save_model_Callback(hObject, eventdata, handles,fileName);
        writeToFeedback(hObject, eventdata, handles, textString);
        if get(handles.Make_Report,'value')    %if autogenerate report
            makeFigures(hObject, eventdata, handles)
            makeReport(hObject, eventdata, handles);
```

```
        end
        writeToFeedback(hObject, eventdata, handles, '**********************');
    end % if allright
end %if foil

function CreateGeometry_PushButton_Callback(hObject, eventdata, handles)
    global  BODY FOIL
switch lower(BODY.Profile)

    case 'vertex'
        BODY = bodyCalc(BODY);
        plotShape(hObject, eventdata, handles)
        errMsg = 'Body geometry plotted';
        try
            plotFoil(hObject, eventdata, handles,handles.AUVfigure)
        catch
            errMsg = sprintf('%s\nNo foil present')
        end
        writeToFeedback(hObject, eventdata, handles, errMsg);

    case 'torpedo'
        n = str2double(get(handles.n_Input,'String'));
        a = str2double(get(handles.a_Input,'String'));
        b = str2double(get(handles.b_Input,'String'));
        Lpoint = str2double(get(handles.Lpoint_Input,'String'));
        Lpp = str2double(get(handles.Lpp_Input,'String'));
        theta = str2double(get(handles.theta_Input,'String'));
        d = str2double(get(handles.D_Input,'String'));

        writeToFeedback(hObject, eventdata, handles, 'Checking input')
        if max(isnan([n, a, b, Lpoint, Lpp theta, d]))
            errMsg = sprintf('Incorrect %s input, NaN',get(handles.shapeParamText,'String'));
        elseif n<=0.5 || n>4
            errMsg = (sprintf('Incorrect input, n=%d\nShould be in range 0.5<=n<4',n));
        elseif a<0||a>=b
            errMsg = (sprintf('Incorrect input, Ln=%d\nShould be a positive number  0<Ln<Lt',a));
        elseif b>=Lpp
            errMsg = (sprintf('Incorrect input, Lt=%d\nShould be a positive number  Ln<Lt<Lpp',b));
        elseif theta<=0||theta>50
            errMsg = (sprintf('Incorrect input, theta=%d\nShould be a positive number  0<theta<50',theta));
        elseif Lpp>Lpoint||Lpp<=0
            errMsg = (sprintf('Incorrect input, Lpp=%d\nShould be a positive number  Lt<Lpp<=L',Lpp));
        elseif d<0
            errMsg = (sprintf('Incorrect input, D=%d\nShould be a positive number',d));
        else
            writeToFeedback(hObject, eventdata, handles, 'Creating geometry...');

            try
                Body = BODY;    % Make copy of current state
                Body.Profile = 'torpedo';
                Body.Param.n = n;
                Body.Param.a = a;
                Body.Param.b = b;
                Body.Param.Lpoint = Lpoint;
                Body.Param.Lpp = Lpp;
                Body.Param.theta = theta;
                Body.Param.d = d;
                BODY = bodyCalc(Body);
                clear Body % clear temporary variable
                plotShape(hObject, eventdata, handles)
                errMsg = 'Body geometry plotted';
            catch
                errMsg = 'Something is wrong';
                errMsg = sprintf('%s\n%s',errMsg, lasterr);
            end % try
        end % if correct input, ( if max(isnan([...])
```

```
        try
            plotFoil(hObject, eventdata, handles, handles.AUVfigure)
        catch
            errMsg = sprintf('%s\nNo foil present')
        end
        writeToFeedback(hObject, eventdata, handles, errMsg);

    otherwise
        error('BODY.profile is not reckognized');
    end % switch / case

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% End of CreateGeometry_PushButton_Callback
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in Help_PushButton.
function Help_PushButton_Callback(hObject, eventdata, handles)
doc hde;


% --- Executes on button press in Make_Report.
function Make_Report_Callback(hObject, eventdata, handles)


% --- Executes on button press in addVertex_PushButton.
function addVertex_PushButton_Callback(hObject, eventdata, handles)
    global BODY

    axes(handles.AUVfigure);
    cAxis = axis;
    plotVertex(hObject, eventdata, handles, 'do not change axis'); % Update the figure

    % Loop, picking up the points.
    % set(textInfo,'visible','off')
    but = 1;
    while but == 1  % As long as the left mouse button is clicked
        [xi,yi,but] = ginput(1); % Read the point from the figure
        if yi<=0
            errorMsg = sprintf('Incorrect datapoint\n => vertex must have a positive radius');
        elseif(xi<=0 || xi>= BODY.Param.Lpp)
            errorMsg = sprintf('Incorrect datapoint\n => vertex Xpos must be in range (0,Lpp)')
        elseif but==1   % The user didn't rightclick
            x = [BODY.Param.x,xi];
            y = [BODY.Param.r,yi];
            xy = sortrows([x;y]')';
            BODY.Param.x = xy(1,:);
            BODY.Param.r = xy(2,:);
            plotVertex(hObject, eventdata, handles, 'do not change axis'); % Update the figure
        else
            errorMsg = sprintf('Ending %s mode',get(handles.addVertex_PushButton,'String'));
        end % if
        try
            writeToFeedback(hObject, eventdata, handles, errorMsg);
        catch
            % It is not necessary do do anything
        end % try
    end % while

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%   end of addVertex_PushButton_Callback
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in editVertex_PushButton.
function editVertex_PushButton_Callback(hObject, eventdata, handles)
    global BODY FOIL
    pos = get(gco, 'position') +  get(gcf, 'position');
```

```
    x = BODY.Param.x(2:length(BODY.Param.x)-1);
    r = BODY.Param.r(2:length(BODY.Param.r)-1);
    xy = editVertexes([x;r], pos);
    errorMsg = 'Empty matrix returned...';
    if ~isempty(xy)
        if max(max(isnan(xy))); errorMsg('Incorrect Input, NaN');
        else
            xy = sortrows(xy')';    % Sort, ascending
            keepVertex = find(sum(xy,1)~=0);  %  elements to keep
            xy = xy(:,keepVertex); % (0,0) elements are removed

            xNew = [BODY.Param.x(1), xy(1,:), BODY.Param.x(length(BODY.Param.x))];
            yNew = [BODY.Param.r(1), xy(2,:), BODY.Param.r(length(BODY.Param.r))];
            if min(xNew)<0 || max(xNew)> BODY.Param.Lpp;
                errorMsg = 'Incorrect input, vertexes must be in range 0<x<Lpp';
            elseif min(xy(2,:)) <= 0
                errorMsg = 'Incorrect input, vertexes must be in range 0 < y';
            else
                try
                    interp1(xNew,yNew,0);
                    BODY.Param.x = xNew;
                    BODY.Param.r = yNew;
                    errorMsg = 'Vertexes updated successfully';
                    errorMsg = sprintf('%s\nRecalculating body shape',errorMsg);
                    BODY = bodycalc(BODY);
                    plotShape(hObject, eventdata, handles);
                    try
                        if ~isempty(FOIL)
                            FOIL = foilcalc(FOIL,BODY);
                            plotFoil(hObject, eventdata, handles, handles.AUVfigure);
                            errorMsg = sprintf('%s\n=>Foil(s) reconfigured',errorMsg);
                        end
                    catch
                        errorMsg = sprintf('%s\n=> Could not reconfigure foilsection(s)',errorMsg);
                    end
                catch
                    errorMsg = 'Incorrect input, data abscissae should be distinct';
                end % try
            end   % if max(max(isnan(...)))
        end %if
    else
        errorMsg = 'No changes executed';
    end % ~isempty
    writeToFeedback(hObject, eventdata, handles, errorMsg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%   end of editVertex_PushButton_Callback
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function Xm_Input_Callback(hObject, eventdata, handles)
    global BODY
    temp = str2double(get(hObject, 'string'));
    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg= 'Xm is NaN';
        set(hObject, 'String',sprintf('%1.4f',BODY.Data.Xm))
    else
        BODY.Data.Xm = temp;
        set(hObject, 'String',sprintf('%1.4f',BODY.Data.Xm))
    end
    try writeToFeedback(hObject, eventdata, handles,errMsg); end

% --- Executes on button press in AutCalcXm.
function AutCalcXm_Callback(hObject, eventdata, handles)
global BODY
    if get(hObject,'value') % Want to calculate rigid body automatically
```

```
        BODY.Data.AutoXm = 1;
        set(handles.Xm_Input,'Enable','off');
        set(handles.Xm_Input,'String','N/A')
        BODY.Data.Xm = [];
    else
        BODY.Data.AutoXm = 0;
        set(handles.Xm_Input,'Enable','on');
        BODY.Data.Xm = BODY.Param.Lpp/2;
        set(handles.Xm_Input,'String',sprintf('%1.4f',BODY.Data.Xm));
    end


function BG_Input_Callback(hObject, eventdata, handles)
    global BODY
    temp = str2double(get(hObject, 'string'));
    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg = 'BG is NaN';
        set(hObject, 'String',sprintf('%1.4f',BODY.Data.BG))
    else
        if temp<0; errMsg = 'Warning, BG<0 (metacentric unstable)';end
        BODY.Data.BG = temp;
        set(hObject, 'String',sprintf('%1.4f',BODY.Data.BG))
    end
    try writeToFeedback(hObject, eventdata, handles, errMsg); end



% --- Executes on button press in MrbAuto_Checkbox.
function MrbAuto_Checkbox_Callback(hObject, eventdata, handles)
global BODY
    if get(hObject,'value') % Want to calculate rigid body automatically
        set(handles.RigidBody_PushButton,'visible','off')
        set(handles.BG_Input,'visible','on')
        set(handles.BG_Text,'visible','on')
    else
        set(handles.RigidBody_PushButton,'visible','on')
        set(handles.BG_Input,'visible','off')
        set(handles.BG_Text,'visible','off')
        BODY.Ma = zeros(6,6); % Default value
    end


% --- Executes on button press in RigidBody_PushButton.
function RigidBody_PushButton_Callback(hObject, eventdata, handles)
global BODY
 pos = get(gco, 'position') +  get(gcf, 'position');
 pos = pos(1:2);
    if isfield(BODY,'Mrb')
        if min(size(BODY.Mrb) == [6,6])
            temp = editRigidBody(BODY.Mrb, pos );
        else
            temp = editRigidBody(zeros(6,6), pos );
        end % if
    else
        temp = editRigidBody(zeros(6,6), pos );
    end % if isfield()

    if max(max(isnan(temp))) % if incorrect input format
        errMsg = 'Incorrect input format (NaN)'
        writeToFeedback(hObject, eventdata, handles, errMsg);
    else
        BODY.Mrb = temp;
    end


% --- Executes on button press in SepPointAuto_Checkbox.
```

```
function SepPointAuto_Checkbox_Callback(hObject, eventdata, handles, varargin)
    global BODY
    if get(hObject,'value')  || ~isempty(varargin)
        set(handles.SepPoint_Input,'Enable','off');
        BODY.Data.AutoSep = 1;
        BODY.Data.sepPoint = [];
        switch lower(BODY.Profile)
            case 'torpedo'
                sepPoint = BODY.Param.a; % Seperation point of torpedo
            case 'vertex'
                % The separation point is on the most negative curve slope
                % for a stremlined body of revolution
                n = 10; % Avrage the input in order to remove "noise"
                Rtemp = [];
                Xtemp = [];
                for i =1:length(Ri)/n
                    if i*n<=length(x);temp = i*n ;else temp = length(x); end
                    Rtemp = [Rtemp; mean(Ri(1+(i-1)*n: temp))];
                    Xtemp = [Xtemp; mean(Xi(1+(i-1)*n: temp))];
                end
                [aTemp, bTemp] = min(diff(Rtemp));
                sepPoint = Xtemp(bTemp);    % Most negative slope
            otherwise
                error('Something is wrong');
        end
        BODY.Data.sepPoint = sepPoint;
        set(handles.SepPoint_Input,'string',sprintf('%1.4f',BODY.Data.sepPoint));

    else
        set(handles.SepPoint_Input,'Enable','on');
        BODY.Data.AutoSep = 0;
    end


function SepPoint_Input_Callback(hObject, eventdata, handles)
    global BODY
    temp = str2double(get(hObject,'string'))
    if isnan(temp) || ~min(size(temp) == [1 1])
        errMsg = 'Incorrect dataformat, Xsep = NaN';
        BODY.Data.sepPoint = BODY.Param.Lpp/2;
        set(hObject,'string',sprintf('%1.4f',BODY.Data.sepPoint));
    elseif temp<=0 || temp > BODY.Param.Lpp
        errMsg = 'Incorrect input, sould be in range (0,Lpp)';
        BODY.Data.sepPoint = BODY.Param.Lpp/2;
        set(hObject,'string',sprintf('%1.4f',BODY.Data.sepPoint))
    else
        BODY.Data.sepPoint = temp;
        set(hObject,'string',sprintf('%1.4f',BODY.Data.sepPoint));
    end
try writeToFeedback(hObject, eventdata, handles, errMsg); end


% --- Executes on button press in Exit_PushButton.
function Exit_PushButton_Callback(hObject, eventdata, handles)
    clear global BODY   % Remove variable from workspace
    clear global FOIL
    close

function load_model_Callback(hObject, eventdata, handles,varargin)
        if isempty(varargin)
            [fileName, filePath] = uigetfile({'*.hde','HDE model'},'Load an HDE model from file');
        else
            filePath = '';
            fileName = char(varargin(1));
        end
        eval(sprintf('unzip(''%s%s'')',filePath,fileName));
```

```
        fileNa = strrep(fileName,'.hde','');
        eval(sprintf('load %s',fileNa));
        delete(handles.figure1)
        handles = guidata(eval(sprintf('hgload(''%s'', ''all'')',fileNa)));
        eval(sprintf('delete %s.mat %s.fig', fileNa, fileNa));
        clear fileNa
        global BODY FOIL
        BODY = Body;    % Variable loaded from previous session
        FOIL = Foil;    % Variable loaded from previous session
        clear Body Foil



function save_model_Callback(hObject, eventdata, handles,varargin)
    global BODY FOIL
    if isempty(varargin)
        [file,DirPath] = uiputfile({'*.hde','HDE model'},'Save HDE model as...');
        fileName = sprintf('%s%s',DirPath,file);
    else
        fileName = char(varargin(1));
    end
    tempHandles = handles;

    fileName = strrep(fileName,'.hde','');
    fileExt = 'hde';
    Body = BODY;
    Foil = FOIL;
    eval(sprintf('hgsave(''%s.fig'', ''all'')',fileName));
    eval(sprintf('save %s.mat Foil Body',fileName));
    eval(sprintf('zip(''%s.%s'',{''%s.fig'',''%s.mat''})',fileName,fileExt,fileName,fileName));
    %eval(sprintf('delete %s.mat %s.fig',fileName,fileName));
    clear Foil Body tempHandles

%    writeToFeedback(hObject, eventdata, handles, textString);



function plotVertex(hObject, eventdata, handles, varargin)
    global BODY;
    axes(handles.AUVfigure);
    oldAxis = axis;
    plot(BODY.Param.x, BODY.Param.r,'b+');
    if isempty(varargin)
        axis equal;
        a = axis;
        axis([-0.05*a(2),1.05*a(2),1.05*a(3),1.05*a(4)]);
    else
        axis(oldAxis);
        text('String', 'Left click to add a vertex, right click to end this mode',...
            'position', [(oldAxis(2)-oldAxis(1))*0.02, 0.45*(oldAxis(4)- oldAxis(3)), 0],...
            'FontWeight', 'demi')
    end
    set(handles.AUVfigure, 'Visible','on')

function plotShape(hObject, eventdata, handles)
    global BODY;
    axes(handles.AUVfigure);
    plot(BODY.shape.x, BODY.shape.r,'b',BODY.shape.x, -BODY.shape.r,'b');
    h = gcf;
    axis equal;
    a = axis;
    axis([-0.05*a(2),1.05*a(2),1.05*a(3),1.05*a(4)]);
    set(handles.AUVfigure, 'Visible','on')

% Function called in order to load a figure describing the parameters of a
% torpdo shaped vehicle
```

```
function makeFigures(hObject, eventdata, handles)
global BODY FOIL
    % Find the correct path for the figure to be saved
    filePath = mfilename('fullpath');
    filePath = filePath(1:length(filePath)-length(mfilename));
    filePath = [filePath 'latex' filesep 'gfx' filesep];

%Make a figure of the body shape
    h = figure('visible','off');
    plot(BODY.shape.x, BODY.shape.r,'b',BODY.shape.x, -BODY.shape.r,'b');
    if ~strcmp(lower(BODY.Profile),'torpedo')
        hold on
        plot(BODY.Param.x,BODY.Param.r,'xr');
        hold off
    end
    axis equal;
    a = axis;
    axis([-0.05*a(2),1.05*a(2),1.05*a(3),1.05*a(4)]);
    print(h,'-dpng',[filePath 'bodySideView']);
    delete(h)

% Make 3d figure used on the frontpage of the report
    NTHETA = 30;
    NDISC = 50;
    theta = 0:2*pi/NTHETA:2*pi;

    X = 0:BODY.Param.Lpp/NDISC:BODY.Param.Lpp;
    R = interp1(BODY.shape.x,BODY.shape.r, X);
    X = ones(NTHETA+1,1)*X;

    h = figure('visible','off');
    surf(X, cos(theta(:))*R, sin(theta(:))*R);
    hold on
    plotFoil(hObject, eventdata, handles, gca, 'plotsurface');
    hold off
    axis equal
    set(get(h,'Children'),'visible','off')
    print(h,'-dpng',[filePath 'body3d']);
    delete(h)

function plotStartFig(hObject, eventdata, handles)
axis(handles.AUVfigure);
plotOpenfig(handles.AUVfigure)

function plot3D(hObject, eventdata, handles)
    global BODY
    NTHETA = 30;
    NDISC = 50;
    theta = 0:2*pi/NTHETA:2*pi;

    X = 0:BODY.Param.Lpp/NDISC:BODY.Param.Lpp;
    R = interp1(BODY.shape.x,BODY.shape.r, X);
    X = ones(NTHETA+1,1)*X;

    axes(handles.AUVfigure); % get the correct axes
    surf(X, cos(theta(:))*R, sin(theta(:))*R)
    plotFoil(hObject, eventdata, handles, handles.AUVfigure, 'plotsurface')
    axis equal
    set(handles.AUVfigure, 'Visible','off')


function plotFoil(hObject, eventdata, handles, AKSER,  varargin)
    global FOIL;
     axes(AKSER);
     hold on
```

