Defence Research and Development Canada
Recherche et développement pour la défense Canada

DRDC | RDDC
technologysciencetechnologie

Deeply Submerged Hydrodynamic, Control, Propulsion, and Dynamic Change Models for Underwater Vehicle Simulation

George D. Watt
DRDC – Atlantic Research Centre

Defence Research and Development Canada
Reference Document
DRDC-RDDC-2019-D160
November 2019

Canada

IMPORTANT INFORMATIVE STATEMENTS

This document was reviewed for Controlled Goods by Defence Research and Development Canada (DRDC) using the Schedule to the *Defence Production Act*.

Endorsement statement: This publication has been published by the Editorial Office of Defence Research and Development Canada, an agency of the Department of National Defence of Canada. Inquiries can be sent to: Publications.DRDC-RDDC@drdc-rddc.gc.ca.

Abstract

In 2011, a DRDC Technology Investment Fund project was initiated to develop a concept for reliably docking unmanned underwater vehicles with a slowly moving submerged submarine. A contractor, Dynamic Systems Analysis Ltd. (DSA), began developing a simulation to evaluate the concept using their own and additional algorithms from DRDC and other contractors. This report documents the deeply submerged hydrodynamic, control, propulsion, and dynamic change models DRDC provided DSA. The report has evolved over a multi-year period to incorporate feedback from DSA.

Résumé

En 2011, on a lancé un projet du Fonds d'investissement technologique de RDDC en vue de concevoir un moyen d'amarrer avec fiabilité des véhicules sous-marins sans équipage (VSSE) à un sous-marin immergé se déplaant lentement. Un entrepreneur, à savoir Dynamic Systems Analysis Ltée (DSA), a entrepris l'élaboration d'une simulation visant à évaluer ce moyen à l'aide de ses propres algorithmes ainsi que d'algorithmes additionnels provenant de RDDC et d'autres entrepreneurs. Le présent rapport documente les modèles relatifs à l'hydrodynamique en immersion profonde, au contrôle, à la propulsion et aux changements dynamiques que RDDC a fournis à DSA. À noter qu'il a été modifié au cours des ans afin d'y intégrer la rétroaction de DSA.

Contents

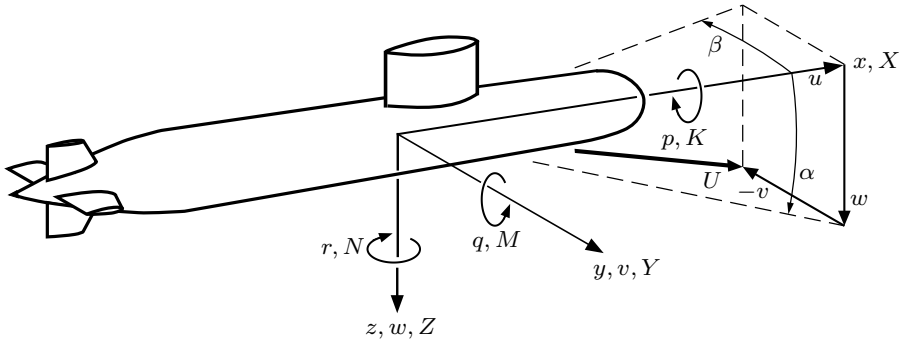
Abstract i
Contents ii
Nomenclature iii

1 Introduction 1
2 Equations of Motion of a Submerged Rigid Body 1
3 Components of the Equations of Motion 5
Free Surface Effects 8
4 The Deeply Submerged Hydrodynamic Coefficient Model 9
5 Heading, Roll, Depth, and Plane Reversal Control 14
Plane Reversal 15
Autopilots 17
6 Propulsion Loads 19
7 Simulation Initialization 22
Submarine Trim 24
UUV Trim 25
8 Dynamic Change Model 28
Limiting the Rate 30
Limiting the Amplitude 33
Soft Limits 34
Hard Limits 35
An Analytic Dynamic Change Algorithm 36
A Numerical Rate Limited Dynamic Change Algorithm 36
9 Concluding Remarks 38

Acknowledgement 38
References 39

Appendices
A BB3 Data 40
B UUV Data 52
C Wageningen B4-70 Family Propeller Loads 61
D Dynamic Change Module 68

Nomenclature



$B = \rho g V$	Vehicle buoyancy.
CB, CG	Centers of buoyancy and gravity.
d	Maximum hull diameter.
D	Propeller diameter.
E_D	Depth error (a distance), the signal to feed to the depth autopilot.
$\overline{BG} = z_G - z_B$	Height of the CB above the CG.
g	Gravitational constant.
$I_{ij}, \mathbf{I}$	Moments and products of inertia in body axes: $\mathbf{I} = \begin{pmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{pmatrix}$
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	Body axis unit vectors.
$J = V_A/(nD)$	Propeller advance ratio.
J_s, J_{sn}	Advance ratio at the actual and nominal self-propulsion point.
$K, M, N, \mathbf{M}$	$\mathbf{M} = K\mathbf{i} + M\mathbf{j} + N\mathbf{k}$ are the body axis moments on the vehicle.
K', M', N'	Body axis moments nondimensionalized by $\rho U^2 \ell^3/2$.
$K_T = T/\left(\rho n^2 D^4\right)$	Propeller thrust coefficient.
$K_Q = Q/\left(\rho n^2 D^5\right)$	Propeller torque coefficient.
ℓ	Vehicle length.
$m, m' = m/\left(\frac{1}{2}\rho \ell^3\right)$	Vehicle mass and dimensionless mass.
$m'_t = m/(\rho V) = W/B$	Vehicle trimmed dimensionless mass.
n	Propeller revolutions per second, rps.
$p, q, r, \mathbf{\Omega}$	$\mathbf{\Omega} = p\mathbf{i} + q\mathbf{j} + r\mathbf{k}$ are body axis angular velocities.
t	Time.
t_D	Thrust deduction fraction.
T, Q	Propeller thrust and torque.
$u, v, w, \mathbf{U}$	$\mathbf{U} = u\mathbf{i} + v\mathbf{j} + w\mathbf{k}$ are the body fixed velocities.
$U = \sqrt{u^2 + v^2 + w^2}$	Overall speed of body.

V	Volume within the hydrodynamic envelope.
V_A	Propeller speed of advance.
w_T	Effective (Taylor) wake fraction for the propeller.
$W = mg$	Vehicle weight; everything (including floodwater) within the hydrodynamic envelope.
$x, y, z, \mathbf{R}$	$\mathbf{R} = x\,\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ are coordinates in body fixed axes.
x_0, y_0, z_0	Inertial (earth-fixed) coordinates locating the body axis origin.
$x_B, y_B, z_B, \mathbf{R}_B$	$\mathbf{R}_B = x_B\,\mathbf{i} + y_B\mathbf{j} + z_B\mathbf{k}$ locates the CB in body axes.
$x_G, y_G, z_G, \mathbf{R}_G$	$\mathbf{R}_G = x_G\,\mathbf{i} + y_G\mathbf{j} + z_G\mathbf{k}$ locates the CG in body axes.
x_P, y_P, z_P	$x_P\,\mathbf{i} + y_P\mathbf{j} + z_P\mathbf{k}$ locates the propeller axis at the blades in body axes.
$x_\delta, x'_\delta = x_\delta/\ell$	Axial location in body axes of a control surface center of pressure.
$X, Y, Z, \mathbf{F}$	$\mathbf{F} = X\,\mathbf{i} + Y\,\mathbf{j} + Z\,\mathbf{k}$ are the body axis forces on the vehicle.
X', Y', Z'	Body axis forces nondimensionalized by $\rho U^2 \ell^2/2$.
$\alpha = \tan^{-1}(w/u)$	Angle of attack.
$\beta = \tan^{-1}(-v/u)$	Angle of drift.
δ_i	Deflection of control surface i; sense from the right hand rule with thumb pointing radially outwards.
$\delta_b, \delta_r, \delta_s, \delta_\phi$	Foreplane, rudder, sternplane, and roll control virtual deflections; sense from the right hand rule using body axes.
δ_D	Virtual depth control deflection (an angle).
η_o	Propeller open-water efficiency.
ν	Kinematic viscosity.
ν	Crossflow velocity magnitude: $\sqrt{v^2 + w^2}$.
ρ	Sea water density.
ψ, θ, ϕ	Yaw, pitch, and roll Euler angles giving body axes orientation relative to the inertial axes.
ψ_P, θ_P	Yaw and pitch angles of the propeller axis relative to the body fixed x axis.

1 Introduction

In 2011, a Defence Research and Development Canada (DRDC) Technology Investment Fund (TIF) project was initiated to develop a concept for reliably docking unmanned underwater vehicles (UUVs) with a slowly moving submerged submarine in littoral waters in the presence of environmental disturbance [1,2]. The concept was to be proven through simulation which required that hydrodynamic and system models for a submarine and UUV be provided to the contractor developing the simulator. This report describes many of those models for a deeply submerged vehicle. Early drafts of this report were used by preliminary versions of the simulator [3]. The simulator continues to evolve based on the final version of this report and other ongoing work. From a hydrodynamics point of view, the contractor's challenge is to merge the deeply submerged hydrodynamic models described herein with free surface effects under waves as modelled by ShipMo3D [4].

The equations of motion used to model the six degrees-of-freedom (6 DOF) motion of the vehicles, and the hydrodynamic force models embedded in those equations, are first described. Then vehicle propulsion, control, and dynamic change models are described. Finally, the generic submarine and UUV hydrodynamic model parameters used in the docking simulation are provided in Appendices A and B.

The dynamic change model described in Section 8 is an extensive revision of that presented by Watt [5]. It is the basis for a new dynamic change algorithm listed in Appendix D.

2 Equations of Motion of a Submerged Rigid Body

The equations of motion are extended versions of Euler's equations for rigid body dynamics. By formulating the equations in body axes, the moments of inertia for the body do not vary in time which greatly simplifies using the equations. In addition, fluid forces acting on the body and control sensors fixed to the body are usually easier to model in body axes. Fossen [6] presents a complete derivation of the extended form of the equations we require (but misnames some body-axes acceleration terms as Coriolis accelerations). The equations are also presented by Gertler and Hagen [7], Feldman [8], and Watt [9].

The equations describe the evolution in time of 12 states that completely define the 6 DOF motion of a rigid body:

$$\mathbf{y} = u, v, w, p, q, r, x_0, y_0, z_0, \phi, \theta, \psi \tag{1}$$

The first 6 of these are translational velocities u, v, w along, and rotational velocities p, q, r about, the vehicle x, y, z body fixed axes. The Euler angles ϕ, θ, ψ describe the roll, pitch, and yaw orientations of the vehicle relative to the x_0, y_0, z_0 inertial axes.