```
     for i=1 : length(FOIL)
         tInc = ceil(length(FOIL(i).Curve.Discrete.root.x)/10);
         tLength = length(FOIL(i).Curve.Discrete.root.x);
         x = [FOIL(i).Curve.Discrete.root.x(1:tInc:tLength);...
                 FOIL(i).Curve.Discrete.tip.x(1:tInc:tLength)];
         yu = [FOIL(i).Curve.Discrete.root.y(1,(1:tInc:tLength));...
                 FOIL(i).Curve.Discrete.tip.y(1,(1:tInc:tLength))];
         yl = [FOIL(i).Curve.Discrete.root.y(2,(1:tInc:tLength));...
             FOIL(i).Curve.Discrete.tip.y(2,(1:tInc:tLength))];
         z = [FOIL(i).Curve.Discrete.root.z(1:tInc:tLength);...
                 FOIL(i).Curve.Discrete.tip.z(1:tInc:tLength)];
         if strcmp(lower(FOIL(i).Plane), 'xy')
             if isempty(varargin)
                 plot3(x,yu,z,'b',x,yl,z,'b')
             else
                 surf(x,z,yu); surf(x,z,yl); % Positive y axis foil
                 surf(x,-z,yu); surf(x,-z,yl); % Negative y axis foil
             end
         elseif strcmp(lower(FOIL(i).Plane), 'xz')
             if isempty(varargin)
                 plot3(x,z,yu,'b',x,-z,yl,'b')
             else
                 surf(x,yu,z); surf(x,yl,z); % Upper foil
                 surf(x,yu,-z); surf(x,yl,-z); % Lower foil
             end
         end
     end
     hold off



function rescaleFoil(hObject, eventdata, handles, factor)
    global FOIL
    for i = 1:length(FOIL)
        try
            Xo = FOIL(i).Xo;
            Xo_new = FOIL(i).Xo*factor;
            Xi_root = FOIL(i).Curve.Discrete.root.x;
            Xi_tip = FOIL(i).Curve.Discrete.tip.x;
            Xi_rootN = Xi_root-Xo + Xo_new;
            Xi_tipN = Xi_tip - Xo + Xo_new;
            FOIL(i).Xo = Xo_new;
            FOIL(i).Curve.Discrete.root.x = Xi_rootN;
            FOIL(i).Curve.Discrete.tip.x = Xi_tipN;
        catch
            error('Something wrong in rescaleFoil function')%
        end % try
    end % for




function writeToFeedback(hObject, eventdata, handles, NewTextString)
    oldString = get(handles.FeedBack_Text,'String');
    NewTextString = sprintf('*%s',NewTextString);
    set(handles.FeedBack_Text,'String',padText(oldString, NewTextString));
    pause(0.2) % Force repaint of the feedback panel

function outText = padText(oldString, NewTextString)
    if length(NewTextString) > length(oldString);
        NewTextString = NewTextString(1:length(oldString));
        temp = [NewTextString; oldString];
        outText = temp(1:size(oldString,1),:);
    else
        temp(1:size(NewTextString,1), 1:length(oldString)) = ' ';
        temp(1:size(NewTextString,1),1:length(NewTextString)) = NewTextString;
```

```
        temp = [temp; oldString];
        outText = temp(1:size(oldString,1),:);
    end


% Check if the format of the foil input is correct
function output = validFoil_Input(hObject, eventdata, handles, Foil)
    global BODY;
    temp = 0; % Assume incorrect input
    try
        Dbody = interp1(BODY.shape.x,BODY.shape.r,Foil.Xo+Foil.Cr/2);
        if Foil.Cr+Foil.Xo > BODY.Param.Lpp
            errMsg = 'Incorrect input, foil is mounted behind body';
            output = 0;
        elseif (Foil.Cr || Foil.Ct) < 0
            errMsg = 'Incorrect input, chord lengths must be a positive number';
            output = 0;
        elseif Foil.Span < Dbody
            errMsg = 'Incorrect input, foil span < diameter of body';
            output = 0;
        else
            output = 1; % Correct input
        end % if
    catch   % There is something wrong with the foil format
        errMsg = 'Foil not added';
        output = 0;
    end % try
    try writeToFeedback(hObject, eventdata, handles, errMsg); end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%   end of validFoil_Input
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function makeReport(hObject, eventdata, handles)
    global BODY FOIL
    oldPath = what;
    oldPath = oldPath.path; % Currently working directory
    LatexPath = mfilename('fullpath');
    LatexPath = LatexPath(1:length(LatexPath)-length(mfilename));
    LatexPath = [LatexPath 'latex'];
    cd(LatexPath);
    fid = fopen('report.pdf','w');
    if fid>0
        fclose(fid);
        createReport(BODY,FOIL);
        [status, er] = dos('pdflatex report');
        if status
            cd(oldPath);
            errMsg = 'Latex must be installed and added to the path, report not generated';
            writeToFeedback(hObject, eventdata, handles, errMsg);
        else
            cd(oldPath);
            errMsg = 'Generating report...';
            writeToFeedback(hObject, eventdata, handles, errMsg);
            cd(LatexPath);
            [status, er] = dos('bibtex report');
            if status
                errMsg = er;
                writeToFeedback(hObject, eventdata, handles, errMsg);
            end
            [status, er] = dos(['openReport.bat' ]);
            if status
                errMsg = er;
                writeToFeedback(hObject, eventdata, handles, errMsg);
            end
```

```
            cd(oldPath);
        end
    else
        cd(oldPath);
        errMsg = ('Could not write report, document already open?');
        writeToFeedback(hObject, eventdata, handles, errMsg);
    end
```

B.1.3 addedMass.m

```
function Ma = addedMass(body, foil, xm, rho)
% function addedMass
%
% Output:
%       Ma = addedMass(body, foil, xm, rho)
%       where Ma is a 6x6 matrix with added mass coefficients according to
%       SNAME(1950)
%
% Input:
%       body - structure of format given by bodycalc.m
%       foil -  a 1xN structure of N foilsections. Each structure should
%               include the following terms:
%           .Cr -   Chord length [m]
%           .Xo -   Distance from nose to leading edge of the root chord
%           .Plane  -   Describe the plane of the foil pair ('XY' or 'XZ')
%           .Span   -   foil span (tip to tip) of the foil pair
%       xm - Reference point for calculations (assuming y = z = 0)
%       rho -   water density (kg/m^3)
%
%
% References:
% SNAME(1950) - The Society of Naval Architects and Marine Engineers, "Nomenaclature
%               for Treating the Motion of a Submerged Body Through a
%               Fluid", Technical and Research Bulletin No. 1-5, April 1950
%
% Fossen(2002)- Fossen, Thor I., "Marine Control Systems",
%               ISBN 82-92356-00-2, Marine Cyberntetics, Trondheim- Norway
%
% Newman(1999)- Newman, J N, "Marine Hydrodynamics", ISBN 0-262-14026-8
%
% Blevins(1979)- Blevins, Robert D, "Formulas for natural frequency and mode shape", ISBN 0-89464-894-2


% Calculates the added mass for a body of revolution
% All expressions are taken from table 4.3 in
% Newman(1999)
x = body.shape.x;   % Discrete points describing the surface of the body
r = body.shape.r;   % Discrete points describing the surface of the body

Ma = zeros(6,6); % Declare added mass coefficients matrix

% Inline function used in calculation for A55 or A66 component
% (  int(c*x^2 dx,x,a,b) )
intFact = inline('1/3*(x-xm)^3','x','xm');


% Contribution from body
% Using trapezoidal method
for i=2:length(x)
    rHat = (r(i)+r(i-1))/2;
    xHat = (x(i)+x(i-1))/2;
    delta = (x(i)-x(i-1));  % Width of panel
    Ma(2,2) = Ma(2,2) +...
        delta*pi*rho*rHat^2; % Contribution of current panel
    Ma(5,3) = Ma(5,3) + ...
        (xm-xHat)*...    % Moment arm
```

```
        delta*pi*rho*rHat^2; % Contribution of current panel
    Ma(5,5) = Ma(5,5) + rho*pi*rHat^2* delta * (xm-xHat)^2 ;
%                        \__________________/   \_________/
%                                dm                arm^2
end
Ma(1,1) = calcA11(r,max(x),rho); % Approximate body as an ellipsoid
Ma(3,3) = Ma(2,2);    % A33 equals A22 for a body of revolution
Ma(6,6) = Ma(5,5);    % A66 equals A55 for a body of revolution
Ma(6,2) = -Ma(5,3);

% Contribution from foil(s)
for i=1:length(foil)

    % calculate added mass terms due to foil(s)
        [A33_22, A53_62, A55_66,A44] = wingPanels(x,r,foil(i),xm,rho);

        if foil(i).Plane == 'XY'
            Ma(3,3) = Ma(3,3) + A33_22;
            Ma(5,3) = Ma(5,3) - A53_62;
            Ma(5,5) = Ma(5,5) + A55_66;

        elseif foil(i).Plane == 'XZ'
            Ma(2,2) = Ma(2,2) + A33_22;
            Ma(6,2) = Ma(6,2) + A53_62;
            Ma(6,6) = Ma(6,6) + A55_66;
        else
            error('Plane \"%s\" is not defined',a);
        end %if
        Ma(4,4) = Ma(4,4) + A44;
end % for

% Prepare output (According to Fossen (2002)) i.e Ma = Ma'
Ma = Ma + Ma' - diag(diag(Ma));


% If the wings are tapered they should be divided into panels,
% if not, the added terms are calculated using two dimensional theory
function [Ma33_22,Ma53_62,Ma55_66,Ma44] = wingPanels(x,r,foil,xm,rho)
        NDISCS = 40;            % Number of discrete panels
        xmin = foil.Xo;      % Leading edge of the foil root chord
        xmax = foil.Xo + foil.Cr;
        center = (xmin + xmax)/2;

        a = interp1(x,r,(xmin+xmax)/2); % Radius of body where foil is mounted
        B = foil.Span/2;     % foil semi-span
        Cr = foil.Cr;        % Root chord
        Ct = foil.Ct;        % Tip chord
        dx = (Cr-Ct)/2;         % Change in x direction
        dy = B-a;               % Change in Y corresponding to change dx

        if dx~=0    % The foil is tapered
            dydx = dy/dx;
            Xtemp = 0:Cr/NDISCS:Cr/2; % Position of discrete panels
            Ytemp = a + dydx*Xtemp;
            [i,j]=find(Ytemp>B); % Check if components of the panels have radius > semispan
            Ytemp(i,j) = B; % If so, radius of these panels = B
            % Mirror panels about center axis
            Bpanels = [Ytemp, Ytemp(length(Ytemp):-1:1)];
            Xpanels = xmin + [Xtemp,Xtemp + Cr/2];
            % Preparations for trapezoidal method
            b = (Bpanels(1:length(Bpanels)-1)+Bpanels(2:length(Bpanels)))/2;
            x = (Xpanels(1:length(Xpanels)-1)+Xpanels(2:length(Xpanels)))/2;
            delta = (Xpanels(2:length(Xpanels))-Xpanels(1:length(Xpanels)-1));
            integrationFactor = delta.*(xm-x).^2; % Numerical integration
        else    % The foil is not tapered
            b = B;
```

```
            delta = Cr;
            x = center;
            integrationFactor = 1/3*((xmax-xm)^3-(xmin-xm)^3); % Solution to int[(xm-x)^2 dx ]^xmax_xmin
        end

        % According to Newman, table 4.3 (But removed a^2; this part is already
        % calculated in body alone configuration)
        Ma33_22 = sum(rho*pi*delta.*(((b.^2-a^2).^2)./b.^2)); % Contribution from foil to A33 or A22
        Ma53_62 = sum(rho*pi*(xm-x).*delta.*(((b.^2-a^2).^2)/b.^2)); % Contr f.w to A53 or A62
        Ma55_66 = sum(rho*pi*(((b.^2-a^2).^2)./b.^2).*integrationFactor);

        % Find the A44 term for each panel of the foil
        alpha = asin((2*a*b)./(a^2+b.^2));
        alpha = pi-alpha;    % pi/2 < alpha < pi  (According to Newman, page 145)
        Ma44 =  sum(rho*a^4*delta.*(pi^-1 * (csc(alpha).^4).*... % Newman, page 145
            (2*alpha.^2 - alpha.*sin(4*alpha) + 0.5*sin(2*alpha).^2) - pi/2));



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%   Function used to estimate the added mass in surge (A_11). The
%%%%%%%%%   added mass is estimated assuming the bodyshape equals an ellipsoid with
%%%%%%%%%   main axis a and b  (a is the lengt, b is the diameter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function A11 = calcA11(Ri,Lpp,rho)
    n = 10; % Avrage the input in order to remove "noise"
    Rtemp = [];

    for i =1:length(Ri)/n
        if i*n<=length(Ri);temp = i*n ;else temp = length(Ri); end
        Rtemp = [Rtemp; mean(Ri(1+(i-1)*n: temp))];
    end
    D = 2*max(Rtemp);

    FinenessRatio = Lpp/D;

 % Data for an ellipsoid (Blevins-1979)
blevinsdata = [ 0.01000000000000   0.63480000000000
                0.10000000000000   0.61480000000000
                0.20000000000000   0.60160000000000
                0.40000000000000   0.57120000000000
                0.60000000000000   0.54470000000000
                0.80000000000000   0.52110000000000
                1.00000000000000   0.50000000000000
                1.50000000000000   0.45570000000000
                2.00000000000000   0.42000000000000
                2.50000000000000   0.39080000000000
                3.00000000000000   0.36600000000000
                5.00000000000000   0.29560000000000
                7.00000000000000   0.25100000000000
                10.00000000000000   0.20710000000000];
muInterp = interp1(blevinsdata(:,1),blevinsdata(:,2), FinenessRatio);   % Interpolate ellipsoid

    % Added mass for an ellipsoid (Blevins-1979).
    A11 = muInterp*4/3*pi*rho*(D/2)^3;
```

B.1.4 bodyAlone.m

```
function BodyOut = bodyalone(body, Xm)
% References:
% Blakelock, John H (1991), "Automatic control of aircrafts and missiles"
%        [Book]
% Bohlmann, Hans Jurgen (1990), "Berechnung hydrodynamischer Koeffizienten von
%       Ubooten zur Vorhersage des Bewegungsverhaltens". PhD-thesis
```

```
%       Universitat Hamburg
% Datcom, (1978), "USAF Stability and Control Datcom", :Authors: Hoak, D E AND Finck, R D
% Hoerner, Sighart F (1985), "Fluid dynamic lift" [Book]
% Myring, D. F (1975), "A theoretical study of body drag in subcritical
%       axisymmetric flow", Journal of Aeronautical Quarterly
% Pitts, William C & Nielsen, Jack N & Kaattari, George E (1957),"Lift and
%       center of pressure of foil-body-tail combinations at subsonic, transonic,
%       and supersonic speeds". NACA tecnical report 1307
% Roskam, Jan (1979), "Airplane flight dynamics and automatic flight
%       controls, Part1" [Book]

Lpp = body.Param.Lpp;
Body = body;
% Lift and moment curve for body
    Xo = body.SeparationPoint;
    So = pi*interp1(body.shape.x,body.shape.r,Xo)^2;
    dx = Xo/1000;
    Xi = 0:dx:Xo;
    Vo = pi*sum(dx*interp1(body.shape.x,body.shape.r,Xi).^2);


    f = Lpp/body.Param.d;    % Body fineness ratio
    k2_k1 = K2_K1(f);   % Function call, Apparent mass factor
    Body.derivatives.f_Info = 'Body fineness ratio (Lpp/d)';
    Body.derivatives.f = f;
    Body.derivatives.k2_k1_Info = 'Apparent mass factor';
    Body.derivatives.k2_k1 = k2_k1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%             Body alone derivatives
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%             Lift curve slope
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Body.derivatives.ClaInfo = 'Lift curve slope (Based on Lpp^2)';
Body.derivatives.Cla = 2*k2_k1*So/Lpp^2;   % Cla based on Lpp^2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%         Body pitching moment
%%%%%%%% The procedure below is derived in Håvard Bø's Msc thesis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Cla = Body.derivatives.Cla;
% Cma based on Lpp^3
Cma = Xm*Cla/Lpp+2*k2_k1/Lpp^3 *(Vo- Xo*So);
Body.derivatives.CmaInfo = 'Pitching moment, body alone, rel to Xm (Based on Lpp^3)';
Body.derivatives.Cma = Cma;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%         Hydrodynamic center, body alone
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Xhat_hc = Cma/Cla; % Hydrodynamic center of body, relative to XM (Based on Lpp)
    Xhc = Xhat_hc*Lpp; % Hydrodynamic center of body, relative to XM
    Body.Sim.Info = 'Hydrodynamic center of body (Rel to Xm- Used in simulation)';
    Body.Sim.pos = [Xhc,0,0]';
    Body.derivatives.Xhat_hc_Info = 'Hydrodynamic Center relative to Xm (based on Lpp)';
    Body.derivatives.Xhat_hc = Xhat_hc;
    Body.derivatives.XhcInfo = 'Hydrodynamic center, relative to Xm (m)';
    Body.derivatives.Xhc = Xhc;
    clear Xhat_hc Xhc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%       Calculate the Damping derivative CD_so according to Datcom
```

```
%%%%%%%%        CD_so = CD_marked + CD_base(CD_marked)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



    Rn = Lpp*body.Data.velocity/body.Data.KinViscousity; % Reynolds number
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Body.Data.Rn = Rn;
    Rn(find(Rn<300)) = 300;
    Rn(find(Rn>1e12)) = 1e12;
    % ITTC mean line
    Cf = (0.075)./(log10(Rn)-2).^2;

    Db =  2*body.shape.r(length(body.shape.r)-1); % Base diameter
    if Db< 1e-3; Db = 0; end
    Body.Sim.Db = Db;
    clear Db
    % syms cf
    % Schoenherr, flat plate friction drag
    % Cf = str2double(char(solve(['0.242/sqrt(Cf) = log10(',num2str(Rn),'*Cf)'])));

    Body.Sim.cdDIVcf = (1+60*(body.Param.d/body.Param.Lpp)^3 + ...
        0.0025*body.Param.Lpp/body.Param.d)*...
        body.WA/(pi*body.Param.d^2/4);

    CDfb = Cf * Body.Sim.cdDIVcf;
    Body.Sim.dbDIVd = Body.Sim.Db / body.Param.d;
    Body.Sim.dragNormal = (pi*body.Param.d^2/4)/body.Param.Lpp^2;
    CDbase = calcCDbase(CDfb, Body.Sim.Db, body.Param.d);
    CD_so = CDfb + CDbase;

    Cd = CD_so*(pi*body.Param.d^2/4)/body.Param.Lpp^2;  % Cd based on Lpp^2
    Body.derivatives.Cd_Info = 'Cd (according to Datcom, based on Lpp^2)';
    Body.derivatives.Cd = Cd;
    Body.derivatives.CDbaseInfo = 'Base drag (based on frontal area)';
    Body.derivatives.CDbase = CDbase;
    Body.derivatives.CDfbInfo = 'Zero lift drag (based on frontal area)';
    Body.derivatives.CDfb = CDfb;

    clear CDbase CDfb CD_so Cd Cf % Clear temporary variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%        Calculate the "pitching-damping derivative",
%%%%%%%%        (Hoak & Finch 1978, Sec 7.2.1.2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Sb = pi*Body.Sim.Db^2;   % Base area (area of the stern)
    CmqNum = (1-Xm/Lpp)^2*Sb - body.Volume/Lpp*...
       ((body.Xc- Xm)/Lpp);
    CmqDen = (1-Xm/Lpp)*Sb - body.Volume/Lpp;
    Cmq = 2*Cma* CmqNum/CmqDen;
    Body.derivatives.Cmq_Info = 'Based on Lpp^4';
    Body.derivatives.Cmq = Cmq;
    clear CmqNum CmqDen Sb  % Clear temporary variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%        Calculate Clq,
%%%%%%%%        source: Datcom (1978)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Body.derivatives.ClqInfo = 'Clq, body alone (Based on Lpp^3)';
    Body.derivatives.Clq = 2*Body.derivatives.Cla* ...
        (1-Body.derivatives.Xhat_hc)*...
```

```
        (pi/4*Body.Sim.Db^2)/Lpp^2;
    clear f k2_k1 Cmq Cma Cla Xhc Xhc% Clear temporary variables


BodyOut = Body;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function used to calculate the apparent mass factor
% input 'f' - finenessratio
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function out = K2_K1(f)
x = [4 6 8 10 12 14 16 18 20];
K = [0.78 0.86 0.91 0.94 0.952 0.963 0.97  0.975 0.98];
out = interp1(x,K,f);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function used to calculate the base drag coefficient
% Source: Datcom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function CDbase = calcCDbase(CD_marked, dBase, dmax);
    CDbase = 0.029*((dBase/dmax)^3)/CD_marked;
```

B.1.5 bodyCalc.m

```
function Body = bodyCalc(BodyInput)
% References:
% Roskam, Jan (1979), "Airplane flight dynamics and automatic flight
%       controls, Part1" [Book]
% Blakelock, John H (1991), "Automatic control of aircrafts and missiles"
%        [Book]
% Hoerner, Sighart F (1985), "Fluid dynamic lift" [Book]
% Myring, D. F (1976), "A theoretical study of body drag in subcritical
%       axisymmetric flow", Journal of Aeronautical Quarterly
% Pitts, William C & Nielsen, Jack N & Kaattari, George E (1957),"Lift and
%       center of pressure of wing-body-tail combinations at subsonic, transonic,
%       and supersonic speeds". NACA tecnical report 1307

NDISC = 3000;   % Number of discrete sections used in forthcoming calculations
availablePro = '';
availablePro = sprintf('%s %s\n',availablePro,'vertex');
availablePro = sprintf('%s %s\n',availablePro,'torpedo');

% ************************************************************************
% Chech for valid inputs and Call the correct procedure for construction of
% the body shape
try
    Body = BodyInput;
    temp = feval(sprintf('%s',BodyInput.Profile), BodyInput.Param, NDISC);
    Body.shape = temp.shape;
    Body.Param.d = 2*max(Body.shape.r);
    try Body.Curve = temp.Curve;end % If the body is torpedo shaped
catch
    temp = sprintf('Procedure %s is not recognized as a valid function, \n', BodyInput.Profile);
    temp = sprintf('%s- or incorrect parameter input.\n',temp);
    temp = sprintf('%sAvailable procedures are:\n',temp);
    error(sprintf('%s%s%s',temp,availablePro,lasterr));
end
% ************************************************************************



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Function used for approximation of a body described by a finite number of
% vertexes. The body is discretizised with the help of Cubic Hermite
% interpolation (pchip(x,y,xx); built in fnction in MatLab)
%
% Output:  structure = vertex(Param, NDISC)
%       where the structure is a STRUCTURE contating the following fields:
%           x:  1xNDISC vector of discrete x positions
%           y:  1xNDISC vector, radius:    r(x)
%
% Input:
%  Param. (A structure containing x, y, Lpp)
%       x:  A vector (1xN) of the position of the x position of the
%           N points
%       r:  A vector (1xN) of the radius corresponding to the x
%           position of the points
%       Lpp:    Length between perpendicular (Length of vehicle) [m]
%
%  NDISC:   Number of discrete panels
function outParam =  vertex(Param, NDISC)
    x = Param.x;    % x position of points
    r = Param.r;    % r(x)
    Xi = 0:Param.Lpp/NDISC:Param.Lpp;

    Ri = pchip(x ,r , Xi); % Cubic Hermite interpolation of body surface
    shape.Info = sprintf('r(x) for the discrete body, \n(Interpreted with help of Cubic Hermite)');
    shape.x = Xi;   % xpos of discrete points
    shape.r = Ri;   % Radius of discrete points
    outParam.shape = shape;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       End of vertex function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function used for calculation of the shape of a torpedo vehicle
% Nose and tail sections are computed according to the polynomals given in
% Myring (1976).
%
% Output:  structure = torpedo(Param, NDISC)
%       where structure is a (1x3) dimensional STRUCTURE each contating
%       the following fields:
%               Curve:      Inline function Curve(x), returns the radius of
%                           the section at a given point
%               min:        Start of the section (relative to nose origin)
%               max:        End of the section (relative to nose origin)
%
% Input:
%  Param.
%   a:      Length of nose section [m]
%   b:      Distance from nose to beginning of tail section [m]
%   d:      Diameter of the cylindrical section [m]
%   n:      Parameter which determines the shape of the nose
%   theta:  Half of the included angle at the tip of the tail [deg]
%   Lpp:    Length between perpendicular (Length of vehicle) [m]
%   Lpoint: Length of vehicle if tail should end in a point (Lpoint>=Lpp)
%
%  NDISC:   Number of discrete panels

function out = torpedo(Param, NDISC)
    a = Param.a; b = Param.b;
    D = Param.d; n = Param.n;
    theta = Param.theta;
```

```
Lpp = Param.Lpp; Lpoint = Param.Lpoint;

syms x;
% Check for valid input
if theta<0;
    error(sprintf('Incorrect input, theta should be a positive number (theta = %1.3f)', theta)); end
if Lpp>Lpoint; error('Lpp > Lpoint (Incorrect description of torpedo shaped vehicle)'); end
if a>b; error('a>b (Tail starts before nose ends)'); end
if n>3; warning(sprintf('Parameter, n=%2.2f, for determination of nose shape is large',n)); end

theta = theta*pi/180;
% Start description of nose section
nose.Info = 'Nose section';
nose.Curve = inline(char(1/2*D*(1-(x-a)^2/a^2)^(1/n)));
nose.min = 0;   % Start of nose section
nose.max = a;   % End of nose section

% Start description of cylindrical section
mid.Info = 'Cylindrical section';
mid.Curve = inline(sprintf('%2.4f',1/2*D),'x');
mid.min = a;    % Start of cylindrical section
mid.max = b;    % End of cylindrical section

% Start description of tail section
tail.Info = 'Tail section';
tail.Curve = 1/2*D-(3/2*D/(Lpoint-b)^2-tan(theta)/(Lpoint-b))*...
    (x-b)^2+(D/(Lpoint-b)^3-tan(theta)/(Lpoint-b)^2)*(x-b)^3;
tail.Curve = inline(char(tail.Curve));
tail.min = b;   % Start of tail section
tail.max = Lpp; % End of tail section

% Prepare return value (Insert the sections into an array of
% structures)
Curve(1) = nose;
Curve(2) = mid;
Curve(3) = tail;

minMax = [Curve.min;Curve.max]; % Min and max for the curves
Rpos = [];

numberOfPoints = NDISC;   % Number of points
Xpos = 0:max(minMax(2,:))/numberOfPoints:max(minMax(2,:))-max(minMax(2,:))/numberOfPoints;

% Discretisize the body into NDISC panels
for i = 1:length(Xpos)
    % Find the curve belonging to current x position
    ActiveCurve = find((minMax(1,:)<=Xpos(i)) & (minMax(2,:)>=Xpos(i)));
    x = Xpos(i);
    % Find radius at the current position
    Rpos = [Rpos, real(Curve(ActiveCurve(1)).Curve(Xpos(i)))];
end
Xpos = [Xpos, Lpp];   % Close body
Rpos = [Rpos, 0];
out.Curve = Curve;
out.shape.Info = 'r(x) for the discrete body';
out.shape.x = Xpos;
out.shape.r = Rpos;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       End of torpedo function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B.1.6 calcVolumSurface.m

```
function out = calcVolumSurface(Body)
% Procedure for calculating volume, wetted surface, momentum and center of
% damping for a body of revolution
```

```
%
% out = calcVolumSurface(Body)
%
% Input:
%       Body:    a structure of type body (output from bodyCalc.m)
% Output:   a structure containing following fields
%           out.Volume  -   Volume of the body
%           out.WA      -   Wetted area of the body
%           out.Sx      -   Volume momentum
%           out.CenterOfDamping -   Center of damping
%           out.SeparationPoint -   the point where separation is likely to
%                                   occour

x = Body.shape.x;
r = Body.shape.r;

% Prepare for trapezoidal method
Xi = (x(1:length(x)-1)+x(2:length(x)))/2;
Ri = (r(1:length(r)-1)+r(2:length(r)))/2;

delta = (x(2:length(x))-x(1:length(x)-1)); % Find width of slice (panel)
% out = Body;

% Find the separtation point for the body.
switch lower(Body.Profile)
case 'torpedo'
    if Body.Data.AutoSep; sepPoint = Body.Param.a; % Seperation point of torpedo
    else sepPoint = Body.Data.sepPoint; end
    % Calculate nose section
    a = Body.Param.a;
    b = Body.Param.b;
    nose = find(Xi<=a);
    mid = find(Xi(find(Xi<=b))>a);
    tail = find(Xi>b);
    % Volume of the various sections
    Vol.info = 'Volume of nose section';
    Vol.val =  volume(Ri(nose), delta(nose));
    Vol(2).info = 'Volume of cylindrical section';
    Vol(2).val = volume(Ri(mid), delta(mid));
    Vol(3).info = 'Volume of tail section';
    Vol(3).val = volume(Ri(tail), delta(tail));
    % Wetted area of the various sections
    WA.info = 'Wetted Area, nose section';
    WA.val =  wettedArea(Ri(nose), delta(nose));
    WA(2).info = 'Wetted Area, cylindrical section';
    WA(2).val = wettedArea(Ri(mid), delta(mid));
    WA(3).info = 'Wetted Area, tail section';
    WA(3).val = wettedArea(Ri(tail), delta(tail));
    % SX for the various sections
    Sx.info = 'Sx, nose section';
    Sx.val =  momentSx(Ri(nose), delta(nose), Xi(nose));
    Sx(2).info = 'Sx, cylindrical section';
    Sx(2).val = momentSx(Ri(mid), delta(mid), Xi(mid));
    Sx(3).info = 'Sx, tail section';
    Sx(3).val = momentSx(Ri(tail), delta(tail), Xi(tail));

    clear nose mid tail a b % Clear temporary variables
    % Prepare output;
    out.Sections.Volume = Vol;
    out.Sections.WA = WA;
    out.Sections.Sx = Sx;

case 'vertex'
    if Body.Data.AutoSep
        % The separation point is on the most negative curve slope
        % for a stremlined body of revolution
```

```
        n = 10; % Avrage the input in order to remove "noise"
        Rtemp = [];
        Xtemp = [];
        for i =1:length(Ri)/n
            if i*n<=length(x);temp = i*n ;else temp = length(x); end
            Rtemp = [Rtemp; mean(Ri(1+(i-1)*n: temp))];
            Xtemp = [Xtemp; mean(Xi(1+(i-1)*n: temp))];
        end
        [aTemp, bTemp] = min(diff(Rtemp));
        sepPoint = Xtemp(bTemp);    % Most negative slope
    else
        sepPoint = Body.Data.sepPoint;
    end % if autocalculate separation point

otherwise
    error(sprintf('Profile %s is not reckognized as a valid input',Body.Profile));
end % if
clear aTemp bTemp Rtemp Xtemp n

% Prepare output structure
outTemp = Body;
outTemp.SeparationPoint = sepPoint;
outTemp.Volume = volume(Ri,delta);
outTemp.WA = wettedArea(Ri,delta);
outTemp.Sx = momentSx(Ri,delta,Xi);
outTemp.Xc_Info = 'Centre of buoyancy';
outTemp.Xc = outTemp.Sx/outTemp.Volume;

out = outTemp;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%  Functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate the volume
function Vol = volume(r,dx)
    Vol = sum(pi*(r.^2).*dx);

% Calculate the wetted area
function Wa = wettedArea(r,dx)
    Wa = sum(2*pi*r.*dx);

% Calculate the momentum contribution
function Sx = momentSx(r,dx,x)
    Sx = sum(pi*(r.^2).*x.*dx);
```

B.1.7 convert2sname.m

```
function out = convert2sname(Ma,Mrb,derivatives, L, Rho)
% out = convert2sname(Ma, Mrb, Der, L, Rho)
%
% calculate the nondimensional static and dynamic parameters for a body
%
% input:
%   Ma  - 6x6 added mass matrix (dimensional)
%   Mrb - 6x6 rigid body system inertia matrix
%   Der - Structure of derivatives (format given by dynDerivatives.m)
%   L   - Characteristic length, used to nondimenionalize Mrb & Ma
%   Rho - Water density (kg/m^3)
%
% output:
%   out - Structure of nondimensional parameters for Ma, Mrb, and D

% Check for correct input
if L <= 0
    error(sprintf('L=%1.2f < 0 \n=> Characteristic length must be a positive number',L));
```

```
elseif min(size(Mrb) ~= [6 6])
    error('Incorrect size of Mrb, should be a 6x6 matrix');
elseif min(size(Ma) ~= [6 6])
    error('Incorrect size of Ma, should be a 6x6 matrix');
elseif Rho<=0
    error('Density, Rho, should be a poitive number');
end

out.Ma = massConvertion(Ma, L, Rho);
out.Mrb = massConvertion(Mrb, L, Rho);

D = zeros(6,6);
bodyDer = derivatives.Body;
for i = 1:length(derivatives.Wing)  % For all wings
    wingDer = derivatives.Wing(i);
    wingBodyDer = derivatives.WingBody(i);
    if lower(wingDer.Plane) == 'xy'
        D = D + convertDerivativesXY(bodyDer, wingDer, wingBodyDer, out.Ma);
    elseif lower(wingDer.Plane) == 'xz'
        D = D + convertDerivativesXZ(bodyDer, wingDer, wingBodyDer, out.Ma);
    else error('Unknown orientation of wing section');
    end %if
end % for

out.D = D;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%        Local functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function nonDim = massConvertion(M,L, Rho)
    nonDim = zeros(6,6);    % Initialize matrix
    for i = 1:6
        for j = 1:6
            if  max(i==[1 2 3]) && max(j==[1 2 3])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^3);
            elseif max(i==[1 2 3]) && max(j==[4 5 6])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^4);
            elseif max(i==[4 5 6]) && max(j==[1 2 3])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^4);
            elseif max(i==[4 5 6]) && max(j==[4 5 6])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^5);
            else error('Something is wrong');
            end % if
        end %for
    end % for
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% end of massConvertion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function D = convertDerivativesXY(body, wing, wingBody, Ma);
    temp = zeros(6,6);

    temp(3,3) = - (body.Cd + wingBody.Cla);
    temp(3,5) = - (wingBody.Clq + Ma(1,1));
    temp(5,3) = wingBody.Cma;
    temp(5,5) = wingBody.Cmq;

    D = temp;

function D = convertDerivativesXZ(body, wing, wingBody, Ma);
    temp = zeros(6,6);

    temp(2,2) = - (body.Cd + wingBody.Cla);
    temp(2,6) = - (wingBody.Clq + Ma(1,1));
    temp(6,6) = wingBody.Cmq;
```

```
    D = temp;
```

B.1.8 dampingTerms.m

```
function [dampDim, varargout] = dampingTerms(Body,Wing)
    derivatives = Body.Der;
    rho = Body.Param.Rho;
    Lpp = Body.Param.Lpp;

   bodyDer = derivatives.Body;
   D = zeros(6,6);
   D(1,1) = -bodyDer.Cd;
   D(2,2) = -bodyDer.Cd;
   D(3,3) = -bodyDer.Cd;

   for i = 1:length(derivatives.Wing)  % For all wings
       wingDer = derivatives.Wing(i);
       wingBodyDer = derivatives.WingBody(i);
       D(1,1) = D(1,1) - derivatives.Wing(i).C_D;

       switch lower(wingDer.Plane)
           case 'xy'
               D = D + convertDerivativesXY(bodyDer, wingDer, wingBodyDer);
           case 'xz'
               D = D + convertDerivativesXZ(bodyDer, wingDer, wingBodyDer);
           otherwise
               error('Unknown orientation of wing section');
       end %switch
       D(4,4) = rollDamping(Body,Body.Der.WingBody(i),Wing(i));
   end % for
   % Non dimensional formulation, (optionally output)
   varargout{1} = -D; % Convert from SNAME to Fossen
   % Dimensional formulation, (standard output)
   basedOn = diag([2 2 2 3 3 3]);
   basedOn(1,2:6) = 3;
   basedOn(2,[1,3:6]) = 3;
   basedOn(3,[1:2,4:6]) = 3;
   basedOn(4,[1:3,5:6]) = 4;
   basedOn(5,[1:4,6]) = 4;
   basedOn(6,[1:5]) = 4;

   Lpp = Lpp*ones(6,6);
   dampDim = -0.5*rho*D.*(Lpp.^basedOn);


   % Make the damping Matrix Dimensional


% Functions ued to form the damping matrix
function D = convertDerivativesXY(body, wing, wingBody);
    temp = zeros(6,6);
    temp(3,3) = - wingBody.Cla; % See Blackelock Eq 1-82
    temp(3,5) = - (wingBody.Clq);
    temp(5,3) = wingBody.Cma;
    temp(5,5) = wingBody.Cmq;
    D = temp;

function D = convertDerivativesXZ(body, wing, wingBody);
    temp = zeros(6,6);
    temp(2,2) = - wingBody.Cla; % See Blackelock Eq 1-82
    temp(2,6) = - (wingBody.Clq);
    temp(6,2) = wingBody.Cma;
    temp(6,6) = wingBody.Cmq;
```

```
    D = temp;

 function Dpp = rollDamping(Body,WingBody,Wing)
    Yr = mean(Wing.Curve.Discrete.root.z);
    Yt = mean(Wing.Curve.Discrete.tip.z);
    Ndisc = 100;
    dY = (Yt-Yr)/Ndisc;
    dXdY = (Wing.Ct-Wing.Cr)/(Yt-Yr); %
    syms y;
    CofY = inline(char(Wing.Cr + dXdY*(y-Wing.Cr)));
    Yi = Yr:dY:Yt;
    Ci = CofY(Yi);
    % Solve the integral numerically
    dCldA = WingBody.ClaWB*Wing.extendedWing.S/Wing.exposedWing.S;
    Dpp = -sum(2*dCldA*Yi.*Ci.*dY);
    Dpp = Dpp/Body.Param.Lpp^3; % Nondimensionalize with respect to Lpp
```