The order in which the Euler angle rotations are applied, when transforming between inertial and body axes, is an integral part of the angle definitions since finite rotations of the angles in different orders give different final orientations. If the body axes are initially aligned with the inertial axes (however much the origins are displaced), an arbitrary reorientation relative to the inertial axes is obtained by

1) yawing about the z axis through an angle ψ,
2) pitching about the relocated y axis through an angle θ, and
3) rolling about the twice relocated x axis through an angle ϕ.

All rotations are in the positive sense as defined by the Right Hand Rule. Thus, the matrix $\mathbf{A}$ transforming a vector description using x_0, y_0, z_0 axes to one using body axes is:

$$\mathbf{A} \equiv \mathbf{A}_\phi \mathbf{A}_\theta \mathbf{A}_\psi \equiv \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\sin\psi\cos\phi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{pmatrix}. \quad (2)$$

Therefore, if $\mathbf{v}$ and $\mathbf{v}_0$ are the vectors specifying the velocity of the vehicle using body axes and inertial axes coordinate systems, respectively:

$$\mathbf{v} \equiv \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \quad \mathbf{v}_0 \equiv \begin{pmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \end{pmatrix} \quad (3)$$

then:

$$\mathbf{v} = \mathbf{A}\mathbf{v}_0 \quad \text{and} \quad \mathbf{v}_0 = \mathbf{A}^{-1}\,\mathbf{v}\,. \quad (4)$$

Since $\mathbf{A}$ is an orthogonal matrix, $\mathbf{A}^{-1} = \mathbf{A}^{\mathrm{T}}$ (where $\mathbf{A}^{\mathrm{T}}$ is the transpose of $\mathbf{A}$).

Because of their definitions, $\dot{\psi}, \dot{\theta}, \dot{\phi}$ are not independent orthogonal components of a vector as are p, q, r. The relationship between the two sets of angular velocities is obtained by summing the contributions of the separately transformed Euler angular velocities to a body axes representation:

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \mathbf{A}_\phi \mathbf{A}_\theta \mathbf{A}_\psi \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} + \mathbf{A}_\phi \mathbf{A}_\theta \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \mathbf{A}_\phi \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} \quad (5)$$

giving:

$$\begin{aligned} p &= -\sin\theta\,\dot{\psi} + \dot{\phi} \\ q &= \sin\phi\cos\theta\,\dot{\psi} + \cos\phi\,\dot{\theta} \\ r &= \cos\phi\cos\theta\,\dot{\psi} - \sin\phi\,\dot{\theta}\,. \end{aligned} \quad (6)$$

These equations result in the following 6 nonlinear ordinary differential equations (ODEs) describing the kinematic relationships between body fixed and inertial velocities:

$$\dot{x}_0 = u\cos\theta\cos\psi + v(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + w(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi) \quad (7a)$$

$$\dot{y}_0 = u\cos\theta\sin\psi + v(\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi) + w(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \quad (7b)$$

$$\dot{z}_0 = -u\sin\theta + v\cos\theta\sin\phi + w\cos\theta\cos\phi \quad (7c)$$

$$\dot{\phi} = p + (r\cos\phi + q\sin\phi)\tan\theta \quad (7d)$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \quad (7e)$$

$$\dot{\psi} = \frac{r\cos\phi + q\sin\phi}{\cos\theta} \quad (7f)$$

Feldman [8] refers to these as 'auxiliary equations' to the dynamics equations of motion derived next. Note that (7d) and (7f) make the equations singular at pitch angles of ± 90 degrees. This singularity can be avoided through the use of quaternions [6] but generally this is not necessary for submarines and other streamlined underwater vehicles which try to avoid large pitch angles.

The overall force $\mathbf{F}$ and moment $\mathbf{M}$ on a vehicle with fixed mass ($dm/dt \equiv 0$) and mass distribution ($d\mathbf{I}/dt \equiv 0$) translating and rotating as a rigid body are:

$$\begin{aligned} \mathbf{F} &= \frac{d}{dt}(\text{momentum}) = m\left[\frac{d\mathbf{U}}{dt} + \boldsymbol{\Omega} \times \mathbf{U} + \frac{d\boldsymbol{\Omega}}{dt} \times \mathbf{R}_G + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times \mathbf{R}_G)\right] \\ \mathbf{M} &= \frac{d}{dt}(\text{angular momentum}) = \mathbf{I} \cdot \frac{d\boldsymbol{\Omega}}{dt} + \boldsymbol{\Omega} \times (\mathbf{I} \cdot \boldsymbol{\Omega}) + m\mathbf{R}_G \times \left(\frac{d\mathbf{U}}{dt} + \boldsymbol{\Omega} \times \mathbf{U}\right) \end{aligned} \tag{8}$$

These equations reduce to the much simpler Euler equations if the body axes are chosen to be the principle axes (so the cross products of inertia are zero) with $\mathbf{R}_G = 0$ (so half the terms in (8) are zero). Although possible to do, this is not practical for real world applications where the CG location is rarely known before a vehicle is put to use, and often changes throughout its life or even throughout a mission if ballast tanks are blown and flooded. Since the equations are often used during the design and evaluation phases, before the vehicle is even built, a convenient location for the body axes origin needs to be chosen early. Common locations are on the hull centerline opposite the hull CB or at the hull midpoint. Putting the origin close to the boat CB results in consistently small $\mathbf{R}_G$ values so that $\mathbf{R}_G$ can usually be neglected during linearized analyses of the equations of motion, such as for stability analyses. Also, keeping the origin close to the CB, a physical hydrodynamic center, probably makes comparisons between hydrodynamic coefficients from different boats more meaningful.

The mass m and moment of inertia $\mathbf{I}$ terms in (8) refer to all the mass enclosed by the hydrodynamic envelope, including any water in ballast tanks or free flooding spaces, because this mass moves with the vehicle. The space that ballast/floodwater occupies is usually subdivided so that any relative movement between vehicle and water is localized enough to not impact stability. Therefore, all such floodwater is assumed to move with the vehicle as a single rigid body.

The equations neglect the dm/dt and $d\mathbf{I}/dt$ contributions to momentum change when a vehicle blows or floods its ballast tanks. The justification for this is that the overall mass change is small (less than 10%) and takes place slowly. By allowing mass and the moments of inertia to vary with time in (8), a quasi-steady model of the change can still be implemented. This would not be appropriate for vehicles like rockets which depend on rapid mass change for propulsion.

Switching the right and left hand sides of (8) and separating them into components gives the six equations of motion for a submerged vehicle (9), shown on the next page. These are first order ODEs in the body axes velocities but, implicitly, are second order in position and angular variables. Integrating these equations once gives u, v, w, p, q, r from (1). Integrating them twice gives body axis position and angular coordinates such as $\int u\,dt$ and $\int p\,dt$ which are not much use. What are useful are the inertial position coordinates and Euler angels $x_0, y_0, z_0, \phi, \theta, \psi$ from (1). These are easily obtained by integrating (7) in parallel with (9).

Axial Force

$$m\Big[\dot{u} - vr + wq - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(pr + \dot{q})\Big] = X \tag{9a}$$

Lateral Force

$$m\Big[\dot{v} - wp + ur - y_G(r^2 + p^2) + z_G(qr - \dot{p}) + x_G(qp + \dot{r})\Big] = Y \tag{9b}$$

Normal Force

$$m\Big[\dot{w} - uq + vp - z_G(p^2 + q^2) + x_G(rp - \dot{q}) + y_G(rq + \dot{p})\Big] = Z \tag{9c}$$

Rolling Moment

$$\begin{aligned} I_x\dot{p} + (I_z - I_y)qr - (\dot{r} + pq)I_{xz} + (r^2 - q^2)I_{yz} + (pr - \dot{q})I_{xy} \\ + m\Big[y_G(\dot{w} - uq + vp) - z_G(\dot{v} - wp + ur)\Big] = K \end{aligned} \tag{9d}$$

Pitching Moment

$$\begin{aligned} I_y\dot{q} + (I_x - I_z)rp - (\dot{p} + qr)I_{xy} + (p^2 - r^2)I_{xz} + (qp - \dot{r})I_{yz} \\ + m\Big[z_G(\dot{u} - vr + wq) - x_G(\dot{w} - uq + vp)\Big] = M \end{aligned} \tag{9e}$$

Yawing Moment

$$\begin{aligned} I_z\dot{r} + (I_y - I_x)pq - (\dot{q} + rp)I_{yz} + (q^2 - p^2)I_{xy} + (rq - \dot{p})I_{xz} \\ + m\Big[x_G(\dot{v} - wp + ur) - y_G(\dot{u} - vr + wq)\Big] = N \end{aligned} \tag{9f}$$

Equations (7) and (9) define 12 nonlinear, coupled, first order ordinary differential equations in the 12 states (1). The simplest way to integrate the equations numerically is to put them in the form:

$$\dot{\mathbf{y}} = f(t, \mathbf{y}) \tag{10}$$

Equations (7) are already in this form. To put (9) into the desired form, the terms:

$$m\left[\boldsymbol{\Omega} \times \mathbf{U} + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times \mathbf{R}_G)\right] \qquad \text{and} \qquad \left[\boldsymbol{\Omega} \times (\mathbf{I} \cdot \boldsymbol{\Omega}) + m\mathbf{R}_G \times (\boldsymbol{\Omega} \times \mathbf{U})\right] \tag{11}$$

from (8) need to be transferred to the right hand sides of (9), and any acceleration dependent hydrodynamic force terms (such as the added mass terms) transferred from the right hand to the left hand sides of (9). This is done in the next section where, in addition, those terms requiring special treatment when merging potential flow derived wave forces with the deeply submerged hydrodynamic forces on the vehicle are identified.