B.1.9 editRigidBody.m

```
function Mrb = editRigidBody(Min, varargin)

% Created by Håvard Bø 28-Apr-2004
try
    figpos = varargin{1};
catch
    figpos = [10 10];
end

arr_size = size(Min);
bd_size = 1; %border size
rLENGTH = 10; % Length of each element
rHEIGHT = 1.3; % Height of each element

buttonHeight = 2;  buttonLength = 15; ButtonSpace = 2;  buttonYpos = 1;
fig_dim(1) = (arr_size(2))*(rLENGTH+bd_size) +bd_size;
fig_dim(2) = (arr_size(1)+1)*(rHEIGHT + bd_size) +  buttonHeight;
%fig_dim = [(arr_size(1))*(rLENGTH+bd_size) +bd_size, (arr_size(2) +1)*rHEIGHT + buttonHeight];
midFig = fig_dim(1)/2;

% Define where the placement of the buttons
    okButtonPos = [midFig-buttonLength-ButtonSpace/2, buttonYpos, buttonLength, buttonHeight];
CaButtonPos = [midFig+ButtonSpace/2, buttonYpos, buttonLength, buttonHeight];
arr_fig = figure('unit','characters','NumberTitle','off','Menubar','none','resize','on','position',...
    [figpos(1), figpos(2), fig_dim(1), fig_dim(2)]);

ok_but = uicontrol('Style','pushbutton','unit','characters','String','OK','position',okButtonPos,...
    'callback','uiresume','tag','ok');
cancel_but = uicontrol('Style','pushbutton','unit','characters','String','Cancel','position',CaButtonPos,...
    'callback','uiresume','tag','cancel');
set(arr_fig, 'Name', 'Rigid Body')

posxb = bd_size;
posy = fig_dim(2)-2*bd_size;
% Create table
for i = 1:arr_size(1)
    posx = bd_size;
    for j = 1:arr_size(2)
        a(i,j)=uicontrol('Style','edit','unit','characters','position', [posx, posy, rLENGTH, rHEIGHT]);
        set(a(i,j),'string', sprintf('%2.3f',Min(i,j)));
        if ispc;    set(a(i,j),'BackgroundColor','white');
        else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')); end
        posx = posx + bd_size + rLENGTH;
    end
    posy = posy-rHEIGHT-bd_size;
end
```

```
    uiwait;
but = gco;

Mrb = zeros(size(Min));
if strcmp(lower(get(but,'tag')),'ok')
    Mrb(:) = str2double(get(a,'string'));
    close;
elseif  strcmp(lower(get(but,'tag')),'cancel')
    Mrb = Min;
    close;
else
    Mrb = Min;
end
```

B.1.10 editVertexes.m

```
function x = editvertexes(xy, varargin);
% Created by Håvard Bø 28-Apr-2004

%function x = editvertexes(r,c);
% X = editvertexes(xy) is used for interactive editing of vertexes
% (can be used in GUI applications in the same manner as the array editor from command line)
%
% xy is an array of [x;y] positions of the vertexes

try
    figpos = varargin{1};
catch
    figpos = [10 10];
end

arr_size = size(xy);
bd_size = 1; %border size
rLENGTH = 10; % Length of each element
rHEIGHT = 1.3; % Height of each element

buttonHeight = 2;  buttonLength = 15; ButtonSpace = 2;  buttonYpos = 1;
arr_size = [2,length(xy)];
fig_dim(1) = (arr_size(2)+1)*(rLENGTH+bd_size) +bd_size;
fig_dim(2) = (arr_size(1)+1)*(rHEIGHT + bd_size) +  buttonHeight;

midFig = fig_dim(1)/2;

% Define where the placement of the buttons
okButtonPos = [midFig-buttonLength-ButtonSpace/2, buttonYpos, buttonLength, buttonHeight];
CaButtonPos = [midFig+ButtonSpace/2, buttonYpos, buttonLength, buttonHeight];

arr_fig = figure('unit','characters','NumberTitle','off','Menubar','none','resize','on','position',...
    [figpos(1), figpos(2), fig_dim(1), fig_dim(2)]);
ok_but = uicontrol('Style','pushbutton','unit','characters','String','OK','position',okButtonPos,...
    'callback','uiresume','tag','ok');
cancel_but = uicontrol('Style','pushbutton','unit','characters','String','Cancel','position',CaButtonPos,...
    'callback','uiresume','tag','cancel');
set(arr_fig, 'Name', 'Edit Vertexes, [0;0] elements will be removed')

posxb = bd_size;
posy = fig_dim(2)-2*bd_size;

% Create tables
for i = 1:arr_size(1)
    posx = bd_size;
    a(i,1)=uicontrol('Style','text','unit','characters','position', [posx posy rLENGTH rHEIGHT],...
        'horizontalalignment', 'right');

    for j = 2:arr_size(2)+1
        posx = posx + bd_size + rLENGTH;
```

```
        a(i,j)=uicontrol('Style','edit','unit','characters','position', [posx posy rLENGTH rHEIGHT]);
        set(a(i,j),'string', sprintf('%1.3f',xy(i,j-1)));
        if ispc;    set(a(i,j),'BackgroundColor','white');
        else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor')); end
    end
    posy = posy-rHEIGHT-bd_size;
end
set(a(1,1),'string','x');
set(a(2,1),'string','r(x)');
uiwait;
but = gco;

if strcmp(lower(get(but,'tag')),'ok')
    [r,c] = size(a);
    if max(size(a)==[0 0]);
        x = [];
    else
        for i = 1:r
            for j = 2:c; x(i,j-1) = str2double(get(a(i,j),'string'));
            end % for
        end % for
        close;
    end % if
elseif  strcmp(lower(get(but,'tag')),'cancel')
    x = [];
    close;
else
    x = [];
end
```

B.1.11 foilalone.m

```
function FoilOut = foilalone(Foil, Xm, Body)
if length(Foil)<1
    error('Incorrect foil Input')
end

for i = 1:length(Foil)
    FoilOut(i) = calcFoils(Foil(i), Xm, Body);
end


function FoilOut = calcFoils(Foil, Xm, Body)
    Lpp = Body.Param.Lpp;
    dx = 0.01; x = 0:dx:1;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculation of C_{L_alpha} using Whicker & Doenhoff (1958)
    Foil.ClaInfo = 'Lift curve slope, Cla (Based on Lpp^2)';
    eta = 0.9; % Correction factor
    Foil.Cla = Foil.S/Lpp^2* ...
        2*pi*Foil.AR/...
        (2 + 1/eta * sqrt( (Foil.AR/cos(Foil.Sweep.C4))^2 + 4*cos(Foil.Sweep.C4)^2));
    clear eta;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate Drag coefficients according to Datcom(1978)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Foil.CdoInfo1 ='Parasite drag coeff, C_D, (Based on Lpp^2)';

    Ytemp = Foil.Curve.upper(x);
    TdivC = mean(Ytemp);    % Average thickness ratio
    [temp, Cxmax] = max(Ytemp); % Find the chordwise position of maximum thickness
    Cxmax = x(Cxmax);
    theta  =  cos(sweepAngle(Cxmax, 0, Foil.AR, Foil.lambda));
```

```
    Rls = -0.993*theta^2 + 2.004*theta + 0.061; % See Bø, H Msc- Thesis
    if Cxmax<0.3; L = 2;
    else L = 1.2; end

    Chat = (Foil.Cr + Foil.Ct)/2; % Mean hydrodynamic chord
    Rn = Foil.Cmac*Body.Data.velocity/Body.Data.KinViscousity; % Reynolds number
    Foil.Rn = Rn;
    % Schoenherr, flat plate friction drag (Not used. underpredict)
    %    Cf = str2double(char(solve(['0.242/sqrt(Cf) = log10(',num2str(Rn),'*Cf)'])));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ITTC mean line

    Rn(find(Rn<300)) = 300;
    Rn(find(Rn>1e12)) = 1e12;
    % ITTC mean line
    Cf = (0.075)./(log10(Rn)-2).^2;

    Foil.Sim.cdDIVcf = (1 + L*TdivC + 100*TdivC^4)*Rls*...
        Foil.Swf/Lpp^2;
    Foil.Cdo = Cf*Foil.Sim.cdDIVcf;
    clear dx x Rls theta temp Cxmax TdivC Y Cxmax Ytemp L ;% Clear temporary variables
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate Clq (dynamic derivative)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Foil.ClqInfo ='Clq, Dynamic derivative (Based on Lpp^3)';
    Chat = 2/3*Foil.Cr*...
        (1+Foil.lambda+Foil.lambda^2)/(1+Foil.lambda);
    Foil.Clq = 1/Lpp*...
        (Chat/2 + 2*(Foil.hc - Xm))*Foil.Cla;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate Cmq (dynamic derivative). Source: Datcom (1978)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Foil.CmqInfo ='Cmq, Dynamic derivative (Based on Lpp^3)';
    xc = (Foil.hc-Xm)/Chat;
    Upsilon  = Foil.AR * (0.5*xc + 2*xc^2)/(Foil.AR + 2*cos(Foil.Sweep.C4)) + 1/24*...
        (Foil.AR^3)*tan(Foil.Sweep.C4)^2/(Foil.AR + 6*cos(Foil.Sweep.C4)) + 1/8;

    Foil.Cmq = -0.7*Foil.cla2d*...
        Foil.S*Chat^2/Lpp^4*...
        Upsilon * cos(Foil.Sweep.C4);
    clear Chat xc;% Clear temporary variables
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Insert the posistions of the 2 foils hydrodynamic center (used in
    % Simulation)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if strcmp(lower(Foil.Plane), 'xy')
        yi = Foil.Ymac;
        zi = 0;
    elseif strcmp(lower(Foil.Plane), 'xz')
        yi = 0;
        zi = Foil.Ymac;
    else
        error(['Plane is not defined: ', Foil.Plane]);
    end
    Foil.Sim.Info = 'Hydrodynamic center of foilpanel (Used in simulation)';
    Foil.Sim.pos = [Xm-Foil.hc,yi,zi]';
    Foil.Sim.neg = [Xm-Foil.hc,-yi,-zi]';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    FoilOut = Foil;

function SAn = sweepAngle(n, LAMBDA, AR, lambda)
m = 0.5;
SAn = atan(tan(LAMBDA) - 4*(n-m)/AR*(1-lambda)/(1+lambda));
```

B.1.12 foilBodyComb.m

```
function [FoilBody, Totaldrag] = foilBodyComb(foil,body,Xm)
% Todo: Check for Cmq and Clq wheter to use Cp or Xm
%       check sign of Cldelta and Cmdelta
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%       Calculate dynamic derivatives for the foil, body combination
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% function out = dynDerivatives(foil,body,Xm)
%
% Input:
%       foil - Structure of foil input (format given by foilalone.m)
%       body - Structure of body parameters (format given by bodyalone.m)
%       Xm   - Reference of calculation (assuming y=z=0)
% Output:
%       Structure of dynamic derivatives
%
% References:
% Blakelock, John H (1991), "Automatic control of aircrafts and missiles"
%        [Book]
% Bohlmann, Hans Jurgen (1990), "Berechnung hydrodynamischer Koeffizienten von
%       Ubooten zur Vorhersage des Bewegungsverhaltens". PhD-thesis
%       Universitat Hamburg
% Hoerner, Sighart F (1985), "Fluid dynamic lift" [Book]
% Myring, D. F (1975), "A theoretical study of body drag in subcritical
%       axisymmetric flow", Journal of Aeronautical Quarterly
% Pitts, William C & Nielsen, Jack N & Kaattari, George E (1957),"Lift and
%       center of pressure of foil-body-tail combinations at subsonic, transonic,
%       and supersonic speeds". NACA tecnical report 1307
% Roskam, Jan (1979), "Airplane flight dynamics and automatic flight
%       controls, Part1" [Book]


RFB = 1.05; % Foil body interference correlation factor. (Interpolated data, (Datcom, Figure 4.3.3.1-37))
Totaldrag = (body.derivatives.CDfb*RFB + body.derivatives.CDbase)*...
    pi*body.Param.d^2/(4*body.Param.Lpp^2); % From frontal area => Lpp^2
if length(foil) ==0;
    FoilBody = [];
    return
end

for i = 1:length(foil)
    % Radius of the body at the place where the foil is mounted
    %   (at center of root chord)
    r = interp1(body.shape.x,body.shape.r,foil(i).Xo+foil(i).Cr/2);
    % Calculate the derivatives for th current combination
    FoilOut(i) = foilbodyInteraction(foil(i), r, body, Xm);
    Totaldrag = Totaldrag + foil(i).Cdo*RFB;
%    WingDerivatives(i) = foil(i).derivatives;
end


FoilBody = FoilOut;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%            Functions used in calculations
%%%%%%%%        Calculates foil-body interaction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function foil_body = foilbodyInteraction(foil,r, body, Xm)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate the hydrodynamic center contribution (X'_ac/Cre)_b(w)
    % Expression is constructed according to Blakelock (1991, page 560)
    % [beta = sqrt(abs(1-M^2))= 1 (where M is the mach number) is 1 for all
    % practical AUV/marine applications]- ref: Håvard Bø 30.03.2004
    s = foil.Span/2; % foil semispan (temporary variable)
    b = foil.Span;  % foil-span (temporary variable)
    D = 2*r;

    A= foil.AR;
    Cr = foil.Cr;
    Lpp = body.Param.Lpp;

    BodyFoil.Plane = foil.Plane;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate  foil body interference factors
    % Source Pitts et.al (1957)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [Kfb, Kbf, kfb, kbf] = interferenceFactors(D,b);
    BodyFoil.Kfb = Kfb;
    BodyFoil.Kbf = Kbf;
    BodyFoil.kfb = kfb;
    BodyFoil.kbf = kbf;
     clear Kfb Kbf kfb kbf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Lift curve slope, foil body combination
    % Source Datcom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    BodyFoil.ClaInfo = 'Lift curve slope foil body combination (Based on Lpp^2)';
    BodyFoil.Cla = (BodyFoil.Kfb+BodyFoil.Kbf)*foil.Cla + body.derivatives.Cla;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Lift from control surface deflection
    % Source Datcom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
BodyFoil.CldeltaInfo = 'Lift from control surface deflection (Based on Lpp^2)';
BodyFoil.Cldelta = (BodyFoil.kfb+BodyFoil.kbf)*foil.Cla;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate hydrodynamic center due to lift carryover
    % Source Datcom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
BodyFoil.sim.XhcInfo = 'Hydrodynamic center, due to lift carryover, relative to Xm (m)';
BodyFoil.sim.Xhc = calcHcfoilbodyComb(foil, foil.D, Xm);
BodyFoil.sim.XhcHatInfo = 'Hydrodynamic center, due to lift relative to Xm (based on Lpp)';
BodyFoil.sim.XhcHat = BodyFoil.sim.Xhc/Lpp;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate the pitching moment
```

```
    % Source Datcom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    BodyFoil.CmaInfo = 'Pitching moment (Based on Lpp^3)';
    BodyFoil.Cma = body.derivatives.Cma + ...
        (Xm-foil.hc)/Lpp * BodyFoil.Kfb * foil.Cla + ...
        BodyFoil.sim.XhcHat * BodyFoil.Kbf * foil.Cla;
        % Contribution from body +
        % body in presence of foil +
        % foil in presence of body
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate the Hydrodynamic center
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    BodyFoil.XhcInfo = 'Hydrodynamic center of pressure, relative to Xm (Based on Lpp)';
    BodyFoil.Xhc = BodyFoil.Cma/BodyFoil.Cla;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate the pithing moment from due to foil incidense
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    BodyFoil.CmdeltaInfo = 'Pitching moment due to foil incidence (Based on Lpp^3)';
    BodyFoil.Cmdelta = -BodyFoil.Cldelta * BodyFoil.sim.XhcHat;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate the  pitching derivative Clq
    % Source: Datcom (1978)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    BodyFoil.ClqInfo = 'foil pitching derivative, Clq (Based on Lpp^3)';
    BodyFoil.Clq = (BodyFoil.Kfb + BodyFoil.Kbf)*foil.Clq+ body.derivatives.Clq;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculate the  pitching derivative Cmq
    % Source: Datcom (1978)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    BodyFoil.CmqInfo = 'foil pitching derivative, Cmq (Based on Lpp^4)';
    BodyFoil.Cmq = (BodyFoil.Kfb + BodyFoil.Kbf)*foil.Cmq+ body.derivatives.Cmq;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    foil_body = BodyFoil;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%       End of function foilbodyInteraction(...)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Local functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Foil body interference factors
% Source: Pitts et.al (1957)
function [Kfb, Kbf, kfb, kbf] = interferenceFactors(d,b)
r = d/2;
s = b/2;
    Kwb_num = (1+(r/s)^4)*((1/2*atan(1/2*(s/r-r/s)))+pi/4) - (r/s)^2*((s/r-r/s)+2*atan(r/s));
    Kwb_den = (1-r/s)^2;
    Kfb = (2/pi)*Kwb_num/Kwb_den; clear Kwb_num Kwb_den;

    Kbw_num = (1-r^2/s^2)^2-2/pi * ...
        ((1+r^4/s^4)*(1/2*atan(1/2*(s/r-r/s)) + pi/4) - ...
        r^2/s^2*((s/r-r/s) +2*atan(r/s)));
    Kbw_den = (1-r/s)^2;
```

```
    Kbf = Kbw_num/Kbw_den; clear Kbw_den Kbw_num

    tau = b/d;
    kfb = 1/pi^2*...
        (1./4.*pi.^2.*(tau+1).^2./tau.^2+pi.*(tau.^2+1).^2./tau.^2./(tau-1).^2.*...
        asin((tau.^2-1)./(tau.^2+1))-2.*pi.*(tau+1)./tau./(tau-1)+...
        (tau.^2+1).^2./tau.^2./(tau-1).^2.*asin((tau.^2-1)./(tau.^2+1)).^2-...
        4./tau.*(tau+1)./(tau-1).*asin((tau.^2-1)./(tau.^2+1))+...
        8./(tau-1).^2.*log((tau.^2+1)./2./tau));

    kbf = Kfb -kfb;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Contribution due to the lift carryover of the foil on the body:
    % (DATCOM, Equation 4.3.2.2-c )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Xhc = calcHcfoilbodyComb(foil, D, Xm)

    b= foil.Span;
    k = D/b;    % diameter to span ratio

    tanLambdaC_4 = tan(foil.Sweep.C4);  % Sweep of c/4
    tanLambdaLe = tan(foil.Sweep.Le); % tan(Lambda_le) Sweep of leading edge

    numBracket = sqrt(1-2*k)*log((1-k)/k+1/k*sqrt(1-2*k))-(1-k)+k*pi/2;
    denBracket = k*(1-k)/sqrt(1-2*k)*log((1-k)/k+1/k*sqrt(1-2*k)) + (1-k)^2/k-pi/2*(1-k);

    % Calculate Xhc/Cre for (\beta AR_e >= 4 )
    BetaAR4 = 1/4 + (b-D)/(2*foil.Cr)*tanLambdaC_4*...
        (-k/(1-k)+ numBracket/denBracket);

    % The proceure below is necessary in order to interpolate the hydrodynamic
    % centre due to lift carryover of the foil on the body if the aspectratio of the
    % exposed foil is less than 4 (Blakelock, page 560)
    %
    if foil.AR < 4 % Interpolation required
        % Calculate Xhc/Cre for (\beta AR_e = 0 )
        gamma = 1/4*(foil.AR*(1+ foil.lambda)* tanLambdaLe);

        if gamma<1 && gamma>=0; BetaAR0 = 0.5*gamma;
        else BetaAR0 = 0.5; end

        % construct polynomal for interpolation
        % Y = aa*(x-4)^2 + bb
        syms x
        aa = (BetaAR0-BetaAR4)/(0-4)^2;
        bb = BetaAR4;
        Y = aa*(x-4)^2+bb;
        Y = inline(char(Y));
        % Interpolated value for (Xhc/Cre)_B(W)
        foilpart = real(Y(foil.AR));
        clear aa bb Y x
    else
        foilpart = BetaAR4;
    end

    Xhc = Xm - (foil.Xo + foil.Cr*foilpart);
```