3 Components of the Equations of Motion

It is convenient to rewrite (9) in matrix form and begin identifying the various contributions to the forces and moments acting on the vehicle:

$$\mathcal{M} \cdot \frac{d\mathcal{V}}{dt} + \mathcal{W} \cdot \mathcal{M} \cdot \mathcal{V} = \mathcal{F}_H + \mathcal{F}_G + \mathcal{F}_P \tag{12}$$

where:

$$\mathcal{M} = \begin{pmatrix} m & 0 & 0 & 0 & mz_G & -my_G \\ 0 & m & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m & my_G & -mx_G & 0 \\ 0 & -mz_G & my_G & I_x & -I_{xy} & -I_{xz} \\ mz_G & 0 & -mx_G & -I_{xy} & I_y & -I_{yz} \\ -my_G & mx_G & 0 & -I_{xz} & -I_{yz} & I_z \end{pmatrix}, \quad \frac{d\mathcal{V}}{dt} = \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix}, \tag{13}$$

$$\mathcal{W} = \begin{pmatrix} 0 & -r & q & 0 & 0 & 0 \\ r & 0 & -p & 0 & 0 & 0 \\ -q & p & 0 & 0 & 0 & 0 \\ 0 & -w & v & 0 & -r & q \\ w & 0 & -u & r & 0 & -p \\ -v & u & 0 & -q & p & 0 \end{pmatrix}, \quad \mathcal{V} = \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix}, \quad \mathcal{F}_H = \begin{pmatrix} X_H \\ Y_H \\ Z_H \\ K_H \\ M_H \\ N_H \end{pmatrix}$$

and:

$$\mathcal{W} \cdot \mathcal{M} \cdot \mathcal{V} = \begin{pmatrix} m\big[wq - vr - (q^2 + r^2)x_G + pqy_G + prz_G\big] \\ m\big[ur - wp - (r^2 + p^2)y_G + qrz_G + qpx_G\big] \\ m\big[vp - uq - (p^2 + q^2)z_G + rpx_G + rqy_G\big] \\ m\big[(vp - uq)y_G + (wp - ur)z_G\big] + qr(I_z - I_y) - pqI_{xz} + (r^2 - q^2)I_{yz} + prI_{xy} \\ m\big[(wq - vr)z_G + (uq - vp)x_G\big] + rp(I_x - I_z) - qrI_{xy} + (p^2 - r^2)I_{xz} + qpI_{yz} \\ m\big[(ur - wp)x_G + (vr - wq)y_G\big] + pq(I_y - I_x) - rpI_{yz} + (q^2 - p^2)I_{xy} + rqI_{xz} \end{pmatrix}. \tag{14}$$

$\mathcal{F}_H$ represents the hydrodynamic forces. The gravitational forces $\mathcal{F}_G$ account for vehicle weight W and buoyancy B and are derived by Imlay [11] and Fossen [6]:

$$\mathcal{F}_G = \begin{pmatrix} X_G \\ Y_G \\ Z_G \\ K_G \\ M_G \\ N_G \end{pmatrix} = \begin{pmatrix} -(W - B)\sin\theta \\ (W - B)\cos\theta\sin\phi \\ (W - B)\cos\theta\cos\phi \\ (y_G W - y_B B)\cos\theta\cos\phi - (z_G W - z_B B)\cos\theta\sin\phi \\ -(x_G W - x_B B)\cos\theta\cos\phi - (z_G W - z_B B)\sin\theta \\ (x_G W - x_B B)\cos\theta\sin\phi + (y_G W - y_B B)\sin\theta \end{pmatrix} \tag{15}$$

Although the propulsion forces $\mathcal{F}_P$ are also hydrodynamic forces, it is convenient to separate and give them special attention. Following Mackay [12], local propeller axes, x', y', z' say,

are defined with their origin located on the propeller axis opposite the blades, at $x_P\mathbf{i}+y_P\mathbf{j}+z_P\mathbf{k}$ in vehicle body axes, and with the x' axis coincident with the propeller axis. If the x and x' axes are not parallel, then the usual Euler yaw and pitch angle transformations, using (ψ_P, θ_P), are used to describe the local propeller axes orientation relative to vehicle body axes. If X_P, K_P are the propeller net-thrust and torque in local propeller axes, and if crossflow forces on the propeller can be ignored, then:

$$\mathcal{F}_P = \begin{pmatrix} X_P \cos\psi_P \cos\theta_P \\ X_P \sin\psi_P \cos\theta_P \\ -X_P \sin\theta_P \\ K_P \cos\psi_P \cos\theta_P - X_P(z_P \sin\psi_P \cos\theta_P + y_P \sin\theta_P) \\ K_P \sin\psi_P \cos\theta_P + X_P(z_P \cos\psi_P \cos\theta_P + x_P \sin\theta_P) \\ -K_P \sin\theta_P - X_P(y_P \cos\psi_P - x_P \sin\psi_P)\cos\theta_P \end{pmatrix}, \quad \lim_{\psi_P,\theta_P \to 0} \mathcal{F}_P = \begin{pmatrix} X_P \\ 0 \\ 0 \\ K_P \\ z_P X_P \\ -y_P X_P \end{pmatrix} \tag{16}$$

In many cases, $\psi_P, \theta_P = 0$ which greatly simplifies (16). Propeller thrust and torque models are discussed in a later section.

The remaining hydrodynamic forces can be thought of as having inviscid and viscous contributions:

$$\mathcal{F}_H = \mathcal{F}_{\text{inviscid}} + \mathcal{F}_{\text{viscous}} \tag{17}$$

For the UUV docking simulation, the viscous forces are the steady or quasi-steady forces on the vehicle which are resolvable in captive model experiments with the propeller absent. They can be a function of any non-acceleration vehicle state (ie, the $d\mathcal{V}/dt$ states are excluded) or associated system states, or the time history of those states. Although this forces the instantaneous generation of circulation (a viscous phenomenon) on the vehicle as incidence angles change, it still allows for the convection along the hull of sail generated circulation to be modelled in a quasi-steady manner using slender body or strip theory, as done by Feldman [8] and also by Mackay [12]. As a result, the only significant hydrodynamic forces with a direct dependence on the acceleration states are inertial in nature, and these are fairly well predicted using potential flow theory.

Lamb [13] and Watt [10] derive the complete steady and unsteady forces on a deeply submerged body moving with 6 DOF through an inviscid incompressible fluid. Imlay [14] and Fossen [6] also present the results. These forces have the form:

$$\mathcal{F}_{\text{potflow}} = \mathcal{A} \cdot \frac{d\mathcal{V}}{dt} + \mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V} \tag{18}$$

where:

$$\mathcal{A} = \begin{pmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{pmatrix} \tag{19}$$

is the added mass matrix. The added mass matrix coefficients are a function of geometry only and are constant for a given deeply submerged vehicle, neglecting insignificant geometry

changes such as control surface deflections. $\mathcal{A}$ has special properties: it is always symmetric and, when the vehicle has a vertical plane of symmetry, half the coefficients are zero:

$$\text{Vertical plane of symmetry:} \quad \mathcal{A} \to \mathcal{A}_S = \begin{pmatrix} X_{\dot{u}} & 0 & X_{\dot{w}} & 0 & X_{\dot{q}} & 0 \\ 0 & Y_{\dot{v}} & 0 & Y_{\dot{p}} & 0 & Y_{\dot{r}} \\ Z_{\dot{u}} & 0 & Z_{\dot{w}} & 0 & Z_{\dot{q}} & 0 \\ 0 & K_{\dot{v}} & 0 & K_{\dot{p}} & 0 & K_{\dot{r}} \\ M_{\dot{u}} & 0 & M_{\dot{w}} & 0 & M_{\dot{q}} & 0 \\ 0 & N_{\dot{v}} & 0 & N_{\dot{p}} & 0 & N_{\dot{r}} \end{pmatrix}. \tag{20}$$

Imlay [14], Watt [10], and Fossen [6] discuss ways of estimating the added mass coefficients for a deeply submerged vehicle based on exact results for ellipsoids.

The unsteady inviscid term $\mathcal{A} \cdot d\mathcal{V}/dt$ from (18) agrees well with real flow $\dot{v}, \dot{w}, \dot{q}, \dot{r}$ acceleration experiments that measure most of the large added mass coefficients. That is, potential flow predictions of important inertia effects absent from the steady and quasi-steady viscous models are adequate where they can be validated. This term can be combined with the steady/quasi-steady viscous hydrodynamic force models without fear of duplication.

The same cannot be said of the $\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V}$ term from (18), shown here in its entirety but with the terms that are zero when there is a vertical plane of symmetry shown in blue:

$$\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V} = \begin{pmatrix} -Y_{\dot{v}}vr - Y_{\dot{p}}pr - Y_{\dot{r}}r^2 + Z_{\dot{u}}uq + Z_{\dot{w}}wq + Z_{\dot{q}}q^2 \\ \quad - Y_{\dot{u}}ur - Y_{\dot{w}}wr - Y_{\dot{q}}qr + Z_{\dot{v}}vq + Z_{\dot{p}}pq + Z_{\dot{r}}qr \\ X_{\dot{u}}ur + X_{\dot{w}}wr + X_{\dot{q}}qr - Z_{\dot{u}}up - Z_{\dot{w}}wp - Z_{\dot{q}}pq \\ \quad + X_{\dot{v}}vr + X_{\dot{p}}pr + X_{\dot{r}}r^2 - Z_{\dot{v}}vp - Z_{\dot{p}}p^2 - Z_{\dot{r}}pr \\ -X_{\dot{u}}uq - X_{\dot{w}}wq - X_{\dot{q}}q^2 + Y_{\dot{v}}vp + Y_{\dot{p}}p^2 + Y_{\dot{r}}pr \\ \quad - X_{\dot{v}}vq - X_{\dot{p}}pq - X_{\dot{r}}qr + Y_{\dot{u}}up + Y_{\dot{w}}wp + Y_{\dot{q}}pq \\ -Y_{\dot{v}}vw - Y_{\dot{p}}wp - Y_{\dot{r}}wr + Z_{\dot{u}}uv + Z_{\dot{w}}vw + Z_{\dot{q}}vq \\ \quad - Y_{\dot{u}}uw - Y_{\dot{w}}w^2 - Y_{\dot{q}}wq + Z_{\dot{v}}v^2 + Z_{\dot{p}}vp + Z_{\dot{r}}vr \\ \quad - M_{\dot{u}}ur - M_{\dot{w}}wr - M_{\dot{q}}qr + N_{\dot{v}}vq + N_{\dot{p}}pq + N_{\dot{r}}qr \\ \quad - M_{\dot{v}}vr - M_{\dot{p}}pr - M_{\dot{r}}r^2 + N_{\dot{u}}uq + N_{\dot{w}}wq + N_{\dot{q}}q^2 \\ X_{\dot{u}}uw + X_{\dot{w}}w^2 + X_{\dot{q}}wq - Z_{\dot{u}}u^2 - Z_{\dot{w}}uw - Z_{\dot{q}}uq \\ \quad + X_{\dot{v}}vw + X_{\dot{p}}wp + X_{\dot{r}}wr - Z_{\dot{v}}uv - Z_{\dot{p}}up - Z_{\dot{r}}ur \\ \quad + K_{\dot{v}}vr + K_{\dot{p}}pr + K_{\dot{r}}r^2 - N_{\dot{v}}vp - N_{\dot{p}}p^2 - N_{\dot{r}}pr \\ \quad + K_{\dot{u}}ur + K_{\dot{w}}wr + K_{\dot{q}}qr - N_{\dot{u}}up - N_{\dot{w}}wp - N_{\dot{q}}pq \\ -X_{\dot{u}}uv - X_{\dot{w}}vw - X_{\dot{q}}vq + Y_{\dot{v}}uv + Y_{\dot{p}}up + Y_{\dot{r}}ur \\ \quad - X_{\dot{v}}v^2 - X_{\dot{p}}vp - X_{\dot{r}}vr + Y_{\dot{u}}u^2 + Y_{\dot{w}}uw + Y_{\dot{q}}uq \\ \quad - K_{\dot{v}}vq - K_{\dot{p}}pq - K_{\dot{r}}qr + M_{\dot{u}}up + M_{\dot{w}}wp + M_{\dot{q}}pq \\ \quad - K_{\dot{u}}uq - K_{\dot{w}}wq - K_{\dot{q}}q^2 + M_{\dot{v}}vp + M_{\dot{p}}p^2 + M_{\dot{r}}pr \end{pmatrix}. \tag{21}$$

These steady state inviscid force predictions contain both rotational loads analogous to those in (11) that result from the rotating frame of reference and steady translational forces such as the historic 'Munk moments' (the pitching moment term $(X_{\dot{u}} - Z_{\dot{w}})uw$ and yawing moment

term $(Y_{\dot{v}} - X_{\dot{u}})uv$). However, these predictions do not agree well with experiments where measurements can be made. Consider, for example, an ellipsoid translating at constant speed and incidence in the vertical plane, so that $d\mathcal{V}/dt = 0$ and $v, p, q, r = 0$. From (21), the body forces on the ellipsoid are zero drag, zero normal force, and $(X_{\dot{u}} - Z_{\dot{w}})uw$ for the pitching moment ($X_{\dot{w}} = Z_{\dot{u}} = 0$ for an ellipsoid). In reality, drag is nonzero, the normal force is measurable, nonlinear, and important to vehicle dynamics, and the Munk moment substantially overpredicts the real moment.