B.1.13 foilCalc.m

```
function Output = foilCalc(FoilInput,Body)
% Calculate various lift and moment coefficients for different wings
%
```

```
% function call:  foilCalc(Foil,Body)
%   where:
%       Body:  Structure of Body (format given by bodyCalc.m)
%
%       Foil: is a structure:
%           Foil.Plane:     In which plane te foil section is mounted ('XY', 'XZ')
%           Foil.Profile:   String with the name of the foil profile
%           Foil.Cr:        Length of root chord [m]
%           Foil.Ct:        Length of tip chord [m]
%           Foil.Span:      Foil span (tip to tip) [m]
%           Foil.Xo: Leading edge of the root chord of the foil [m]

% Available foils are:
% * NACA-0012
availableFoils = '';
availableFoils = sprintf('%sNACA_0012 \n',availableFoils);


% References:
% Roskam, Jan (1979), "Airplane flight dynamics and automatic flight
%       controls, Part1" [Book]
% Blakelock, John H (1991), "Automatic control of aircrafts and missiles"
%        [Book]
% Datcom, (1978), "USAF Stability and Control Datcom", :Authors: Hoak, D E AND Finck, R D
%
% Hoerner, Sighart F (1985), "Fluid dynamic lift" [Book]
% Pitts, William C & Nielsen, Jack N & Kaattari, George E (1957),"Lift and
%       center of pressure of foil-body-tail combinations at subsonic, transonic,
%       and supersonic speeds". NACA tecnical report 1307
% Whicker, L F & Fehlner, L F (1958),  "Free-stream characteristics of a family of low-aspect-ratio,
%       all movable control surfaces for application to ship design", Navy Department-
%       The David W. Taylor Model Basin

for i = 1 : length(FoilInput)
    Foil = FoilInput(i);
    % ************************************************************************
    % Chech for valid inputs and call the correct procedure
    try
        foilData = feval(sprintf('%s',Foil.Profile));
    catch error(sprintf('Foil profile "%s" is not available\n Available foils are:\n %s',...
            Foil.Profile, availableFoils));
    end
    Lpp = Body.Param.Lpp;   % Nondimensionalize coefficients with Lpp
    Output(i) = currentFoil(Foil, Body, Lpp, foilData);
    % ************************************************************************
end % for

function FoilOut = currentFoil(Foil, Body , Lpp, foilData)
    % Find diameter where foil is mounted
    Foil.D = 2*interp1(Body.shape.x, Body.shape.r, Foil.Xo + Foil.Cr/2);
    Foil.lambda = Foil.Ct/Foil.Cr;   % Foil taper ratio
    Foil.Ymac = 1/3* (1+2*Foil.lambda)/(1+Foil.lambda)*(Foil.Span-Foil.D)/2 + Foil.D/2;
    % Mean hydrodynamic chord (Roskam-1979, page 68)
    Foil.Cmac = 2/3*Foil.Cr*(1+Foil.lambda+Foil.lambda^2)/(1+Foil.lambda); % Chord length at Cp

    % Load the lift curve from functions
    Foil.Curve = foilData.curve;

    % Prepare foilprofile used in plotting and volume/mass calculations
    NDISCw = 100;   % Number of discrete points
    Xtemp = 0:1/NDISCw:1;
    Yw_tempU = Foil.Curve.upper(Xtemp);  % Foil curve is normalized (0,1)
    Yw_tempL = Foil.Curve.lower(Xtemp);  % Foil curve is normalized (0,1)
    Ytip = Foil.Ct * [Yw_tempU;Yw_tempL];
    Xtip = Foil.Xo + Foil.Cr/2 + Foil.Ct*(-0.5 + Xtemp);
    Yroot = Foil.Cr * [Yw_tempU;Yw_tempL];
```

```
    Xroot = Foil.Xo + Foil.Cr*Xtemp;
    Pointed.root.x = Xroot;
    Pointed.root.y = Yroot;
    Pointed.tip.x = Xtip;
    Pointed.tip.y = Ytip;

    Di = 2*interp1(Body.shape.x, Body.shape.r, Xroot);
    z =  0.5 * [Di; Foil.Span* ones(1,length(Xroot))];
    Pointed.root.z = z(1,:);
    Pointed.tip.z = z(2,:);
    Foil.Curve.Discrete = Pointed;
    Foil.Volume = calcFoilVolume(Pointed);

    Foil.cla2d = foilData.c_l;  % Two dimensional lift coeff

    clear foilData Pointed Xroot Yroot Xtip Ytip% Clear temporary variables
    clear Yw_tempU Yw_tempL Xtemp z Di NDISCw

    % Calculate foil areas (There are two foils corresponding to one foil section)
    Foil.Sinfo = ('Exposed foil area (m^2)');
    Foil.S  = 2*calcFoilArea(Foil.Cr, Foil.Ct, Foil.Span/2, Foil.D/2); % Area of exposed foils

    % Calculate wetted surface of the foil
    Foil.SwfInfo = ('Wetted area (m^2)');
    dx = 0.01; x = 0:dx:1;
    Foil.Swf = 2*Foil.S*...
        2*sum(sqrt(Foil.Curve.upper(x).^2 + x.^2)*dx); % See Bø, H Msc Thesis;

    % Calculate foil aspect ratio
    Foil.ARinfo = 'Aspect ratio, exposed foil';
    Foil.AR = (Foil.Span-Foil.D)^2/Foil.S; % Aspect ratio exposed foil

    % Calculate foil sweep angles used in further calculations
    Foil.Sweep.Le = sweepAngle(0, 0, Foil.AR, Foil.lambda);
    Foil.Sweep.C4 = sweepAngle(0.25, 0, Foil.AR, Foil.lambda);
    Foil.Sweep.C4 = sweepAngle(0.25, 0, Foil.AR, Foil.lambda);

% Hydrodynamic center of foil relative to nose
    Foil.hcInfo = 'Hydrodynamic center of foil relative to nose (m)';
    Foil.hc = Foil.Xo + Foil.Cr/4 + (Foil.Ymac-Foil.D/2)*tan(Foil.Sweep.C4);



    % Prepare output structure
    FoilOut = Foil;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%   End of currentFoil() function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Calculate Chord lenght of the extended foil
% Input:
%       Cr: foil root chord [m]
%       Ct: foil tip chord [m]
%       D: Diameter of the body [m]
%       Span: tip to tip foil span [m]
function Cre = calcExtCr(Cr,Ct,D,Span)
    temp = (Cr-Ct)/((Span-D)/2); % D-chord/ D-span
    Cre = Foil.Cr + (D/2)*temp; % Extended foil max chord

% Calculate area of the foil
% Input:
%       Cr: foil root chord [m]
%       Ct: foil tip chord [m]
```

```
%       Yt:
%       Yi:
function Area = calcFoilArea(Cr,Ct,Yt,Yi)
    Area = 1/2*(Cr+Ct)*(Yt-Yi);


function volume = calcFoilVolume(Pointed)
    % Prepare for trapezoidal integration
    dx = 0.5 * ((Pointed.root.x(1:length(Pointed.root.x)-1) +...
                Pointed.root.x(2:length(Pointed.root.x))) - ...
                (Pointed.tip.x(1:length(Pointed.root.x)-1) +...
                Pointed.tip.x(2:length(Pointed.root.x))));
    dz = 0.5 * ((Pointed.root.z(1:length(Pointed.root.z)-1) +...
                Pointed.root.z(2:length(Pointed.root.z))) - ...
                (Pointed.tip.z(1:length(Pointed.root.z)-1) +...
                Pointed.tip.z(2:length(Pointed.root.z))));
    h = sqrt((dx.^2 + dz.^2));  % Mean height of box
    % Used for calculation of base area
    Yir = 0.5* (Pointed.root.y(:,1:length(Pointed.root.y)-1) + ...
        Pointed.root.y(:,2:length(Pointed.root.y)));
    Yit = 0.5* (Pointed.tip.y(:,1:length(Pointed.tip.y)-1) + ...
        Pointed.tip.y(:,2:length(Pointed.tip.y)));
    dXir = Pointed.root.x(2:length(Pointed.root.x)) -...
                Pointed.root.x(1:length(Pointed.root.x)-1);
    dXit = Pointed.tip.x(2:length(Pointed.tip.x)) -...
                Pointed.tip.x(1:length(Pointed.tip.x)-1);

    % There are two foils for each section
    volume = sum(h.*... % height of one box
        (dXit.*(Yit(1,:)-Yit(2,:)) +... % area of tip
        dXir.*(Yir(1,:)-Yir(2,:)))); % area of root




% Description of foil profiles
    % Parameters is found in Abbot (1959). In order to implement new foil
    % series, the parameters "dLdadeg" and "c_d" should be specified. Abbot
    % has a collection of these parameters for a number of foil profiles.
    % The parameters for the NACA_0012 foil profile is found in Abbot page
    % 463 (for the section drag coeff c_d) and figure 57(a) for the Lift
    % curve slop (pr degree)
    % In addition,

function out = NACA_0012()
    dLdadeg = 0.11;    % N/deg (Lift curve slope) Abbot (1959, fig 57(a))
    % Calculate Sectional drag coeff, Abbot (1959, p 463)
    % Approximately:  Y = aa*x^2+bb
    syms x
    bb = 0.006;  % Min Cd for cl = 0
    aa = (0.013-bb)/1.2^2;
    Y = inline(char(aa*x^2+bb)); % Curve fit
    out.c_d = Y(dLdadeg); % Sectional drag coefficient
    out.c_l = dLdadeg*180/pi; % N/rad
    t = 0.12;   % maximum thickness as a fraction of chord
    % inline function for normalised curve, Abbot (1959, p. 113)
    Y = sprintf('%1.5f* (0.2969*sqrt(x)-0.126*x-0.3516*x.^2+0.2843*x.^3-0.1015*x.^4)',t/0.2);
    out.curve.upper = inline(Y);
    out.curve.lower = inline(['-',Y]);

% Sweep angle at n chordwise location
% (LAMBDA = sweepangle at 0,5*chordlength) - See i.e Roskam
 function SAn = sweepAngle(n, LAMBDA, AR, lambda)
    m = 0.5;
    SAn = atan(tan(LAMBDA) - 4*(n-m)/AR*(1-lambda)/(1+lambda));
```

B.1.14 foilDialog.m

```
function varargout = FoilDialog(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FoilDialog_OpeningFcn, ...
                   'gui_OutputFcn',  @FoilDialog_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before FoilDialog is made visible.
function FoilDialog_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to FoilDialog (see VARARGIN)

% Choose default command line output for FoilDialog
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

if strcmp(get(hObject,'Visible'),'off')
    initialize_gui(hObject, handles);
end

% UIWAIT makes FoilDialog wait for user response (see UIRESUME)
uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = FoilDialog_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

% Clear wing dialog
delete(handles.figure1);

% --- Executes during object creation, after setting all properties.
function Cr_EditBox_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function Cr_EditBox_Callback(hObject, eventdata, handles)
    temp = str2double(get(hObject,'String'));% returns contents of Ct as a double

    if isnan(temp) || ~min(size(temp) == [1 1])
        set(hObject, 'string', sprintf('%1.4f',0.09))
    elseif temp < 0
        set(hObject, 'string', sprintf('%1.4f',0.09))
    else
       set(hObject, 'string', sprintf('%1.4f',temp))
    end


% --- Executes during object creation, after setting all properties.
function Ct_EditBox_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function Ct_EditBox_Callback(hObject, eventdata, handles)
    temp = str2double(get(hObject,'String'));% returns contents of Ct as a double
    if isnan(temp) || ~min(size(temp) == [1 1])
        set(hObject, 'string', sprintf('%1.4f',0.06))
    elseif temp < 0
        set(hObject, 'string', sprintf('%1.4f',0.06))
    else
       set(hObject, 'string', sprintf('%1.4f',temp))
    end

% --- Executes during object creation, after setting all properties.
function WingSpan_EditBox_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg; set(hObject,'BackgroundColor','white');
else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function WingSpan_EditBox_Callback(hObject, eventdata, handles)
    temp = str2double(get(hObject,'String'));% returns contents of Ct as a double

    if isnan(temp) || ~min(size(temp) == [1 1])
        set(hObject, 'string', sprintf('%1.4f',0.44))
    elseif temp < 0
        set(hObject, 'string', sprintf('%1.4f',0.44))
    else
       set(hObject, 'string', sprintf('%1.4f',temp))
    end


function initialize_gui(fig_handle, handles)


% --- Executes on button press in WingApply_Button.
function WingApply_Button_Callback(hObject, eventdata, handles)
    Profiles = get(handles.WingProfile_ListBox,'String');
    Profile = Profiles(get(handles.WingProfile_ListBox,'Value')); % Get active profile
    Xo = str2double(get(handles.Ledge_EditBox,'String'));
    Planes = get(handles.Plane_ListBox,'String');
    Plane = Planes(get(handles.Plane_ListBox,'Value')); % Get active plane
    Ct = str2double(get(handles.Ct_EditBox,'String'));
    Cr = str2double(get(handles.Cr_EditBox,'String'));
    Span = str2double(get(handles.WingSpan_EditBox,'String'));
```

```
    Rho = str2double(get(handles.WingDensity_Input,'String'));

    Wing.Profile = char(Profile);
    Wing.Rho = Rho;
    Wing.Xo = Xo;
    Wing.Plane = char(Plane);
    Wing.Ct = Ct;
    Wing.Cr = Cr;
    Wing.Span = Span;

    handles.output = Wing;
    % Update handles structure
    guidata(hObject, handles);
    uiresume(handles.figure1);


% --- Executes on button press in WingCancel_Button.
function WingCancel_Button_Callback(hObject, eventdata, handles)
     handles.output = [];
    % Update handles structure
    guidata(hObject, handles);
    uiresume(handles.figure1);


% --- Executes during object creation, after setting all properties.
function WingProfile_ListBox_CreateFcn(hObject, eventdata, handles)
if ispc; set(hObject,'BackgroundColor','white');
else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in WingProfile_ListBox.
function WingProfile_ListBox_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function Plane_ListBox_CreateFcn(hObject, eventdata, handles)
if ispc ; set(hObject,'BackgroundColor','white');
else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in Plane_ListBox.
function Plane_ListBox_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function Ledge_EditBox_CreateFcn(hObject, eventdata, handles)
if ispc ; set(hObject,'BackgroundColor','white');
else set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function Ledge_EditBox_Callback(hObject, eventdata, handles)
    temp = str2double(get(hObject,'String'));% returns contents of Ct as a double
    if isnan(temp) || ~min(size(temp) == [1 1])
        set(hObject, 'string', sprintf('%1.4f',1.225))
    elseif temp < 0
        set(hObject, 'string', sprintf('%1.4f',1.225))
    else
       set(hObject, 'string', sprintf('%1.4f',temp))
    end

% --- Executes during object creation, after setting all properties.
function WingDensity_Input_CreateFcn(hObject, eventdata, handles)
if ispc; set(hObject,'BackgroundColor','white');
else  set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function WingDensity_Input_Callback(hObject, eventdata, handles)
    temp = str2double(get(hObject,'String'));%

    if isnan(temp) || ~min(size(temp) == [1 1])
        set(hObject, 'string', sprintf('%d',2700))
    elseif temp < 0
        set(hObject, 'string', sprintf('%d',2700))
    else
       set(hObject, 'string', sprintf('%d',temp))
    end
```

B.1.15 massNonDim.m

```
function nonDim = massNonDim(M,L, Rho)
    nonDim = zeros(6,6);    % Initialize matrix
    for i = 1:6
        for j = 1:6
            if  max(i==[1 2 3]) && max(j==[1 2 3])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^3);
            elseif max(i==[1 2 3]) && max(j==[4 5 6])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^4);
            elseif max(i==[4 5 6]) && max(j==[1 2 3])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^4);
            elseif max(i==[4 5 6]) && max(j==[4 5 6])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^5);
            else error('Something is wrong');
            end % if
        end %for
    end % for
```

B.1.16 MrigidBody.m

```
function nonDim = massNonDim(M,L, Rho)
    nonDim = zeros(6,6);    % Initialize matrix
    for i = 1:6
        for j = 1:6
            if  max(i==[1 2 3]) && max(j==[1 2 3])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^3);
            elseif max(i==[1 2 3]) && max(j==[4 5 6])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^4);
            elseif max(i==[4 5 6]) && max(j==[1 2 3])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^4);
            elseif max(i==[4 5 6]) && max(j==[4 5 6])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^5);
            else error('Something is wrong');
            end % if
        end %for
    end % for
```

B.1.17 rudderTerms.m

```
function nonDim = massNonDim(M,L, Rho)
    nonDim = zeros(6,6);    % Initialize matrix
    for i = 1:6
        for j = 1:6
            if  max(i==[1 2 3]) && max(j==[1 2 3])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^3);
            elseif max(i==[1 2 3]) && max(j==[4 5 6])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^4);
            elseif max(i==[4 5 6]) && max(j==[1 2 3])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^4);
            elseif max(i==[4 5 6]) && max(j==[4 5 6])
                nonDim(i,j) = M(i,j)/(0.5*Rho*L^5);
```

```
            else error('Something is wrong');
            end % if
        end %for
    end % for
```

Appendix C

WAMIT- runs

Below, are the input files necessary for estimation of the added-mass coefficients of an AUV, identical to the one considered in Chapter 4, with the aid of WAMIT. to use WAMIT for and output files used The vehicle is identical . The din a sequence of WAMIT runs of an AUV similar to the vehicle planned to undergo towing tank experiments in the *Marine Cybernetics Towing Tank* during the fall of 2004. All files presented in the following are also included in a digital format in Appendix D.

C.1 Input files

In this section, input files used in the WAMIT runs are described in some detail. For additional information, the reader is advised to consult WAMIT-Inc (2003).

C.1.1 Fnames.wam

This file is optional, and is simply a list of the other input file names including their respective extensions.

```
Maia.cfg
Maia.pot
Maia.frc
Maia.gdf
```

C.1.2 Maia.cfg

This is the configuration file used to specify various parameters and options in WAMIT. For details about the various parameters, the reader is advised to consult (WAMIT-Inc 2003, Section 3.7).

```
MAXSCR=1024
ILOWHI=1
IPERIO=1
IALTFRC=2
IALTPOT=2
IRR=0
ISOLVE=1
NUMHDR=1
NOOUT=1 1 1 1 0 1 1 1 1
USERID_PATH=\WAMITv6   (directory for *.exe, *.dll, and userid.wam)
panel_size= 0.02
IPNLBPT=1
```

C.1.3 Maia.pot

This file is the *Potential Control File* used for input of various parameters to the POTEN subprogram in WAMIT. It should be noted that *Alternative Form*

2, IALTPOT=2, is used (WAMIT-Inc 2003, Section 3.2). Important variables are:

HBOT Water depth. (1.5m is the depth of the *Marine Cybernetics Towing Tank*). A number less than or equal zero is interpreted to mean that the water depth is infinite.

NPER Is the number of wave periods to be analyzed. A negative number (-10) is used in Section C.2.2 and C.2.3.

PER Is the wave period(s), T, in seconds. In Section C.2.2 and C.2.3, {0.1 1} and {1 10} are used respectively.

XBODY(3) is the depth of the body fixed reference point (center of buoyancy). Note that the z-axis is positive upward in WAMIT.

```
WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø
 1.5                          HBOT, 1.5M waterdepth
  1     1                     IRAD, IDIFF
 1                            NPER (array PER follows)
 0                            Wave period, in seconds
 1                            NBETA (array BETA follows)
 0                            Wave headings
 1                            NBODY
 Maia.gdf                     MultiSurf model (igdef=2)
 0. 0. -0.75 0.0              XBODY, body fixed coord vs global
 1  1  1  1  1  1             IMODE(1-6)
 0                            NEWMDS
```

C.1.4 Maia.frc

This file is used for input of various parameters to the FORCE subprogram in WAMIT. Because it is possible to specify separately three independent external force matrices including the mass matrix of the body, an external damping matrix, and an external stiffness matrix, *Alternative Form 2* is used. This permits the analysis of bodies which are not freely floating in waves, with arbitrary linear external forces and moments, and also permits the specification of the complete body mass matrix instead of the simpler radii of gyration. In this work, however, only the mass matrix was specified.

```
WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø
 1    0    0    0    0    0  0  0 0     IOPTN(1-9)
 1025.0  RHO, Denisity of sea water
  0.0         0.0          -0.02  Center of gravity
  1         Read massmatrix (next 6 lines)
 47.01      0.0      0.0      0.0       0.0      0.0
   0.00    47.01     0.0      0.0       0.0      0.0
   0.00      0.0    47.01     0.0       0.0      0.0
   0.00      0.0      0.0    0.2327       0.0      0.0
   0.00      0.0      0.0    0.0       8.4416      0.0
   0.00      0.0      0.0    0.0       0.0      8.4416
 0      Do not read any external damping matrix
 0      Do not read any external stifness matrix
 0                                          NBETAH
 0                                          NFIELD
```

C.1.5 Maia.gdf

This is the file that describe the geometry of the body to be analyzed. In this work, the Relation Geometry Kernel (RGKernel) of the CAD program MultiSurf is used to specify the geometry. Important variables are:

ULEN is the dimensional length characterizing the body dimension. This parameter corresponds to the quantity used to nondimensionalize the quantities output from WAMIT.

ISY The body is symmetric about the $x-z$ plane (Only half of the body need to be modelled in MultiSurf)

NPATCH is the number of surfaces described in the *ObjecList* of the MultiSurf model.

WettedSurfs Name of the ObjectList included in the MultiSurf model Maia.ms2.

```
WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø
 1.64  9.80665  ULEN, GRAV
  0  1  ISX, ISY
  4   2  NPATCH, IGDEF
  3  NLINES (lines to follow)
Maia.ms2
WettedSurfs
0  0  1 FAST,DivMult, inward normals
```

C.1.6 Maia.ms2

```
MultiSurf 1.24
Units: m kg
Symmetry:  y
Extents:  -1.637 -200.000 -0.100 1.637 200.000 0.100
View:  -30.00 60.00 0
Places: 3
Layers: 30000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
DivMult: 1
Attribute: __Nudge DOUBLE  SYSTEM(0.10000000000000001)
Attribute: __layer11 STRING  SYSTEM("CenterLine")
Attribute: __layer2 STRING  SYSTEM("Nose-Points")
Attribute: __layer3 STRING  SYSTEM("Tail-Points")
Attribute: __layer4 STRING  SYSTEM("AUV-Curve")
Attribute: __layer5 STRING  SYSTEM("Foil-Points")
Attribute: __layer6 STRING  SYSTEM("TipPoints")
Attribute: __layer7 STRING  SYSTEM("TopCurves-Foil")
Attribute: __layer9 STRING  SYSTEM("CurvesFoilLower")
Attribute: __layer20 STRING  SYSTEM("FoilMagnet")
Attribute: __layer1 STRING  SYSTEM("Surfaces")
BeginModel;
AbsPoint NoseVertex 14 1 L:2 / 0.781 0.000 0.000 ;
RelPoint p0 14 1 L:2 / NoseVertex -0.002 0.027 0.000 ;
RelPoint p1 14 1 L:2 / NoseVertex -0.004 0.034 0.000 ;
RelPoint p2 14 1 L:2 / NoseVertex -0.008 0.043 0.000 ;
RelPoint p3 14 1 L:2 / NoseVertex -0.012 0.049 0.000 ;
RelPoint p4 14 1 L:2 / NoseVertex -0.018 0.056 0.000 ;
RelPoint p5 14 1 L:2 / NoseVertex -0.025 0.062 0.000 ;
RelPoint p6 14 1 L:2 / NoseVertex -0.035 0.068 0.000 ;
RelPoint p7 14 1 L:2 / NoseVertex -0.050 0.076 0.000 ;
RelPoint p8 14 1 L:2 / NoseVertex -0.070 0.083 0.000 ;
RelPoint p9 14 1 L:2 / NoseVertex -0.100 0.091 0.000 ;
RelPoint p10 14 1 L:2 / NoseVertex -0.140 0.097 0.000 ;
```

```
RelPoint p11 14 1 L:2 / NoseVertex -0.200 0.100 0.000 ;
RelPoint TailLe 14 1 L:3 / p11 -1.200 0.000 0.000 ;
RelPoint p13 14 1 L:3 / TailLe -0.030 -0.003 0.000 ;
RelPoint p14 14 1 L:3 / TailLe -0.050 -0.007 0.000 ;
RelPoint p15 14 1 L:3 / TailLe -0.080 -0.018 0.000 ;
RelPoint p16 14 1 L:3 / TailLe -0.100 -0.026 0.000 ;
RelPoint p17 14 1 L:3 / TailLe -0.130 -0.041 0.000 ;
RelPoint p18 14 1 L:3 / TailLe -0.150 -0.052 0.000 ;
RelPoint p19 14 1 L:3 / TailLe -0.180 -0.069 0.000 ;
RelPoint p20 14 1 L:3 / TailLe -0.200 -0.080 0.000 ;
RelPoint p21 14 1 L:3 / TailLe -0.220 -0.090 0.000 ;
RelPoint p22 14 1 L:3 / TailLe -0.230 -0.095 0.000 ;
RelPoint TailEnd 14 1 L:3 / TailLe -0.240 -0.100 0.000 ;
BCurve AUVbodyCurve 11 1 8x4 L:4 / * 2
    { NoseVertex p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 TailLe p13 p14
      p15 p16 p17 p18 p19 p20 p21 p22 TailEnd } ;
Line l0 6 1 1x1 L:11 / * NoseVertex TailEnd ;
RevSurf BodySurf 2 3 16x8 16x8 0 L:1 / * AUVbodyCurve l0 -90.0000
     90.0000 ;
RelPoint FoilLe 14 1 L:5 / NoseVertex -1.255 0.000 0.000 ;
RelPoint FoilLe0 14 1 L:5 / FoilLe -0.005 0.000 0.003 ;
RelPoint FoilLe1 14 1 L:5 / FoilLe -0.010 0.000 0.004 ;
RelPoint FoilLe2 14 1 L:5 / FoilLe -0.015 0.000 0.005 ;
RelPoint FoilLe3 14 1 L:5 / FoilLe -0.025 0.000 0.005 ;
RelPoint FoilLe4 14 1 L:5 / FoilLe -0.035 0.000 0.005 ;
RelPoint FoilLe5 14 1 L:5 / FoilLe -0.045 0.000 0.005 ;
RelPoint FoilLe6 14 1 L:5 / FoilLe -0.055 0.000 0.004 ;
RelPoint FoilLe7 14 1 L:5 / FoilLe -0.065 0.000 0.003 ;
RelPoint FoilLe8 14 1 L:5 / FoilLe -0.075 0.000 0.002 ;
RelPoint FoilLe9 14 1 L:5 / FoilLe -0.085 0.000 0.001 ;
RelPoint FoilLe10 14 1 L:5 / FoilLe -0.090 0.000 0.000 ;
BCurve AUVbodyCurve0 11 1 8x4 L:7 / * 2
    { FoilLe FoilLe0 FoilLe1 FoilLe2 FoilLe3 FoilLe4 FoilLe5 FoilLe6
      FoilLe7 FoilLe8 FoilLe9 FoilLe10 } ;
RelPoint FoilMagnet 14 1 L:20 / FoilLe -0.045 200.000 0.000 ;
ProjSnake2 n0 13 1 8x4 L:7 / * AUVbodyCurve0 BodySurf FoilMagnet 0.0000
     ;
RelPoint FoilLetTip 14 1 L:6 / FoilLe -0.015 0.220 0.000 ;
RelPoint FoilLe13 14 1 L:6 / FoilLetTip -0.005 0.000 0.003 ;
RelPoint FoilLe14 14 1 L:6 / FoilLetTip -0.008 0.000 0.003 ;
RelPoint FoilLe15 14 1 L:6 / FoilLetTip -0.015 0.000 0.004 ;
RelPoint FoilLe16 14 1 L:6 / FoilLetTip -0.020 0.000 0.004 ;
RelPoint FoilLe17 14 1 L:6 / FoilLetTip -0.029 0.000 0.003 ;
RelPoint FoilLe18 14 1 L:6 / FoilLetTip -0.030 0.000 0.003 ;
RelPoint FoilLe19 14 1 L:6 / FoilLetTip -0.035 0.000 0.003 ;
RelPoint FoilLe20 14 1 L:6 / FoilLetTip -0.045 0.000 0.002 ;
RelPoint FoilLe21 14 1 L:6 / FoilLetTip -0.055 0.000 0.001 ;
RelPoint FoilLe22 14 1 L:6 / FoilLetTip -0.060 0.000 0.000 ;
BCurve AUVbodyCurve1 11 1 8x4 L:7 / * 2
    { FoilLetTip FoilLe13 FoilLe14 FoilLe15 FoilLe16 FoilLe17 FoilLe18
      FoilLe19 FoilLe20 FoilLe21 FoilLe22 } ;
RuledSurf FoilTop 2 3 32x1 1x1 0 L:1 / * n0 AUVbodyCurve1 ;
Line MirrLineFoil 6 1 1x1 L:2 / * FoilLetTip FoilLe22 ;
MirrCurve AUVbodyCurve2 11 1 8x4 L:9 / * AUVbodyCurve0 l0 ;
ProjSnake n1 13 1 8x4 L:9 / * AUVbodyCurve2 BodySurf FoilMagnet ;
MirrCurve AUVbodyCurve3 11 1 8x4 L:9 / * AUVbodyCurve1 MirrLineFoil ;
RuledSurf FoilBottom 2 3 32x1 1x1 0 L:1 / * AUVbodyCurve3 n1 ;
RuledSurf FoilTip 2 3 32x1 1x1 0 L:1 / * AUVbodyCurve1 AUVbodyCurve3 ;
ObjectList WettedSurfs /
    { BodySurf FoilTop FoilBottom FoilTip } ;
EndModel;
```

C.2 Output

It is important to observe that all added-mass, A(I,J), and damping coefficients, B(I,J), from WAMIT are nondimensionalized according to

$$\bar{A}_{ij} = \frac{A_{ij}}{\rho L^k} \quad , \quad \bar{B}_{ij} = \frac{B_{ij}}{\rho L^k \omega} \, ,$$

where L is the length scale defined by the input parameter ULEN in the GDF file (Section C.1.5), ω is the wave frequency (rad/s); and $k = 3$ for i, $j = 1$, 2, 3, $k = 4$ for $i = 1$, 2, 3, $j = 4$, 5, 6 or $i = 4$, 5, 6, $j = 1$, 2, 3 and $k = 5$ for i, $j = 4$, 5, 6.