These discrepancies occur because the inviscid predictions do not account for flow shear, circulation, or separation effects that substantially impact steady viscous flows. These effects take time to develop and so do not impact acceleration forces to the same degree. Therefore, it is preferable to use experimental measurements and/or semi-empirical estimation methods to predict these steady forces whenever possible, and the $\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V}$ terms as supplements when no other information is available.

The inviscid force used in (17) is therefore:

$$\mathcal{F}_{\text{inviscid}} = \mathcal{A} \cdot \frac{d\mathcal{V}}{dt} + \mathcal{W}\mathcal{A}\mathcal{V}_{\text{mod}} \tag{22}$$

where $\mathcal{W}\mathcal{A}\mathcal{V}_{\text{mod}}$ contains the physics from $\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V}$ not modelled by $\mathcal{F}_{\text{viscous}}$. The equations of motion can now be put in the form (10):

$$(\mathcal{M} - \mathcal{A})\frac{d\mathcal{V}}{dt} = -\mathcal{W} \cdot \mathcal{M} \cdot \mathcal{V} + \mathcal{F}_{\text{viscous}} + \mathcal{W}\mathcal{A}\mathcal{V}_{\text{mod}} + \mathcal{F}_G + \mathcal{F}_P\,. \tag{23}$$

The $\mathcal{A}$ coefficients and $\mathcal{F}_{\text{viscous}} + \mathcal{W}\mathcal{A}\mathcal{V}_{\text{mod}}$ model are discussed in §4.

Free Surface Effects

The incorporation of free surface effects into the deeply submerged hydrodynamic model for the submarine is discussed in detail in [3] but is briefly summarized here. The worst case UUV docking scenario is when the submarine is travelling at a Froude number below 0.1 with its centerline approximately two hull diameters below the free surface. The steady state inviscid forces resulting from standing waves generated by the forward speed of the boat in calm water are not modelled because their influence can be neutralized with adjustments to propeller RPM and control surface deflections. However, the unsteady forces on the submarine resulting from interaction with surface waves must be modelled. We assume incident wave amplitudes are small relative to their wave lengths and that wave lengths are the same order as the submarine length. A linearized unsteady inviscid analysis shows that the unsteady forces are wave frequency dependent added mass, radiation (damping), and incident and diffraction wave excitation forces [15]. These forces can be predicted by a frequency domain potential flow analysis by a program like ShipMo3D [4] and stored in a database as a function of wave frequency and heading and vehicle speed and depth. The radiation and incident and diffraction wave excitation data are converted to time domain forces using the Cummins equation [16] extended to a body-fixed frame [3]. They are zero at zero and infinite wave frequencies and at infinite depth, and can be superposed on the deeply submerged quasi-steady hydrodynamic forces. The ShipMo3D added mass coefficients, on the other hand, reduce to the deeply submerged added mass coefficients (19) at infinite depth, so they must replace the deeply submerged added mass coefficients in (23).

The unsteady forces added in this manner are purely inviscid in nature and do not account for the unsteady viscous interaction between the wave induced flow and the lifting components on the vehicle. This interaction alters the circulation and therefore the lift on an appendage. It is not accounted for in the current version of the UUV docking simulation [3]. However, it is possible to do so by applying Morison's method (see next paragraph) on a localized basis to individual vehicle control surfaces, and to superpose these unsteady effects on the current quasi-steady viscous force models. This should work smoothly providing the submarine is fast enough that flow reversal does not occur.

For the UUV, free surface effects are modelled using Morison's formula [15], a quasi-steady approach that superposes wave field velocities on the actual UUV velocities in (23) to model incident wave excitation. This is acceptable because UUVs are small compared with incident surface wave lengths, so the wave induced velocity they experience is approximately uniform at any given time. Also, being many hull diameters below the free surface, UUVs are effectively deeply submerged; that is, added mass coefficients retain their constant deeply submerged values and radiation and diffraction effects are negligible.

4 The Deeply Submerged Hydrodynamic Coefficient Model

The viscous hydrodynamic models for the UUV docking vehicles are based on coefficients obtained from the DRDC Submarine Simulation Program (DSSP) [12]. DSSP also incorporates capabilities from the Estimate Submarine Added Masses (ESAM) program [10] that let it predict deeply submerged added mass coefficients. However, DSSP does not make use of ESAM's ability to model hull interference effects on appendages nor does it provide the contributions of individual vehicle components to the added mass totals. These advantages are obtained by using ESAM directly. Component contributions to the added masses are needed if a simplified ShipMo3D model (see §3.1) is used that neglects small appendages. For example, tailplanes and bowplanes add complexity and computation time to ShipMo3D predictions but have minimal impact on near surface added mass frequency dependence; however, these control surfaces do impact the constant deeply submerged added mass magnitudes. Therefore, simplified ShipMo3D added mass predictions can be made complete by adding the missing component constant contributions from ESAM.

DSSP runs in several modes. It simulates static towing tank tests, captive model dynamic tests, or free swimming maneuvers. It can generate maneuvering limitation diagrams (MLDs) or a conventional Gertler and Hagen [7] type coefficient based hydrodynamic model for use by itself or other simulators. Normally DSSP uses a nonlinear component based hydrodynamic model for forces that is more sophisticated than a coefficient based model; it also models the convection in time of sail generated vorticity along the submarine afterbody (as per Feldman [8]) during dynamic simulations, which is not done by conventional coefficient based models. This level of sophistication is likely overkill in our UUV docking simulations so we avoid its complexity by using DSSP's version of the conventional Gertler and Hagen model, as described next.

The $\mathcal{F}_{\text{viscous}}$ components for (23) are based on the Gertler and Hagen [7] type hydrodynamic coefficients generated by DSSP51 (DSSP version 5.1) in coefficient generation mode. These empirically determined functions (24) are shown on the next page. They are not readily represented as products of matrices and vectors, as was possible in the last section. They do not attempt to reproduce the $\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V}$ physics but, as discussed next, do duplicate some of it. Table 1 lists all the coefficients used in these equations.

Viscous Axial Force

$$X_{\text{viscous}} = \left(X_{uu} + X_{uu\delta_b\delta_b}\delta_b^2 + X_{uu\delta_r\delta_{r0}}\delta_r^2 + X_{uu\delta_s\delta_{s0}}\delta_s^2\right)u^2 + X_{vv0}v^2 + X_{vr}vr + X_{ww0}w^2 + X_{wq}wq + X_{pr}pr + X_{qq}q^2 + X_{rr}r^2 \quad (24a)$$

Viscous Lateral Force

$$Y_{\text{viscous}} = \left(Y_{uu} + Y_{uu\delta_b}\delta_b + Y_{uu\delta_{r0}}\delta_r + Y_{uu\delta_s}\delta_s\right)u^2 + \left(Y_{uv0}v + Y_{up}p + Y_{ur0}r + Y_{u|r|\delta_r}|r|\delta_r\right)u + Y_{vw}vw + Y_{vq}vq + Y_{wp}wp + Y_{wr}wr + Y_{pq}pq + Y_{p|p|}p|p| + Y_{qr}qr + Y_{r|r|}r|r| + \left(Y_{v\nu0}v + Y_{v\nu|r|v|}\,\text{sign}(r,v)\right)\sqrt{v^2+w^2} \quad (24b)$$

Viscous Normal Force

$$Z_{\text{viscous}} = \left(Z_{uu} + Z_{uu\delta_b}\delta_b + Z_{uu\delta_s0}\delta_s\right)u^2 + \left(Z_{uw0}w + Z_{uq0}q + Z_{u|w|}|w| + Z_{u|q|\delta_s}|q|\delta_s\right)u + Z_{vv}v^2 + Z_{vp}vp + Z_{vr}vr + Z_{pp}p^2 + Z_{pr}pr + Z_{q|q|}q|q| + Z_{rr}r^2 + \left(Z_{w\nu0}w + Z_{|w\nu|}|w| + Z_{w\nu|q|w|}\,\text{sign}(q,w)\right)\sqrt{v^2+w^2} \quad (24c)$$

Viscous Rolling Moment

$$K_{\text{viscous}} = \left(K_{uu0} + K_{uu\delta_b}\delta_b + K_{uu\delta_{r0}}\delta_r + K_{uu\delta_s0}\delta_s\right)u^2 + \left(K_{uv}v + K_{up}p + K_{ur}r\right)u + K_{vw}vw + K_{vq}vq + K_{wp}wp + K_{wr}wr + K_{pq}pq + K_{p|p|}p|p| + K_{qr}qr + K_{v\nu}v\sqrt{v^2+w^2} \quad (24d)$$

Viscous Pitching Moment

$$M_{\text{viscous}} = \left(M_{uu} + M_{uu\delta_b}\delta_b + M_{uu\delta_r\delta_{r0}}\delta_r^2 + M_{uu\delta_s0}\delta_s\right)u^2 + \left(M_{uw0}w + M_{uq0}q + M_{u|w|}|w| + M_{u|q|\delta_s}|q|\delta_s\right)u + M_{vv}v^2 + M_{vp}vp + M_{vr}vr + M_{pp}p^2 + M_{pr}pr + M_{q|q|}q|q| + M_{rr}r^2 + \left(M_{w\nu0}w + M_{|w\nu|}|w| + M_{q\nu}q\right)\sqrt{v^2+w^2} \quad (24e)$$

Viscous Yawing Moment

$$N_{\text{viscous}} = \left(N_{uu} + N_{uu\delta_b\delta_b}\delta_b^2 + N_{uu\delta_r0}\delta_r + N_{uu\delta_s\delta_{s0}}\delta_s^2\right)u^2 + \left(N_{uv0}v + N_{up}p + N_{ur0}r + N_{u|r|\delta_r}|r|\delta_r\right)u + N_{vw}vw + N_{vq}vq + N_{wp}wp + N_{wr}wr + N_{pq}pq + N_{qr}qr + N_{r|r|}r|r| + \left(N_{v\nu0}v + N_{r\nu}r\right)\sqrt{v^2+w^2} \quad (24f)$$

NOTE: $\text{sign}(x,y) = y\,|x/y|$ gives the magnitude $|x|$ the sign of y.