C.2.1 Wave period, zero

```
 -------------------------------------------------------------------------

                       WAMIT  Version 6.1

     Copyright (c) 1999-2002 WAMIT Incorporated
     Copyright (c) 1998 Massachusetts Institute of Technology

 -------------------------------------------------------------------------

     The WAMIT software performs computations of wave interactions with
     floating or submerged vessels.  WAMIT is a registered trademark of
     WAMIT Incorporated.  This copy of the WAMIT software is licensed to

            CESOS
            Norwegian University of Science and Technology
            Trondheim, Norway

 For educational use only at the above location. Release date 20 JUN 2002

 -------------------------------------------------------------------------


 High-order panel method (ILOWHI=1)          Panel_Size =        0.02000

 Input from Geometric Data File:         Maia.gdf
 WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø

 Input from Potential Control File:      maia.pot
 WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø


 POTEN run date and starting time:        14-Jul-2004  --  16:13:57
    Period       Time           RAD      DIFF  (max iterations)
    0.0000    16:14:48          -1

 Gravity:     9.80665               Length scale:        1.64000
 Water depth:         1.50000       Water density:     1025.00000
 Panel quadrature indices:       IQUAD = 0   ILOG = 0   IDIAG = 0
 Source formulation index:                   ISOR =     0
 Irregular frequency index:                   IRR =     0
 Diffraction/scattering formulation index: ISCATT =     0
 Number of blocks used in linear system:   ISOLVE =     1
 Number of unknowns in linear system:        NEQN =   408

 BODY PARAMETERS:

 NPATCH:     4              IGDEF:    2              Symmetry:   Y=0
```

```
           Patch   NU  NV    KU  KV   IQUI IQVI
              1    86   1     3   3     4    4
              2     5   7     3   3     4    4
              3     5   7     3   3     4    4
              4     4   1     3   3     4    4

XBODY =    0.0000 YBODY =    0.0000 ZBODY =   -0.7500 PHIBODY =   0.0
Volumes (VOLX,VOLY,VOLZ):          0.455842E-01 0.456761E-01 0.456465E-01
Center of Buoyancy (Xb,Yb,Zb):    -0.000934     0.000000     0.000001
Hydrostatic and gravitational restoring coefficients:
C(3,3),C(3,4),C(3,5): 0.134415E-06  0.00000     0.960473E-10
C(4,4),C(4,5),C(4,6):              0.126805E-03  0.00000     0.589192E-05
       C(5,5),C(5,6):                           0.126814E-03  0.00000
Center of Gravity  (Xg,Yg,Zg):     0.000000     0.000000    -0.020000

Global body and external mass matrix:
  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  0.0000E+00  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  0.0000E+00  0.0000E+00  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00
  0.0000E+00  0.0000E+00  0.0000E+00  2.3270E-01  0.0000E+00  0.0000E+00
  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  8.4416E+00  0.0000E+00
  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  8.4416E+00


 -------------------------------------------------------------------------
                            Output from  WAMIT
 -------------------------------------------------------------------------
 FORCE run date and starting time:                14-Jul-2004 -- 16:14:48
 -------------------------------------------------------------------------
 I/O Filenames:  maia.frc            maia.p2f            maia.out
 WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø


 *************************************************************************

 Wave period = zero                                  Wavenumber = infinite
 -------------------------------------------------------------------------


    ADDED-MASS COEFFICIENTS
     I     J         A(I,J)

     1     1   4.709911E-04
     1     3   6.261392E-07
     1     5   4.489813E-06
     2     2   9.234812E-03
     2     4  -1.849953E-08
     2     6   1.010350E-04
     3     1   4.119678E-07
     3     3   9.720550E-03
     3     5   1.158742E-04
     4     2  -1.832205E-08
     4     4   1.616972E-06
     4     6   1.370622E-09
     5     1   4.443521E-06
     5     3   1.158718E-04
     5     5   5.642324E-04
     6     2   1.073890E-04
     6     4   1.092597E-09
     6     6   5.119582E-04
```

C.2.2 Wave period, [0.1:0.1:1]

```
 -------------------------------------------------------------------------
```

```
                    WAMIT  Version 6.1

    Copyright (c) 1999-2002 WAMIT Incorporated
    Copyright (c) 1998 Massachusetts Institute of Technology

------------------------------------------------------------------------

    The WAMIT software performs computations of wave interactions with
    floating or submerged vessels.  WAMIT is a registered trademark of
    WAMIT Incorporated.  This copy of the WAMIT software is licensed to

           CESOS
           Norwegian University of Science and Technology
           Trondheim, Norway

For educational use only at the above location. Release date 20 JUN 2002

------------------------------------------------------------------------


High-order panel method (ILOWHI=1)          Panel_Size =        0.02000

Input from Geometric Data File:         Maia.gdf
WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø

Input from Potential Control File:      maia.pot
WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø


POTEN run date and starting time:        14-Jul-2004  --  15:58:49
   Period       Time           RAD      DIFF  (max iterations)
   0.1000    15:59:48          -1      -1
   0.2000    16:00:03          -1      -1
   0.3000    16:00:18          -1      -1
   0.4000    16:00:33          -1      -1
   0.5000    16:00:48          -1      -1
   0.6000    16:01:03          -1      -1
   0.7000    16:01:17          -1      -1
   0.8000    16:01:31          -1      -1
   0.9000    16:01:45          -1      -1
   1.0000    16:01:59          -1      -1

Gravity:     9.80665               Length scale:        1.64000
Water depth:         1.50000       Water density:     1025.00000
Panel quadrature indices:       IQUAD = 0   ILOG = 0   IDIAG = 0
Source formulation index:                   ISOR =     0
Irregular frequency index:                   IRR =     0
Diffraction/scattering formulation index: ISCATT =     0
Number of blocks used in linear system:   ISOLVE =     1
Number of unknowns in linear system:        NEQN =   408

BODY PARAMETERS:

NPATCH:     4               IGDEF:    2              Symmetry:   Y=0
           Patch   NU  NV    KU  KV   IQUI IQVI
              1    86   1     3   3     4    4
              2     5   7     3   3     4    4
              3     5   7     3   3     4    4
              4     4   1     3   3     4    4

XBODY =    0.0000 YBODY =    0.0000 ZBODY =   -0.7500 PHIBODY =   0.0
Volumes (VOLX,VOLY,VOLZ):          0.455842E-01 0.456761E-01 0.456465E-01
Center of Buoyancy (Xb,Yb,Zb):    -0.000934     0.000000     0.000001
Hydrostatic and gravitational restoring coefficients:
C(3,3),C(3,4),C(3,5): 0.134415E-06  0.00000     0.960473E-10
```

```
C(4,4),C(4,5),C(4,6):              0.126805E-03  0.00000     0.589192E-05
       C(5,5),C(5,6):                           0.126814E-03  0.00000
Center of Gravity  (Xg,Yg,Zg):     0.000000     0.000000    -0.020000

Global body and external mass matrix:
  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  0.0000E+00  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  0.0000E+00  0.0000E+00  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00
  0.0000E+00  0.0000E+00  0.0000E+00  2.3270E-01  0.0000E+00  0.0000E+00
  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  8.4416E+00  0.0000E+00
  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  8.4416E+00


-------------------------------------------------------------------------
                          Output from  WAMIT
-------------------------------------------------------------------------
FORCE run date and starting time:                14-Jul-2004 -- 16:01:59
-------------------------------------------------------------------------
I/O Filenames:  maia.frc            maia.p2f            maia.out
WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø


*************************************************************************

Wave period (sec) =  1.000000E-01        Wavenumber (kL) =  6.602112E+02
-------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

    1     1   4.709280E-04  -2.884305E-16
    1     3   6.292692E-07   9.272727E-16
    1     5   4.516100E-06   3.284177E-16
    2     2   9.234471E-03  -5.413982E-15
    2     4  -1.874278E-08   9.651816E-18
    2     6   1.010406E-04  -4.343106E-15
    3     1   4.124289E-07   8.431499E-19
    3     3   9.720317E-03   6.410310E-17
    3     5   1.158753E-04   5.368299E-18
    4     2  -1.835170E-08  -4.239865E-20
    4     4   1.616968E-06   1.329308E-23
    4     6   1.362281E-09   5.727324E-22
    5     1   4.467154E-06   3.621793E-20
    5     3   1.158626E-04   8.627186E-18
    5     5   5.642078E-04  -9.511445E-20
    6     2   1.073873E-04  -2.162051E-15
    6     4   1.129767E-09  -7.318184E-18
    6     6   5.119530E-04  -4.609956E-15


*************************************************************************

Wave period (sec) =  2.000000E-01        Wavenumber (kL) =  1.650528E+02
-------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

    1     1   4.707403E-04   3.010478E-16
    1     3   6.370091E-07  -2.398987E-16
    1     5   4.580054E-06  -1.727839E-17
    2     2   9.233481E-03  -4.402962E-15
    2     4  -1.900244E-08  -1.855292E-18
    2     6   1.010328E-04   3.596654E-16
```

```
    3     1   4.249681E-07  -7.882122E-19
    3     3   9.718772E-03  -1.677063E-17
    3     5   1.158583E-04  -7.850746E-19
    4     2  -1.867566E-08   5.996225E-21
    4     4   1.616974E-06  -2.336240E-24
    4     6   1.390679E-09  -2.256026E-22
    5     1   4.529833E-06   1.640072E-19
    5     3   1.158560E-04  -1.648423E-19
    5     5   5.641785E-04  -2.306480E-19
    6     2   1.073723E-04   6.196715E-16
    6     4   1.145865E-09   8.817852E-19
    6     6   5.119388E-04   5.293354E-16

 ************************************************************************

 Wave period (sec) =  3.000000E-01        Wavenumber (kL) =  7.335680E+01
 ------------------------------------------------------------------------


    ADDED-MASS AND DAMPING COEFFICIENTS
     I     J         A(I,J)         B(I,J)

    1     1   4.704155E-04   3.968403E-17
    1     3   6.502449E-07  -1.522776E-17
    1     5   4.694490E-06   8.907250E-18
    2     2   9.231649E-03   7.314102E-15
    2     4  -1.950073E-08   4.126075E-18
    2     6   1.010199E-04   2.965704E-16
    3     1   4.384823E-07  -3.368762E-21
    3     3   9.716596E-03   4.401478E-19
    3     5   1.158453E-04   5.052470E-20
    4     2  -1.920011E-08  -1.533789E-20
    4     4   1.616972E-06   4.648030E-24
    4     6   1.435363E-09  -2.925411E-23
    5     1   4.645203E-06  -1.559198E-20
    5     3   1.158403E-04   3.476715E-20
    5     5   5.641074E-04  -1.118790E-20
    6     2   1.073471E-04   2.902937E-16
    6     4   1.218093E-09   2.867318E-18
    6     6   5.119142E-04   3.336990E-16

 ************************************************************************

 Wave period (sec) =  4.000000E-01        Wavenumber (kL) =  4.126320E+01
 ------------------------------------------------------------------------


    ADDED-MASS AND DAMPING COEFFICIENTS
     I     J         A(I,J)         B(I,J)

    1     1   4.699350E-04   5.330578E-18
    1     3   6.711857E-07  -2.392821E-17
    1     5   4.871003E-06  -3.908434E-17
    2     2   9.228945E-03  -1.646622E-16
    2     4  -2.035658E-08  -1.201923E-17
    2     6   1.009961E-04  -9.306943E-17
    3     1   4.533836E-07   1.255771E-19
    3     3   9.713172E-03  -5.387274E-18
    3     5   1.158173E-04   1.218350E-19
    4     2  -2.007100E-08   3.795368E-23
    4     4   1.616971E-06   1.757684E-23
    4     6   1.547516E-09   1.904955E-22
    5     1   4.819567E-06  -1.099640E-19
    5     3   1.158119E-04  -9.752527E-21
```

```
    5     5   5.639940E-04  -1.095228E-19
    6     2   1.073110E-04  -1.003939E-16
    6     4   1.309383E-09   3.826573E-18
    6     6   5.118750E-04   1.003605E-16


 ************************************************************************

 Wave period (sec) =  5.000000E-01        Wavenumber (kL) =  2.640845E+01
 ------------------------------------------------------------------------


    ADDED-MASS AND DAMPING COEFFICIENTS
     I     J         A(I,J)         B(I,J)

    1     1   4.692617E-04   1.728706E-13
    1     3   7.008917E-07   8.684791E-15
    1     5   5.123762E-06  -7.353080E-14
    2     2   9.225048E-03   7.745260E-12
    2     4  -2.158270E-08   4.032976E-15
    2     6   1.009694E-04   1.124007E-13
    3     1   4.743512E-07  -4.605088E-14
    3     3   9.708235E-03   1.276624E-11
    3     5   1.157815E-04  -4.457222E-14
    4     2  -2.135713E-08   6.611194E-15
    4     4   1.616971E-06   6.362624E-17
    4     6   1.722366E-09  -2.251760E-15
    5     1   5.070368E-06  -8.231588E-14
    5     3   1.157777E-04  -4.345519E-14
    5     5   5.638280E-04   6.966589E-13
    6     2   1.072607E-04   1.137486E-13
    6     4   1.463255E-09  -1.442382E-15
    6     6   5.118171E-04   4.067164E-13


 ************************************************************************

 Wave period (sec) =  6.000000E-01        Wavenumber (kL) =  1.833920E+01
 ------------------------------------------------------------------------


    ADDED-MASS AND DAMPING COEFFICIENTS
     I     J         A(I,J)         B(I,J)

    1     1   4.683414E-04   1.922345E-10
    1     3   7.427205E-07   6.171495E-13
    1     5   5.485003E-06  -7.338564E-11
    2     2   9.219452E-03   6.540605E-09
    2     4  -2.340956E-08   5.565262E-12
    2     6   1.009217E-04   1.087894E-10
    3     1   5.150540E-07  -6.967725E-12
    3     3   9.701132E-03   8.651817E-09
    3     5   1.157242E-04   2.829496E-11
    4     2  -2.332017E-08   7.171796E-12
    4     4   1.616968E-06   4.274381E-14
    4     6   2.012098E-09  -2.348399E-12
    5     1   5.426095E-06  -7.888380E-11
    5     3   1.157210E-04   2.293231E-11
    5     5   5.635784E-04   4.838597E-10
    6     2   1.071848E-04   1.135365E-10
    6     4   1.689328E-09  -1.831134E-12
    6     6   5.117276E-04   3.227459E-10


 ************************************************************************
```

```
 Wave period (sec) =  7.000000E-01        Wavenumber (kL) =  1.347370E+01
 -------------------------------------------------------------------------


    ADDED-MASS AND DAMPING COEFFICIENTS
     I     J         A(I,J)         B(I,J)

     1     1   4.670599E-04   8.401687E-09
     1     3   8.044736E-07  -8.532202E-10
     1     5   6.021754E-06  -7.111739E-09
     2     2   9.210885E-03   3.077778E-07
     2     4  -2.639404E-08   2.537382E-10
     2     6   1.008494E-04   4.791892E-09
     3     1   5.624594E-07  -1.282172E-09
     3     3   9.690152E-03   3.816500E-07
     3     5   1.156360E-04   2.832275E-09
     4     2  -2.665645E-08   3.033706E-10
     4     4   1.616967E-06   1.173752E-12
     4     6   2.542375E-09  -8.672846E-11
     5     1   5.959282E-06  -7.346487E-09
     5     3   1.156336E-04   2.730516E-09
     5     5   5.631492E-04   2.286871E-08
     6     2   1.070633E-04   5.148261E-09
     6     4   2.124200E-09  -7.195874E-11
     6     6   5.115672E-04   1.422859E-08


 *************************************************************************

 Wave period (sec) =  8.000000E-01        Wavenumber (kL) =  1.031580E+01
 -------------------------------------------------------------------------


    ADDED-MASS AND DAMPING COEFFICIENTS
     I     J         A(I,J)         B(I,J)

     1     1   4.652366E-04   1.469365E-07
     1     3   8.959914E-07  -9.355689E-09
     1     5   6.853449E-06  -5.896412E-08
     2     2   9.197249E-03   3.199916E-06
     2     4  -3.132488E-08   2.134328E-09
     2     6   1.007292E-04   4.640644E-08
     3     1   6.327522E-07  -1.133796E-08
     3     3   9.672735E-03   3.956657E-06
     3     5   1.154941E-04   3.278586E-08
     4     2  -3.188530E-08   2.456862E-09
     4     4   1.616961E-06   6.730696E-12
     4     6   3.397474E-09  -7.141944E-10
     5     1   6.785970E-06  -6.042765E-08
     5     3   1.154884E-04   3.258518E-08
     5     5   5.623851E-04   2.228980E-07
     6     2   1.068713E-04   5.001867E-08
     6     4   2.908153E-09  -6.180888E-10
     6     6   5.112893E-04   1.392429E-07


 *************************************************************************

 Wave period (sec) =  9.000000E-01        Wavenumber (kL) =  8.150761E+00
 -------------------------------------------------------------------------


    ADDED-MASS AND DAMPING COEFFICIENTS
     I     J         A(I,J)         B(I,J)

     1     1   4.626475E-04   4.865831E-07
```

```
   1     3   1.049478E-06  -3.996153E-08
   1     5   8.350898E-06  -4.085883E-07
   2     2   9.179269E-03   1.419897E-05
   2     4  -3.696028E-08   7.860383E-09
   2     6   1.005960E-04   1.861782E-07
   3     1   7.766721E-07  -4.375577E-08
   3     3   9.648960E-03   1.684336E-05
   3     5   1.152914E-04   1.607833E-07
   4     2  -3.788047E-08   8.796000E-09
   4     4   1.616961E-06   1.660910E-11
   4     6   3.751743E-09  -2.340049E-09
   5     1   8.267736E-06  -4.114377E-07
   5     3   1.152887E-04   1.606536E-07
   5     5   5.613333E-04   1.189342E-06
   6     2   1.066280E-04   2.104491E-07
   6     4   3.240769E-09  -2.072853E-09
   6     6   5.110257E-04   5.478361E-07

*********************************************************************

Wave period (sec) =  1.000000E+00        Wavenumber (kL) =  6.602188E+00
---------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

    1     1   4.585422E-04   1.855594E-06
    1     3   1.269610E-06  -1.975475E-07
    1     5   1.030547E-05  -2.167814E-06
    2     2   9.163548E-03   3.742164E-05
    2     4  -3.968802E-08   1.822265E-08
    2     6   1.005415E-04   4.208446E-07
    3     1   9.593930E-07  -1.957890E-07
    3     3   9.624320E-03   4.661277E-05
    3     5   1.151283E-04   4.462850E-07
    4     2  -4.051353E-08   1.992896E-08
    4     4   1.616968E-06   2.519584E-11
    4     6   2.753779E-09  -4.039066E-09
    5     1   1.019623E-05  -2.158890E-06
    5     3   1.151240E-04   4.466573E-07
    5     5   5.610074E-04   3.216449E-06
    6     2   1.064270E-04   5.346740E-07
    6     4   2.289975E-09  -3.590595E-09
    6     6   5.110796E-04   1.159657E-06
```

C.2.3 Wave period, [1:1:10]

```
---------------------------------------------------------------------

                     WAMIT  Version 6.1

    Copyright (c) 1999-2002 WAMIT Incorporated
    Copyright (c) 1998 Massachusetts Institute of Technology

---------------------------------------------------------------------

    The WAMIT software performs computations of wave interactions with
    floating or submerged vessels.  WAMIT is a registered trademark of
    WAMIT Incorporated.  This copy of the WAMIT software is licensed to

           CESOS
           Norwegian University of Science and Technology
```

```
           Trondheim, Norway

 For educational use only at the above location. Release date 20 JUN 2002

 ------------------------------------------------------------------------


 High-order panel method (ILOWHI=1)          Panel_Size =        0.02000

 Input from Geometric Data File:         Maia.gdf
 WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø

 Input from Potential Control File:      maia.pot
 WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø


 POTEN run date and starting time:        14-Jul-2004  --  15:58:49
    Period       Time           RAD      DIFF  (max iterations)
    0.1000    15:59:48          -1       -1
    0.2000    16:00:03          -1       -1
    0.3000    16:00:18          -1       -1
    0.4000    16:00:33          -1       -1
    0.5000    16:00:48          -1       -1
    0.6000    16:01:03          -1       -1
    0.7000    16:01:17          -1       -1
    0.8000    16:01:31          -1       -1
    0.9000    16:01:45          -1       -1
    1.0000    16:01:59          -1       -1

 Gravity:     9.80665               Length scale:        1.64000
 Water depth:         1.50000       Water density:     1025.00000
 Panel quadrature indices:       IQUAD = 0   ILOG = 0   IDIAG = 0
 Source formulation index:                   ISOR =     0
 Irregular frequency index:                   IRR =     0
 Diffraction/scattering formulation index: ISCATT =     0
 Number of blocks used in linear system:   ISOLVE =     1
 Number of unknowns in linear system:        NEQN =   408

 BODY PARAMETERS:

 NPATCH:     4              IGDEF:    2              Symmetry:   Y=0
            Patch   NU  NV    KU  KV   IQUI IQVI
               1    86   1     3   3     4    4
               2     5   7     3   3     4    4
               3     5   7     3   3     4    4
               4     4   1     3   3     4    4

 XBODY =    0.0000 YBODY =    0.0000 ZBODY =   -0.7500 PHIBODY =   0.0
 Volumes (VOLX,VOLY,VOLZ):          0.455842E-01 0.456761E-01 0.456465E-01
 Center of Buoyancy (Xb,Yb,Zb):    -0.000934     0.000000     0.000001
 Hydrostatic and gravitational restoring coefficients:
 C(3,3),C(3,4),C(3,5): 0.134415E-06  0.00000     0.960473E-10
 C(4,4),C(4,5),C(4,6):               0.126805E-03  0.00000     0.589192E-05
        C(5,5),C(5,6):                            0.126814E-03  0.00000
 Center of Gravity  (Xg,Yg,Zg):     0.000000     0.000000    -0.020000

 Global body and external mass matrix:
   4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   0.0000E+00  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   0.0000E+00  0.0000E+00  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00
   0.0000E+00  0.0000E+00  0.0000E+00  2.3270E-01  0.0000E+00  0.0000E+00
   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  8.4416E+00  0.0000E+00
   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  8.4416E+00


 ------------------------------------------------------------------------
```

```
                        Output from  WAMIT
-------------------------------------------------------------------------
FORCE run date and starting time:               14-Jul-2004 -- 16:01:59
-------------------------------------------------------------------------
I/O Filenames:  maia.frc            maia.p2f            maia.out
WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø


*************************************************************************

Wave period (sec) =  1.000000E-01        Wavenumber (kL) =  6.602112E+02
-------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

    1     1   4.709280E-04  -2.884305E-16
    1     3   6.292692E-07   9.272727E-16
    1     5   4.516100E-06   3.284177E-16
    2     2   9.234471E-03  -5.413982E-15
    2     4  -1.874278E-08   9.651816E-18
    2     6   1.010406E-04  -4.343106E-15
    3     1   4.124289E-07   8.431499E-19
    3     3   9.720317E-03   6.410310E-17
    3     5   1.158753E-04   5.368299E-18
    4     2  -1.835170E-08  -4.239865E-20
    4     4   1.616968E-06   1.329308E-23
    4     6   1.362281E-09   5.727324E-22
    5     1   4.467154E-06   3.621793E-20
    5     3   1.158626E-04   8.627186E-18
    5     5   5.642078E-04  -9.511445E-20
    6     2   1.073873E-04  -2.162051E-15
    6     4   1.129767E-09  -7.318184E-18
    6     6   5.119530E-04  -4.609956E-15


*************************************************************************

Wave period (sec) =  2.000000E-01        Wavenumber (kL) =  1.650528E+02
-------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

    1     1   4.707403E-04   3.010478E-16
    1     3   6.370091E-07  -2.398987E-16
    1     5   4.580054E-06  -1.727839E-17
    2     2   9.233481E-03  -4.402962E-15
    2     4  -1.900244E-08  -1.855292E-18
    2     6   1.010328E-04   3.596654E-16
    3     1   4.249681E-07  -7.882122E-19
    3     3   9.718772E-03  -1.677063E-17
    3     5   1.158583E-04  -7.850746E-19
    4     2  -1.867566E-08   5.996225E-21
    4     4   1.616974E-06  -2.336240E-24
    4     6   1.390679E-09  -2.256026E-22
    5     1   4.529833E-06   1.640072E-19
    5     3   1.158560E-04  -1.648423E-19
    5     5   5.641785E-04  -2.306480E-19
    6     2   1.073723E-04   6.196715E-16
    6     4   1.145865E-09   8.817852E-19
    6     6   5.119388E-04   5.293354E-16
```

```
************************************************************************

Wave period (sec) =  3.000000E-01        Wavenumber (kL) =  7.335680E+01
------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

    1     1   4.704155E-04   3.968403E-17
    1     3   6.502449E-07  -1.522776E-17
    1     5   4.694490E-06   8.907250E-18
    2     2   9.231649E-03   7.314102E-15
    2     4  -1.950073E-08   4.126075E-18
    2     6   1.010199E-04   2.965704E-16
    3     1   4.384823E-07  -3.368762E-21
    3     3   9.716596E-03   4.401478E-19
    3     5   1.158453E-04   5.052470E-20
    4     2  -1.920011E-08  -1.533789E-20
    4     4   1.616972E-06   4.648030E-24
    4     6   1.435363E-09  -2.925411E-23
    5     1   4.645203E-06  -1.559198E-20
    5     3   1.158403E-04   3.476715E-20
    5     5   5.641074E-04  -1.118790E-20
    6     2   1.073471E-04   2.902937E-16
    6     4   1.218093E-09   2.867318E-18
    6     6   5.119142E-04   3.336990E-16


************************************************************************

Wave period (sec) =  4.000000E-01        Wavenumber (kL) =  4.126320E+01
------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

    1     1   4.699350E-04   5.330578E-18
    1     3   6.711857E-07  -2.392821E-17
    1     5   4.871003E-06  -3.908434E-17
    2     2   9.228945E-03  -1.646622E-16
    2     4  -2.035658E-08  -1.201923E-17
    2     6   1.009961E-04  -9.306943E-17
    3     1   4.533836E-07   1.255771E-19
    3     3   9.713172E-03  -5.387274E-18
    3     5   1.158173E-04   1.218350E-19
    4     2  -2.007100E-08   3.795368E-23
    4     4   1.616971E-06   1.757684E-23
    4     6   1.547516E-09   1.904955E-22
    5     1   4.819567E-06  -1.099640E-19
    5     3   1.158119E-04  -9.752527E-21
    5     5   5.639940E-04  -1.095228E-19
    6     2   1.073110E-04  -1.003939E-16
    6     4   1.309383E-09   3.826573E-18
    6     6   5.118750E-04   1.003605E-16


************************************************************************

Wave period (sec) =  5.000000E-01        Wavenumber (kL) =  2.640845E+01
------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)
```

```
   1     1   4.692617E-04   1.728706E-13
   1     3   7.008917E-07   8.684791E-15
   1     5   5.123762E-06  -7.353080E-14
   2     2   9.225048E-03   7.745260E-12
   2     4  -2.158270E-08   4.032976E-15
   2     6   1.009694E-04   1.124007E-13
   3     1   4.743512E-07  -4.605088E-14
   3     3   9.708235E-03   1.276624E-11
   3     5   1.157815E-04  -4.457222E-14
   4     2  -2.135713E-08   6.611194E-15
   4     4   1.616971E-06   6.362624E-17
   4     6   1.722366E-09  -2.251760E-15
   5     1   5.070368E-06  -8.231588E-14
   5     3   1.157777E-04  -4.345519E-14
   5     5   5.638280E-04   6.966589E-13
   6     2   1.072607E-04   1.137486E-13
   6     4   1.463255E-09  -1.442382E-15
   6     6   5.118171E-04   4.067164E-13

*****************************************************************************

Wave period (sec) =  6.000000E-01        Wavenumber (kL) =  1.833920E+01
-----------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

   1     1   4.683414E-04   1.922345E-10
   1     3   7.427205E-07   6.171495E-13
   1     5   5.485003E-06  -7.338564E-11
   2     2   9.219452E-03   6.540605E-09
   2     4  -2.340956E-08   5.565262E-12
   2     6   1.009217E-04   1.087894E-10
   3     1   5.150540E-07  -6.967725E-12
   3     3   9.701132E-03   8.651817E-09
   3     5   1.157242E-04   2.829496E-11
   4     2  -2.332017E-08   7.171796E-12
   4     4   1.616968E-06   4.274381E-14
   4     6   2.012098E-09  -2.348399E-12
   5     1   5.426095E-06  -7.888380E-11
   5     3   1.157210E-04   2.293231E-11
   5     5   5.635784E-04   4.838597E-10
   6     2   1.071848E-04   1.135365E-10
   6     4   1.689328E-09  -1.831134E-12
   6     6   5.117276E-04   3.227459E-10

*****************************************************************************

Wave period (sec) =  7.000000E-01        Wavenumber (kL) =  1.347370E+01
-----------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

   1     1   4.670599E-04   8.401687E-09
   1     3   8.044736E-07  -8.532202E-10
   1     5   6.021754E-06  -7.111739E-09
   2     2   9.210885E-03   3.077778E-07
   2     4  -2.639404E-08   2.537382E-10
   2     6   1.008494E-04   4.791892E-09
   3     1   5.624594E-07  -1.282172E-09
```

```
    3     3   9.690152E-03   3.816500E-07
    3     5   1.156360E-04   2.832275E-09
    4     2  -2.665645E-08   3.033706E-10
    4     4   1.616967E-06   1.173752E-12
    4     6   2.542375E-09  -8.672846E-11
    5     1   5.959282E-06  -7.346487E-09
    5     3   1.156336E-04   2.730516E-09
    5     5   5.631492E-04   2.286871E-08
    6     2   1.070633E-04   5.148261E-09
    6     4   2.124200E-09  -7.195874E-11
    6     6   5.115672E-04   1.422859E-08

 ************************************************************************

 Wave period (sec) =  8.000000E-01        Wavenumber (kL) =  1.031580E+01
 ------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

    1     1   4.652366E-04   1.469365E-07
    1     3   8.959914E-07  -9.355689E-09
    1     5   6.853449E-06  -5.896412E-08
    2     2   9.197249E-03   3.199916E-06
    2     4  -3.132488E-08   2.134328E-09
    2     6   1.007292E-04   4.640644E-08
    3     1   6.327522E-07  -1.133796E-08
    3     3   9.672735E-03   3.956657E-06
    3     5   1.154941E-04   3.278586E-08
    4     2  -3.188530E-08   2.456862E-09
    4     4   1.616961E-06   6.730696E-12
    4     6   3.397474E-09  -7.141944E-10
    5     1   6.785970E-06  -6.042765E-08
    5     3   1.154884E-04   3.258518E-08
    5     5   5.623851E-04   2.228980E-07
    6     2   1.068713E-04   5.001867E-08
    6     4   2.908153E-09  -6.180888E-10
    6     6   5.112893E-04   1.392429E-07

 ************************************************************************

 Wave period (sec) =  9.000000E-01        Wavenumber (kL) =  8.150761E+00
 ------------------------------------------------------------------------


   ADDED-MASS AND DAMPING COEFFICIENTS
    I     J         A(I,J)         B(I,J)

    1     1   4.626475E-04   4.865831E-07
    1     3   1.049478E-06  -3.996153E-08
    1     5   8.350898E-06  -4.085883E-07
    2     2   9.179269E-03   1.419897E-05
    2     4  -3.696028E-08   7.860383E-09
    2     6   1.005960E-04   1.861782E-07
    3     1   7.766721E-07  -4.375577E-08
    3     3   9.648960E-03   1.684336E-05
    3     5   1.152914E-04   1.607833E-07
    4     2  -3.788047E-08   8.796000E-09
    4     4   1.616961E-06   1.660910E-11
    4     6   3.751743E-09  -2.340049E-09
    5     1   8.267736E-06  -4.114377E-07
    5     3   1.152887E-04   1.606536E-07
    5     5   5.613333E-04   1.189342E-06
```

```
    6     2   1.066280E-04   2.104491E-07
    6     4   3.240769E-09  -2.072853E-09
    6     6   5.110257E-04   5.478361E-07


 ************************************************************************

 Wave period (sec) =  1.000000E+00        Wavenumber (kL) =  6.602188E+00
 ------------------------------------------------------------------------


    ADDED-MASS AND DAMPING COEFFICIENTS
     I     J         A(I,J)         B(I,J)

     1     1   4.585422E-04   1.855594E-06
     1     3   1.269610E-06  -1.975475E-07
     1     5   1.030547E-05  -2.167814E-06
     2     2   9.163548E-03   3.742164E-05
     2     4  -3.968802E-08   1.822265E-08
     2     6   1.005415E-04   4.208446E-07
     3     1   9.593930E-07  -1.957890E-07
     3     3   9.624320E-03   4.661277E-05
     3     5   1.151283E-04   4.462850E-07
     4     2  -4.051353E-08   1.992896E-08
     4     4   1.616968E-06   2.519584E-11
     4     6   2.753779E-09  -4.039066E-09
     5     1   1.019623E-05  -2.158890E-06
     5     3   1.151240E-04   4.466573E-07
     5     5   5.610074E-04   3.216449E-06
     6     2   1.064270E-04   5.346740E-07
     6     4   2.289975E-09  -3.590595E-09
     6     6   5.110796E-04   1.159657E-06
```

C.2.4 Infinite depth

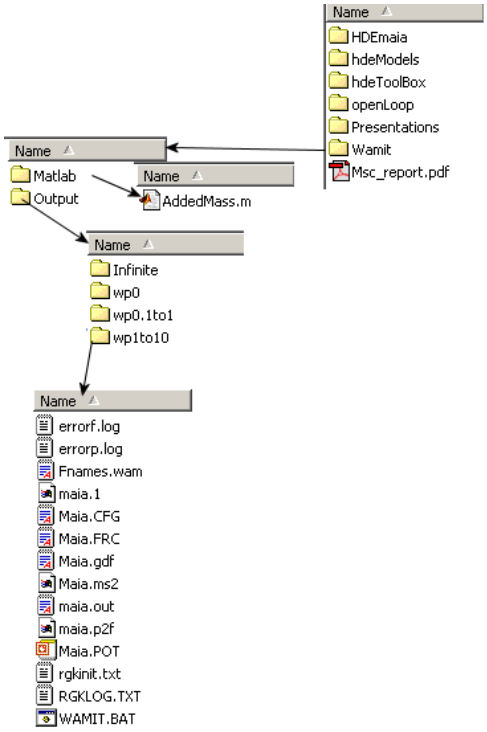
```
 ------------------------------------------------------------------------

                      WAMIT  Version 6.1

     Copyright (c) 1999-2002 WAMIT Incorporated
     Copyright (c) 1998 Massachusetts Institute of Technology

 ------------------------------------------------------------------------

     The WAMIT software performs computations of wave interactions with
     floating or submerged vessels.  WAMIT is a registered trademark of
     WAMIT Incorporated.  This copy of the WAMIT software is licensed to

            CESOS
            Norwegian University of Science and Technology
            Trondheim, Norway

 For educational use only at the above location. Release date 20 JUN 2002

 ------------------------------------------------------------------------


 High-order panel method (ILOWHI=1)          Panel_Size =        0.00600

 Input from Geometric Data File:         Maia.gdf
 WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø

 Input from Potential Control File:      maia.pot
 WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø
```

```
POTEN run date and starting time:        28-Jul-2004  --  00:55:13
   Period       Time           RAD      DIFF  (max iterations)
   0.0000    01:10:53          -1

Gravity:     9.80665               Length scale:        1.64000
Water depth:        infinite       Water density:     1025.00000
Panel quadrature indices:       IQUAD = 0   ILOG = 0   IDIAG = 0
Source formulation index:                   ISOR =     0
Irregular frequency index:                   IRR =     0
Diffraction/scattering formulation index: ISCATT =     0
Number of blocks used in linear system:   ISOLVE =     1
Number of unknowns in linear system:        NEQN =  1731

BODY PARAMETERS:

NPATCH:     4              IGDEF:    2              Symmetry:   Y=0
           Patch   NU  NV    KU  KV   IQUI IQVI
              1   286   1     3   3     4    4
              2    16  21     3   3     4    4
              3    16  21     3   3     4    4
              4    11   1     3   3     4    4

XBODY =    0.0000 YBODY =    0.0000 ZBODY = -100.0000 PHIBODY =   0.0
Volumes (VOLX,VOLY,VOLZ):          0.456068E-01 0.456823E-01 0.456897E-01
Center of Buoyancy (Xb,Yb,Zb):    -0.001051     0.000000     0.025819
Hydrostatic and gravitational restoring coefficients:
C(3,3),C(3,4),C(3,5): 0.139246E-06  0.00000     0.240133E-08
C(4,4),C(4,5),C(4,6):               0.289848E-03  0.00000     0.663792E-05
       C(5,5),C(5,6):                             0.289857E-03  0.00000
Center of Gravity  (Xg,Yg,Zg):     0.000000     0.000000    -0.020000

Global body and external mass matrix:
  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  0.0000E+00  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  0.0000E+00  0.0000E+00  4.7010E+01  0.0000E+00  0.0000E+00  0.0000E+00
  0.0000E+00  0.0000E+00  0.0000E+00  2.3270E-01  0.0000E+00  0.0000E+00
  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  8.4416E+00  0.0000E+00
  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  8.4416E+00


------------------------------------------------------------------------
                          Output from  WAMIT
------------------------------------------------------------------------
FORCE run date and starting time:               28-Jul-2004 -- 01:10:53
------------------------------------------------------------------------
I/O Filenames:  maia.frc            maia.p2f            maia.out
WAMIT- calc of indian AUV (Maia); Msc Thesis by Håvard Bø


************************************************************************

Wave period = zero                                 Wavenumber = infinite
------------------------------------------------------------------------


   ADDED-MASS COEFFICIENTS
    I     J         A(I,J)

    1     1   4.733859E-04
    1     3   1.324246E-09
    1     5   5.533647E-10
    2     2   9.199041E-03
    2     4  -2.700559E-09
    2     6   1.131656E-04
```

```
3     1  -5.011303E-10
3     3   9.700041E-03
3     5   1.150010E-04
4     2  -3.317342E-10
4     4   1.596680E-06
4     6  -6.026137E-11
5     1  -2.737651E-09
5     3   1.149854E-04
5     5   5.635803E-04
6     2   1.190352E-04
6     4  -4.095232E-11
6     6   5.080176E-04
```

Appendix D

CD Contents



The contents of the CD is shown in the figure above. Except for the directory WAMIT, the contents can easily be explained in words.

- HDEmaia: An hde model of MAYA used to obtain Figure 5.9
- hdeModels: hde models of MAYA, REMUS and the vehicle of Ridley et al. (2003) used for comparison in Chapter 5
- hdeToolBox: Contains the necessary files for installing the MATLAB toolbox developed it this thesis
- openLoop: Contains the Simulink diagram and transfer functions used in Chapter 4
- Presentations: Powerpoint presentation used at ISR

The directory WAMIT, contains the in and output files used for added mass estimation. The directory Matlab, simply contains a script used to dimensionalize the output from WAMIT. The structure for the other directories are as shown in the figure above. A detailed description of the various files is provided in Appendix C.