Axial Force	**Lateral Force**	**Normal Force**	**Rolling Moment**	**Pitching Moment**	**Yawing Moment**
X_{uu}	Y_{uu}	Z_{uu}	K_{uu0}	M_{uu}	N_{uu}
$X_{uu\delta_b\delta_b}$	$Y_{uu\delta_b}$	$Z_{uu\delta_b}$	$K_{uu\delta_b}$	$M_{uu\delta_b}$	$N_{uu\delta_b\delta_b}$
$X_{uu\delta_r\delta_r 0}$	$Y_{uu\delta_r 0}$	$Z_{uu\delta_s 0}$	$K_{uu\delta_r 0}$	$M_{uu\delta_r\delta_r 0}$	$N_{uu\delta_r 0}$
$X_{uu\delta_s\delta_s 0}$	$Y_{uu\delta_s}$	Z_{uw0}	$K_{uu\delta_s 0}$	$M_{uu\delta_s 0}$	$N_{uu\delta_s\delta_s 0}$
X_{vv0}	Y_{uv0}	Z_{uq0}	K_{uv}	M_{uw0}	N_{uv0}
X_{vr}	Y_{up}	$Z_{u\|w\|}$	K_{up}	M_{uq0}	N_{up}
X_{ww0}	Y_{ur0}	$Z_{u\|q\|\delta_s}$	K_{ur}	$M_{u\|w\|}$	N_{ur0}
X_{wq}	$Y_{u\|r\|\delta_r}$	Z_{vv}	K_{vw}	$M_{u\|q\|\delta_s}$	$N_{u\|r\|\delta_r}$
X_{pr}	Y_{vw}	Z_{vp}	K_{vq}	M_{vv}	N_{vw}
X_{qq}	Y_{vq}	Z_{vr}	$K_{v\nu}$	M_{vp}	N_{vq}
X_{rr}	$Y_{v\nu 0}$	$Z_{w\nu 0}$	K_{wp}	M_{vr}	$N_{v\nu 0}$
	$Y_{v\nu\|r\|v\|}$	$Z_{w\nu\|q\|w\|}$	K_{wr}	$M_{w\nu 0}$	N_{wp}
	Y_{wp}	Z_{pp}	K_{pq}	M_{pp}	N_{wr}
	Y_{wr}	Z_{pr}	$K_{p\|p\|}$	M_{pr}	N_{pq}
	Y_{pq}	$Z_{q\|q\|}$	K_{qr}	$M_{q\nu}$	N_{qr}
	$Y_{p\|p\|}$	Z_{rr}		$M_{q\|q\|}$	$N_{r\nu}$
	Y_{qr}	$Z_{\|w\nu\|}$		M_{rr}	$N_{r\|r\|}$
	$Y_{r\|r\|}$			$M_{\|w\nu\|}$	

Table 1 *The 96 DSSP51 viscous coefficients ($\nu = \sqrt{v^2 + w^2}$). DSSP anticipates a future propulsive state dependency in some coefficients by adding '0' to their subscripts to indicate propulsive state is neglected (this gets dropped in some places in this report).*

The $\mathcal{WAV}_{\text{mod}}$ vector for (23) is obtained with the following reasoning. All $\mathcal{W}\cdot\mathcal{A}\cdot\mathcal{V}$ terms are retained that do not duplicate the physics already contained in DSSP empirical models. DSSP modelling that does account for an inviscid $\mathcal{W}\cdot\mathcal{A}\cdot\mathcal{V}$ term must be based on an experiment that reproduced the coupled motions described by the $\mathcal{W}\cdot\mathcal{A}\cdot\mathcal{V}$ term. For example, the $\mathcal{W}\cdot\mathcal{A}\cdot\mathcal{V}$ lateral force term $X_{\dot{q}}qr$ is retained because the DSSP term $Y_{qr}qr$ is not based on a model test in which q and r motions are modelled simultaneously. DSSP generates its Y_{qr} coefficient based on viscous effects (primarily due to circulation) anticipated by a rational model of the local flow interacting with local geometry, effects that are nonexistent in a potential flow. In this case, the best that can be done is to superpose the potential and viscous terms. Thus:

- $\mathcal{W}\cdot\mathcal{A}\cdot\mathcal{V}$ terms modelling pure translation (ie, terms second order in u, v, or w) are deleted since DSSP force and moment models are based on coupled translation experiments.
- $\mathcal{W}\cdot\mathcal{A}\cdot\mathcal{V}$ lateral force terms $X_{\dot{u}}ur + X_{\dot{r}}r^2$ and yawing moment term $Y_{\dot{r}}ur$ (there is no yawing moment r^2 term) are deleted because DSSP empirically models in-plane forces from coupled u and r motions. The DSSP models do not account for simultaneous rotation and in-plane incidence (vr effects).
- $\mathcal{W}\cdot\mathcal{A}\cdot\mathcal{V}$ normal force terms $-X_{\dot{u}}uq - X_{\dot{q}}q^2$ and pitching moment term $-Z_{\dot{q}}uq$ are deleted for identical reasons.

All other $\mathcal{W}\cdot\mathcal{A}\cdot\mathcal{V}$ terms are retained. Unlike the current DSSP rotation models, many rotation experiment databases do include in-plane incidence effects as well as measurements for all forces

and moments, so this $\mathcal{WAV}_{\text{mod}}$ vector is particular to the current version of DSSP (DSSP51):

$$\mathcal{WAV}_{\text{mod}} = \begin{pmatrix} -Y_{\dot{v}}vr - Y_{\dot{p}}pr - Y_{\dot{r}}r^2 + Z_{\dot{u}}uq + Z_{\dot{w}}wq + Z_{\dot{q}}q^2 \\ \quad - Y_{\dot{u}}ur - Y_{\dot{w}}wr - Y_{\dot{q}}qr + Z_{\dot{v}}vq + Z_{\dot{p}}pq + Z_{\dot{r}}qr \\ \\ X_{\dot{w}}wr + X_{\dot{q}}qr - Z_{\dot{u}}up - Z_{\dot{w}}wp - Z_{\dot{q}}pq \\ \quad + X_{\dot{v}}vr + X_{\dot{p}}pr - Z_{\dot{v}}vp - Z_{\dot{p}}p^2 - Z_{\dot{r}}pr \\ \\ -X_{\dot{w}}wq + Y_{\dot{v}}vp + Y_{\dot{p}}p^2 + Y_{\dot{r}}pr \\ \quad - X_{\dot{v}}vq - X_{\dot{p}}pq - X_{\dot{r}}qr + Y_{\dot{u}}up + Y_{\dot{w}}wp + Y_{\dot{q}}pq \\ \\ -Y_{\dot{p}}wp - Y_{\dot{r}}wr + Z_{\dot{q}}vq - Y_{\dot{q}}wq + Z_{\dot{p}}vp + Z_{\dot{r}}vr \\ \quad - M_{\dot{u}}ur - M_{\dot{w}}wr - M_{\dot{q}}qr + N_{\dot{v}}vq + N_{\dot{p}}pq + N_{\dot{r}}qr \\ \quad - M_{\dot{v}}vr - M_{\dot{p}}pr - M_{\dot{r}}r^2 + N_{\dot{u}}uq + N_{\dot{w}}wq + N_{\dot{q}}q^2 \\ \\ X_{\dot{q}}wq + X_{\dot{p}}wp + X_{\dot{r}}wr - Z_{\dot{p}}up - Z_{\dot{r}}ur \\ \quad + K_{\dot{v}}vr + K_{\dot{p}}pr + K_{\dot{r}}r^2 - N_{\dot{v}}vp - N_{\dot{p}}p^2 - N_{\dot{r}}pr \\ \quad + K_{\dot{u}}ur + K_{\dot{w}}wr + K_{\dot{q}}qr - N_{\dot{u}}up - N_{\dot{w}}wp - N_{\dot{q}}pq \\ \\ -X_{\dot{q}}vq + Y_{\dot{p}}up - X_{\dot{p}}vp - X_{\dot{r}}vr + Y_{\dot{q}}uq \\ \quad - K_{\dot{v}}vq - K_{\dot{p}}pq - K_{\dot{r}}qr + M_{\dot{u}}up + M_{\dot{w}}wp + M_{\dot{q}}pq \\ \quad - K_{\dot{u}}uq - K_{\dot{w}}wq - K_{\dot{q}}q^2 + M_{\dot{v}}vp + M_{\dot{p}}p^2 + M_{\dot{r}}pr \end{pmatrix}. \tag{25}$$

As with (21), the terms in blue are zero when the vehicle has a vertical plane of symmetry. The $\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V}$ terms accounted for by DSSP are therefore:

$$\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V} - \mathcal{WAV}_{\text{mod}} = \begin{pmatrix} 0 \\ X_{\dot{u}}ur + X_{\dot{r}}r^2 \\ -X_{\dot{u}}uq - X_{\dot{q}}q^2 \\ -Y_{\dot{v}}vw + Z_{\dot{u}}uv + Z_{\dot{w}}vw - Y_{\dot{u}}uw - Y_{\dot{w}}w^2 + Z_{\dot{v}}v^2 \\ X_{\dot{u}}uw + X_{\dot{w}}w^2 - Z_{\dot{u}}u^2 - Z_{\dot{w}}uw - Z_{\dot{q}}uq + X_{\dot{v}}vw - Z_{\dot{v}}uv \\ -X_{\dot{u}}uv - X_{\dot{w}}vw + Y_{\dot{v}}uv + Y_{\dot{r}}ur - X_{\dot{v}}v^2 + Y_{\dot{u}}u^2 + Y_{\dot{w}}uw \end{pmatrix} \tag{26}$$

Table 2 compares the coefficients from common $\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V}$ and DSSP terms for a generic submarine geometry. Many $\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V}$ coefficients are too small to matter and could be ignored. Others are potentially significant and deserve additional study to determine their accuracy, perhaps using computational fluid dynamics.

Appendices A and B list the coefficient values for all the viscous and added mass coefficients for the vehicles used in the UUV docking simulation. These values depend only on vehicle geometry and the form of the above equations. As previously discussed, this coefficient model is not perfect but is adequate for the UUV docking simulation.

$F_{ij} \cdot i \cdot j$	$\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V}$ Coefficient $\times 1000$	DSSP Coefficient $\times 1000$	$\frac{\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V} \text{ Coeff.}}{\text{DSSP Coeff.}}$
$X_{vr}vr$	$-Y'_{\dot{v}} = 27.7768$	$X'_{vr} = -6.8886$	-4.03
$X_{wq}wq$	$Z'_{\dot{w}} = -22.6347$	$X'_{wq} = 9.8280$	-2.30
$X_{pr}pr$	$-Y'_{\dot{p}} = 0.3735$	$X'_{pr} = 0.1952$	1.91
$X_{qq}q^2$	$Z'_{\dot{q}} = -0.3666$	$X'_{qq} = 2.7437$	-0.13
$X_{rr}r^2$	$-Y'_{\dot{r}} = -0.0681$	$X'_{rr} = 2.1967$	-0.03
$Y_{up}up$	$-Z'_{\dot{u}} = 0.0039$	$Y'_{up} = -5.2360$	-0.00
$Y_{wp}wp$	$-Z'_{\dot{w}} = 22.6347$	$Y'_{wp} = -11.5692$	-1.96
$Y_{wr}wr$	$X'_{\dot{w}} = -0.0039$	$Y'_{wr} = -7.7509$	0.00
$Y_{pq}pq$	$-Z'_{\dot{q}} = 0.3666$	$Y'_{pq} = -1.4961$	-0.25
$Y_{qr}qr$	$X'_{\dot{q}} = 0.0135$	$Y'_{qr} = -1.1151$	-0.01
$^*Y_{ur}ur$	$X'_{\dot{u}} = -1.0439$	$Y'_{ur0} = 6.7718$	-0.15
$Z_{vp}vp$	$Y'_{\dot{v}} = -27.7768$	$Z'_{vp} = 12.9317$	-2.15
$Z_{pp}p^2$	$Y'_{\dot{p}} = -0.3735$	$Z'_{pp} = 0.0000$	$-\infty$
$Z_{pr}pr$	$Y'_{\dot{r}} = 0.0681$	$Z'_{pr} = -1.8625$	-0.04
$^*Z_{uq}uq$	$-X'_{\dot{u}} = 1.0439$	$Z'_{uq0} = -9.1553$	-0.11
$K_{ur}ur$	$-M'_{\dot{u}} = -0.0135$	$K'_{ur} = 0.0211$	-0.64
$K_{vq}vq$	$N'_{\dot{v}} + Z'_{\dot{q}} = -0.2986$	$K'_{vq} = 0.2669$	-1.12
$K_{wp}wp$	$-Y'_{\dot{p}} = 0.3735$	$K'_{wp} = 0.0000$	∞
$K_{wr}wr$	$-M'_{\dot{w}} - Y'_{\dot{r}} = 0.2986$	$K'_{wr} = 0.0610$	4.90
$K_{pq}pq$	$N'_{\dot{p}} = -0.0144$	$K'_{pq} = 0.0296$	-0.49
$K_{qr}qr$	$-M'_{\dot{q}} + N'_{\dot{r}} = -0.1204$	$K'_{qr} = 0.0108$	-11.15
$^*K_{uv}uv$	$Z'_{\dot{u}} = -0.0039$	$K'_{uv} = -3.4554$	0.00
$^*K_{vw}vw$	$-Y'_{\dot{v}} + Z'_{\dot{w}} = 5.1421$	$K'_{vw} = 0.0000$	∞
$M_{vp}vp$	$-N'_{\dot{v}} = -0.0681$	$M'_{vp} = 1.9672$	-0.03
$M_{vr}vr$	$K'_{\dot{v}} = -0.3735$	$M'_{vr} = -3.7042$	0.10
$M_{pp}p^2$	$-N'_{\dot{p}} = 0.0144$	$M'_{pp} = -0.0024$	-6.06
$M_{pr}pr$	$K'_{\dot{p}} - N'_{\dot{r}} = 1.2754$	$M'_{pr} = -0.5339$	-2.39
$M_{rr}r^2$	$K'_{\dot{r}} = -0.0144$	$M'_{rr} = -0.4081$	0.04
$^*M_{uu}u^2$	$-Z'_{\dot{u}} = 0.0039$	$M'_{uu} = 0.0101$	0.39
$^*M_{uw}uw$	$X'_{\dot{u}} - Z'_{\dot{w}} = 21.5909$	$M'_{uw0} = 7.4104$	2.91
$^*M_{uq}uq$	$-Z'_{\dot{q}} = 0.3666$	$M'_{uq0} = -6.6453$	-0.06
$N_{up}up$	$M'_{\dot{u}} + Y'_{\dot{p}} = -0.3600$	$N'_{up} = -0.4685$	0.77
$N_{vq}vq$	$-K'_{\dot{v}} - X'_{\dot{q}} = 0.3600$	$N'_{vq} = 5.6816$	0.06
$N_{wp}wp$	$M'_{\dot{w}} = -0.3666$	$N'_{wp} = 1.8715$	-0.20
$N_{pq}pq$	$-K'_{\dot{p}} + M'_{\dot{q}} = -1.1550$	$N'_{pq} = 0.5563$	-2.08
$N_{qr}qr$	$-K'_{\dot{r}} = 0.0144$	$N'_{qr} = 0.4321$	0.03
$^*N_{uv}uv$	$-X'_{\dot{u}} + Y'_{\dot{v}} = -26.7329$	$N'_{uv0} = -15.8623$	1.69
$^*N_{ur}ur$	$Y'_{\dot{r}} = 0.0681$	$N'_{ur0} = -5.3654$	-0.01
$^*N_{vw}vw$	$-X'_{\dot{w}} = 0.0039$	$N'_{vw} = 19.4393$	0.00

*Excluded from $\mathcal{WAV}_{\text{mod}}$

Table 2 *Common $\mathcal{W} \cdot \mathcal{A} \cdot \mathcal{V}$ and DSSP terms and their dimensionless coefficient values for the BB3 submarine (Appendix A).*

5 Heading, Roll, Depth, and Plane Reversal Control

Simulators like DSSP can provide different modes of hydrodynamic control by deflecting individual appendages independently. They can be deflected in pairs, which is appropriate for the top and bottom rudders for heading control, the fore and sternplanes can be coupled and deflected simultaneously for depth control, or all the tailplanes can be coupled for roll control (which is unusual but sometimes is used in UUVs).

For unconventional 'X' or '⅄' tailplane configurations, the equivalent of sternplane and rudder control can still be provided using δ_r, δ_s derivatives by associating appropriate combinations of appendage deflections with the required control mode. For example, an X rudder provides depth control by deflecting all its planes either up or down, and heading control by deflecting them all to the same side. Simultaneous depth and heading control is achieved by superposing the individual appendage deflection requirements.

For the UUV docking simulation, we define rudder, sternplane, foreplane, and roll control deflections, $\delta_r, \delta_s, \delta_b$, and δ_ϕ, as virtual deflections that are created from combinations of control surface deflections δ_i, $i = 1$ to N_c, where the vehicle has N_c available control surfaces to deflect. This is done by assigning weights k_{ji} for each mode j to each control surface i as follows:

$$\delta_i = \delta_{ti} + k_{ri}\,\delta_r + k_{si}\delta_s + k_{bi}\delta_b + k_{\phi i}\,\delta_\phi. \tag{27}$$

The δ_{ti} offset is a trim term that may be used to establish equilibrium at the beginning of a simulation (see §7). The limits on plane deflections are control surface specific, not deflection mode specific, so each δ_i needs to be monitored separately to ensure its limit is not exceeded. Also, the δ_i are defined as positive when the deflection is clockwise looking radially outward (the right hand rule with your thumb pointing outward). As a result, k_{si} for a starboard sternplane in a '+' tail configuration will have a sign opposite to that for the port sternplane, as shown in Table 3. This is consistent with how control surface deflections and limits are implemented in DSSP and it provides the flexibility necessary for handling unusual control requirements (such as roll control) as well as unusual tailplane configurations.

Control Surface	i	k_{ri}	k_{si}	k_{bi}	$k_{\phi i}$	δ_i Limits (degs.)
Bottom Rudder	1	1.0	0.0	0.0	−1.0	−30.0 to 30.0
Top Rudder	2	−1.0	0.0	0.0	−1.0	−30.0 to 30.0
Starboard Sternplane	3	0.0	1.0	0.0	−1.0	−30.0 to 30.0
Port Sternplane	4	0.0	−1.0	0.0	−1.0	−30.0 to 30.0
Starboard Bowplane	5	0.0	0.0	1.0	0.0	−25.0 to 25.0
Port Bowplane	6	0.0	0.0	−1.0	0.0	−25.0 to 25.0

Table 3 *Control surface indexing and modal weights for a vehicle with a + tail configuration using roll control.*

Heading and roll can be controlled by giving explicit δ_r and δ_ϕ virtual deflection commands. However, depth control is quite different as it is implemented through two sets of planes (foreplanes and sternplanes) and must contend with plane reversal and possibly a pitch limitation. Explicit δ_s and δ_b commands can always be given but it is also convenient to use a higher level virtual depth deflection command δ_D that determines the fore and sternplane

deflections as follows:

$$\begin{aligned} \delta_b &= k_{Db} C_{pr_b}(u)\, \delta_D \\ \delta_s &= k_{Ds} C_{pr_s}(u)\, \delta_D. \end{aligned} \tag{28}$$

For submarines with bowplanes, these weights are $k_{Ds} = 1$ and $k_{Db} = -1$, so a positive δ_D pitches the nose down allowing the submarine to increase depth by driving forward. The $C_{pr_s}(u)$ function provides plane reversal compensation as a function of speed and is discussed below. Bowplanes are sometimes disabled at high speed, which can be implemented through the $C_{pr_b}(u)$ function. Retracting the bowplanes cannot be modelled without changing vehicle geometry.

If a virtual deflection request becomes arbitrarily high, the associated control surfaces are simply deflected until they hit their stops, which may occur at different times for different control surfaces.

Plane Reversal

Plane reversal is a phenomenon that effects vertical plane control surfaces located aft of the pitch neutral point x_{np}, the center of pressure of the hydrodynamic normal force. A linearized analysis, good for small incidence angles, gives:

$$\frac{x_{\mathrm{np}}}{\ell} = \frac{-M'_{uw}}{Z'_{uw}} \tag{29}$$

This puts x_{np} from 10 to 30% of the hull length aft of the nose. A nonlinear analysis shows that x_{np} moves towards the midhull region as incidence increases.

Underwater vehicles require tailplanes for stability. These are usually the largest control surfaces so it makes sense to use them for depth control. However, they are aft of the neutral point so to change depth they must generate a normal force in a direction opposite to that of the desired depth change. This pitches the vehicle so the nose is pointing in the direction of the desired depth change, and the propeller drives the vehicle forward in that direction.

The force that pitches the vehicle is the net effect of deflected tailplanes and hull hydrodynamic loading, and varies as u^2. This is opposed by static stability, the static pitch restoring moment (vehicle weight $\times$ $\overline{BG} \times \theta$) which does not change with speed. So as speed increases for a given pitch angle (ie, constant pitch restoring moment), the required tailplane deflection angle must decrease to keep the hydrodynamic force causing the pitch angle constant as well.

The net vertical velocity of the vehicle ($\dot{z}_0$, (7c)) results from two opposing components: the vertical component of the vehicle's axial velocity $-u \sin\theta \sim -u\theta$ driven by the propeller and the vertical component of the normal velocity $w \cos\theta \sim w$ driven by tailplane loading. As speed u increases for a given θ, w/u will decrease because the tailplane deflection is decreasing to keep tailplane loading constant. Similarly, as speed decreases, w/u will increase for a given θ and eventually w will exceed $u\theta$ resulting in the vehicle changing depth in the direction of w even though the nose is pitched the opposite way — see Figure 1. Thus, when $\alpha \sim w/u > \theta$, the planes need to reverse themselves to effect a depth change in the desired sense.

The linearized vertical plane equations of motion for a perfectly trimmed boat in equilibrium with an arbitrary normal force Z_a applied at $x = x_a$ are:

$$0 = Z_{uw} uw + Z_a \tag{30a}$$

$$0 = M_{uw} uw - Z_a x_a - mg\overline{BG}\,\theta. \tag{30b}$$

a) *High speed.* **b)** *Low speed.*

Figure 1 *Vehicle trajectories during depth changes at high and low speed.*

Solving for Z_a from the first equation and substituting into the second gives an expression for the balance of moments in terms of incidence w/u and pitch angle θ:

$$\left(\frac{x_{np}}{\ell} - \frac{x_a}{\ell}\right)\frac{w}{u} + \frac{m'}{Z'_{uw}}\frac{g\overline{BG}\,\theta}{u^2} = 0. \tag{31}$$

As expected, θ is proportional to the moment arm $x_{np} - x_a$ and the applied force which, by (30a), is proportional to w. Since $Z'_{uw} < 0$ and if $x_{np} > x_a$, w and θ will have the same sign.

Tailplane reversal should be triggered when the vehicle passes through that 'critical' state where it is neither ascending nor descending; ie, when $\dot{z}_0 = 0 \Rightarrow \theta \sim w/u$. Substituting this into (31) gives, for a given speed u, the critical point x_{cp} on the hull at which an applied force of any magnitude will not change depth:

$$\frac{x_{cp}}{\ell} = \frac{x_{np}}{\ell} + \frac{m'}{Z'_{uw}}\frac{g\overline{BG}}{u^2}. \tag{32}$$

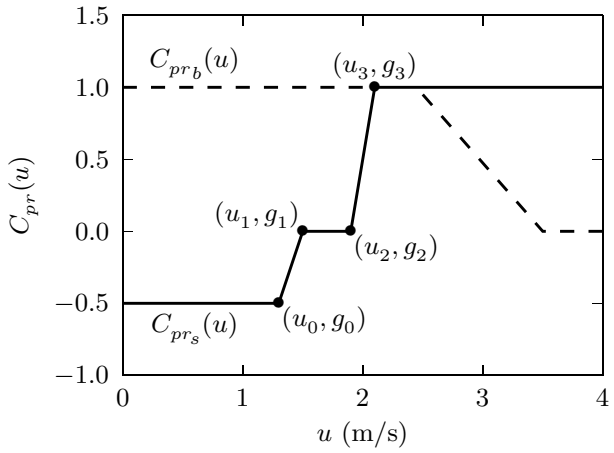
Alternatively, at a given point x_a, the critical velocity u_c at which an applied force will not change depth is:

$$u_c = \sqrt{g\overline{BG}\,\frac{-m'}{Z'_{uw}\left(\frac{x_{\rm np}}{\ell} - \frac{x_a}{\ell}\right)}}. \tag{33}$$

The applied force does change the pitch angle but it does so in concert with w so that the pitch and incidence angles are always equal. Equations (30) apply regardless of whether Z_a is a hydrodynamic force varying with u^2, as assumed here, or a constant buoyancy force as assumed in §7.

For submarines, (33) puts u_c around 5 m/s for sailplanes and from 1 to 2 m/s for sternplanes. For UUVs, u_c reduces as $\sqrt{\overline{BG}}$ and is generally below 0.5 m/s or so for tailplanes. UUVs generally don't bother with plane reversal because they usually maintain a small amount of reserve buoyancy which causes them to loose control at velocities higher than u_c, as discussed in §7.

Since underwater vehicles loose depth control around the critical speed, submarines and some UUVs add foreplanes to compensate. Foreplanes are close to or forward of the neutral point, have little impact on moment, and always generate a force in the direction of the desired depth change.



$$C_{pr_s}(u) = \begin{cases} -0.5, & u < 1.3 \\ -3.75 + 2.5u, & u < 1.5 \\ 0, & u < 1.9 \\ -9.5 + 5.0u, & u < 2.1 \\ 1, & \text{otherwise} \end{cases}$$

$$C_{pr_b}(u) = \begin{cases} 1.0, & u < 2.5 \\ 3.5 - u, & u < 3.5 \\ 0, & u < 5.0 \\ 0, & u < 6.0 \\ 0, & \text{otherwise} \end{cases}$$

Figure 2 *Using the $C_{pr}(u)$ function and (27) to implement sternplane reversal and to disable bowplanes at high speed. Bowplanes are not disabled in the UUV Docking simulation.*

DSSP implements plane reversal using the $C_{pr}(u)$ function:

$$C_{pr}(u) = \begin{cases} g_0, & \text{for } u < u_0 \\ \dfrac{g_0 u_1 - g_1 u_0 + (g_1 - g_0)u}{u_1 - u_0}, & \text{for } u < u_1 \\ \dfrac{g_1 u_2 - g_2 u_1 + (g_2 - g_1)u}{u_2 - u_1}, & \text{for } u < u_2 \\ \dfrac{g_2 u_3 - g_3 u_2 + (g_3 - g_2)u}{u_3 - u_2}, & \text{for } u < u_3 \\ g_3, & \text{otherwise} \end{cases} \tag{34}$$

where $u_c \approx (u_1 + u_2)/2$ and the u_i, g_i coordinates must be supplied. An example of this function is plotted in Figure 2 for submarine sternplanes; the sternplanes are zeroed for a small region around u_c and reversed for speeds clearly below u_c. Also shown in Figure 2 is an example of how this function can be used for disabling the bowplanes at high speeds; note that disabling is not retracting — the undeflected bowplane remains part of the vehicle geometry.

Autopilots

Autopilots are used to provide automatic heading, roll, and depth control when necessary. Autopilots take as input an error signal and use it to calculate and output a correction to δ_r, δ_ϕ, or δ_D. Heading and roll autopilots can use the difference between the desired and current heading and roll angle as their error signal. However, depth changes are generally made using pitch changes and some vehicles (eg, submarines) have pitch limits that the autopilot must accommodate. Therefore, DSSP defines depth error, E_D, as:

$$E_D = \ell_z \sin\theta + \Big(z_{0_c} - z_0\Big)_{-\ell_z \sin\theta_L}^{\ell_z \sin\theta_L} \tag{35}$$

where z_{0_c} is commanded depth, $\ell_z \approx \ell$, and $\theta_L > 0$ is the pitch limit. The first term in (35) is an offset that puts zero error at a point ℓ_z forward of the body axes origin on the body x axis; ℓ_z can be thought of as a 'look ahead' distance. The second term is the depth error

truncated by the indicated limits so that the requested E_D is consistent with the pitch limit; this implements a 'soft' pitch limit, one that may be temporarily exceeded by an under damped autopilot. DSSP uses $\ell_z = \ell/2$ as its default but values as large as ℓ have been used.

The first term in (35) assumes the vehicle is proceeding in the direction it is pointing, which is only approximately correct at speeds over the critical speed. As shown in Figure (1a), the planes that induce the pitch angle also induce a normal velocity w and hence an angle of attack α. Since the vehicle is proceeding in the direction of U, the first term in (35) should really be $\ell_z \sin(\theta - \alpha)$. However, α is not readily known on a real vehicle and it is small relative to θ at high speeds, so DSSP ignores it.

At speeds below the critical speed and when sternplanes are reversed (Figure 1b), the first term in (35) actually corrects in the wrong direction. Therefore, when sternplanes are reversed, when speeds are so low that both θ and α are small, it is probably better to ignore the first term in (35) and not even bother with the pitch limit, as in:

$$E_D = \begin{cases} \text{Equation (35)} & u > u_c \\ z_{0_c} - z_0 & \text{otherwise.} \end{cases} \tag{36}$$

Alternatively, (35) could be improved by estimating the dependence of α and θ on δ_D. Assume the vehicle is trimmed such that the bow and sternplane deflections are fully responsible for α and θ. Then steady state linearized estimates (identified by subscript e) of these quantities can be made using the normal force and pitching moment equations of motion:

$$\alpha_e = w/u = -\frac{Z'_{uu\delta_b}\delta_b + Z'_{uu\delta_s}\delta_s}{Z'_{uw}} \tag{37a}$$

$$\theta_e = \frac{M'_{uu\delta_b}\delta_b + M'_{uu\delta_s}\delta_s + M'_{uw}\alpha_e}{m' g \overline{BG}}\, u^2. \tag{37b}$$

Using (28), a steady state linearized relation between δ_D and θ_e can then be obtained:

$$\frac{m' g \overline{BG}\, \theta_e}{u^2} = \left[\left(x'_{np} - x'_{\delta_b}\right) Z'_{uu\delta_b} k_{Db} C_{pr_b} + \left(x'_{np} - x'_{\delta_s}\right) Z'_{uu\delta_s} k_{Ds} C_{pr_s} \right] \delta_D \tag{38}$$

where $x'_\delta = -M'_{uu\delta}/Z'_{uu\delta}$ is the effective axial location of the force generated by δ. By replacing θ_e in (38) with $\pm\theta_L$, we have simple estimates for the limits the autopilot can apply to the δ_D signal it outputs. With the pitch limit accounted for in this manner, the depth error signal fed to the autopilot can simply be:

$$E_D = \ell_z \sin(\theta - \alpha_e) + z_{0_c} - z_0. \tag{39}$$

The combination of (38) and (39) applies to all speeds and should be better than (35). Hopefully it will avoid chatter at low speed.

Note that the coefficient of δ_D in (38) will be zero for some u close to u_c (when θ_L imposes no limit on δ_D), so it is best to evaluate and test it before moving it to the denominator on LHS of the equation when calculating the δ_D limits. Also, θ in (39) is not θ_e; it is the true pitch angle measured by the vehicle.

Equations (36), (38) and (39) have yet to be tested.

6 Propulsion Loads

Propulsion is modelled in the same manner as in DSSP but independently of it. The propeller is assumed to belong to the Wageningen B 4-70 family of propellers, as presented by van Lammeren et al [17]. These are four-bladed propellers with a blade area ratio $A_E/A_0 = 0.7$. The pitch to diameter ratio P/D can vary from 0.5 to 1.4. The B 4-70 propellers are useful because good four quadrant open water thrust and torque data are available for them, although we only use two of these quadrants because our vehicles are always moving forward. This family of propellers is used to reproduce reasonable propulsion characteristics for most vehicles. This approach is not intended to accurately model a specific propeller and, for this reason, Reynolds number corrections are not made to propeller thrust and torque. A specific propeller for which two quadrant thrust and torque characteristics were known could easily be modelled if that was desirable.

The B 4-70 series open water thrust and torque data are available as the cubic spline surface shown in Figure 3. This surface has been least-squares fitted to digitized data from the first two quadrants of Figure 36 from van Lammeren et al [17]. These data are nondimensionalized in such a way that, unlike the conventional thrust and torque coefficients K_T and K_Q, singularities are avoided as propeller RPM goes through zero:

$$\begin{aligned} C_T &= \frac{T}{\frac{\rho\pi D^2}{8}\left[V_A^2 + (0.7\pi nD)^2\right]} \\ C_Q &= \frac{Q}{\frac{\rho\pi D^3}{8}\left[V_A^2 + (0.7\pi nD)^2\right]}. \end{aligned} \tag{40}$$

These coefficients are plotted as a function of the geometric pitch to diameter ratio P/D of the propeller and the hydrodynamic pitch angle:

$$\beta = \tan^{-1}\frac{V_A}{0.7\pi nD} \tag{41}$$

which avoids the singularity inherent in the conventional advance ratio J at zero RPM:

$$J = \frac{V_A}{nD} = 0.7\pi\tan\beta. \tag{42}$$

The C_T and C_Q surfaces are provided by the Fortran 90 routine listed in Appendix C. The DSSP51 propulsion model used a less smooth and more limited version of the same C_T, C_Q surfaces, but DSSP52 uses this new representation.

We use the same estimates as DSSP for wake fraction w_T and thrust deduction t_D which determine the behind-the-boat performance of the propellers. These empirical corrections are dependent on propeller diameter. The wake fraction w_T is a measure of the wake deficit the propeller sees and is used to relate the speed of advance V_A of the propeller through the local fluid to the forward speed u of the boat:

$$V_A = (1 - w_T)u \tag{43}$$

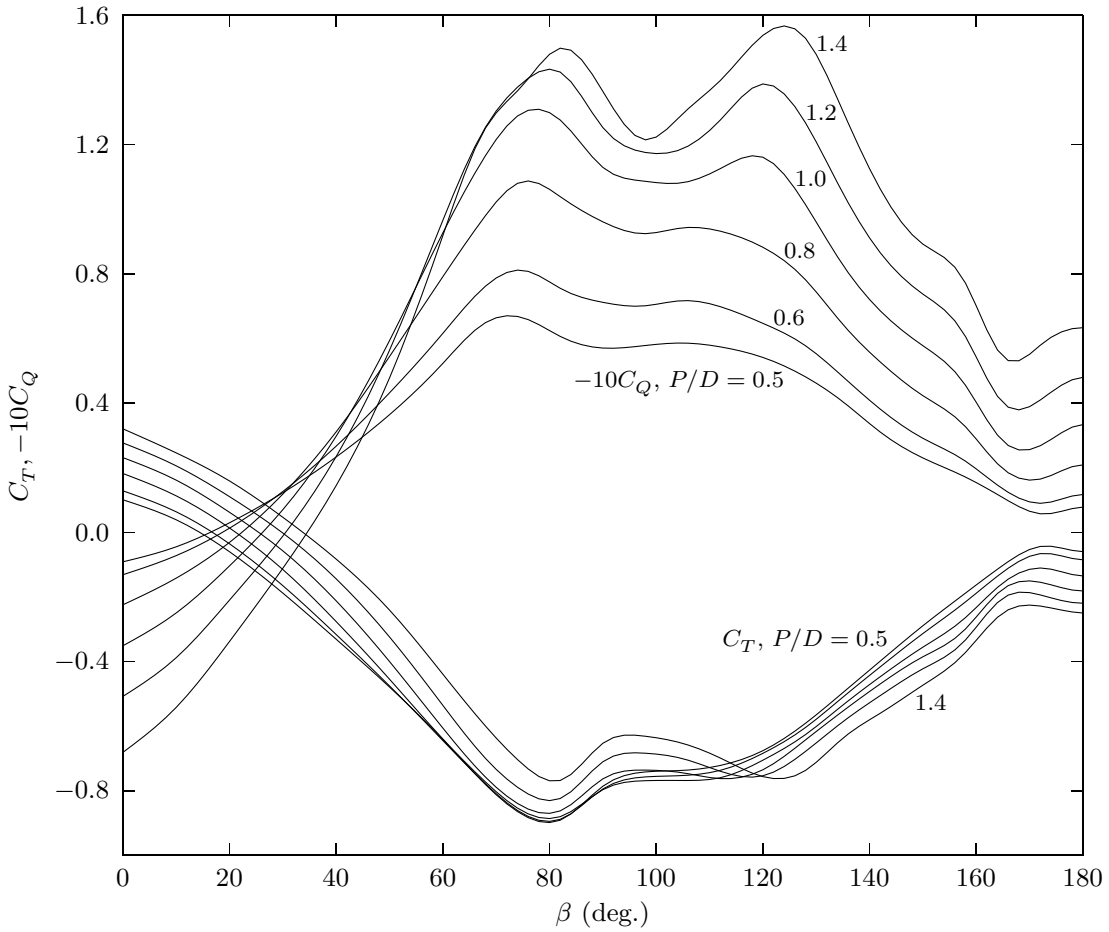


Figure 3 *The 2D cubic spline fit to the Wageningen B 4-70 screw series two quadrant thrust and torque data $C_T(\beta, P/D)$ and $C_Q(\beta, P/D)$.*

The thrust deduction factor t_D accounts for the reduction in pressure on the hull afterbody caused by the propeller. The net thrust and torque on the vehicle are:

$$\begin{aligned} X_P(u,n) &= (1-t_D)T \\ K_P(u,n) &= s_K Q \end{aligned} \tag{44}$$

where:

$$s_K = \begin{cases} -1, & \text{for a righthanded propeller (clockwise rotation looking forward)} \\ 1, & \text{for a lefthanded propeller.} \end{cases} \tag{45}$$

Propeller open water efficiency η_o and behind-the-boat efficiency η_B are [18]:

$$\eta_o = \frac{TV_A}{2\pi n Q_o}, \quad \eta_B = \frac{TV_A}{2\pi n Q} = \frac{J}{2\pi}\frac{K_T}{K_Q} = \frac{J}{2\pi}\frac{C_T}{C_Q} \tag{46}$$

where Q_o is the torque required in open water and Q is the torque required behind the boat. The difference between the two is small and not readily predicted so we just use η_o in what follows, and drop the 'o' subscript on Q.

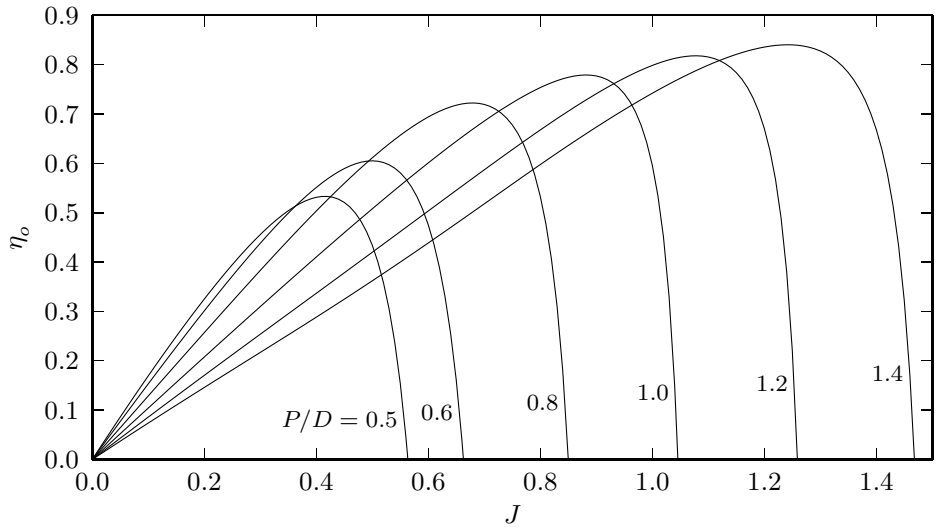


Figure 4 *The openwater efficiency of the Wageningen B 4-70 series propellers, based on the C_T, C_Q surface splines of Figure 3.*

The B 4-70 open-water efficiencies are shown in Figure 4 as a function of advance ratio, as is conventional. These result from measurements in open water with a homogeneous inflow. Behind the boat the inhomogeneous inflow is roughly modelled by the wake fraction.

Propellers are designed for a particular operating point, usually the one where efficiency needs to be maximized. For our vehicles, this is the self-propulsion point, that J value (J_s say) at which the propeller operates when the properly trimmed vehicle is in steady state straight and level flight. J_s may or may not depend on forward speed, depending on how the vehicle is trimmed to achieve equilibrium during straight and level flight (the subject of the next section). For submarines, J_s can be constant for a properly trimmed boat. For UUVs which retain a minimal level of buoyancy, J_s is approximately constant only from moderate to high speeds.

In general, the larger the propeller diameter, the slower its rotational speed for a given thrust and the more efficient it is. However, diameter is usually constrained by practical considerations, such as weight, space, or, for the BB3 submarine, the propeller diameter was determined by that used in a scale model test of its BB2 predecessor [19]. Since our simulation propellers have to be from the B 4-70 series, all that remains to finalize their designs is to choose a P/D ratio, which is done by optimizing η_o at the self-propulsion point.

We define the nominal self-propulsion state for an underwater vehicle to be when it is travelling at constant speed in straight and level flight at zero incidence such that:

$$\dot{z}_0, \dot{y}_0, \dot{u}, v, w, p, q, r, \phi, \theta, \psi, \delta_b, \delta_s, \delta_r = 0. \tag{47}$$

Combining (47) with the axial force component of (23) gives:

$$\begin{aligned} 0 &= X_{uu}u^2 + X_P \\ &= X_{uu}u^2 + (1-t_D)\frac{\rho\pi D^2}{8}\left[V_A^2 + (0.7\pi nD)^2\right] C_T(\beta, P/D) \\ &= X_{uu}u^2 + (1-t_D)\frac{\rho\pi D^2}{8}(1-w_T)^2u^2\left[1+\left(\frac{0.7\pi}{J}\right)^2\right] C_T\left(\tan^{-1}\frac{J}{0.7\pi}, P/D\right) \end{aligned} \tag{48}$$

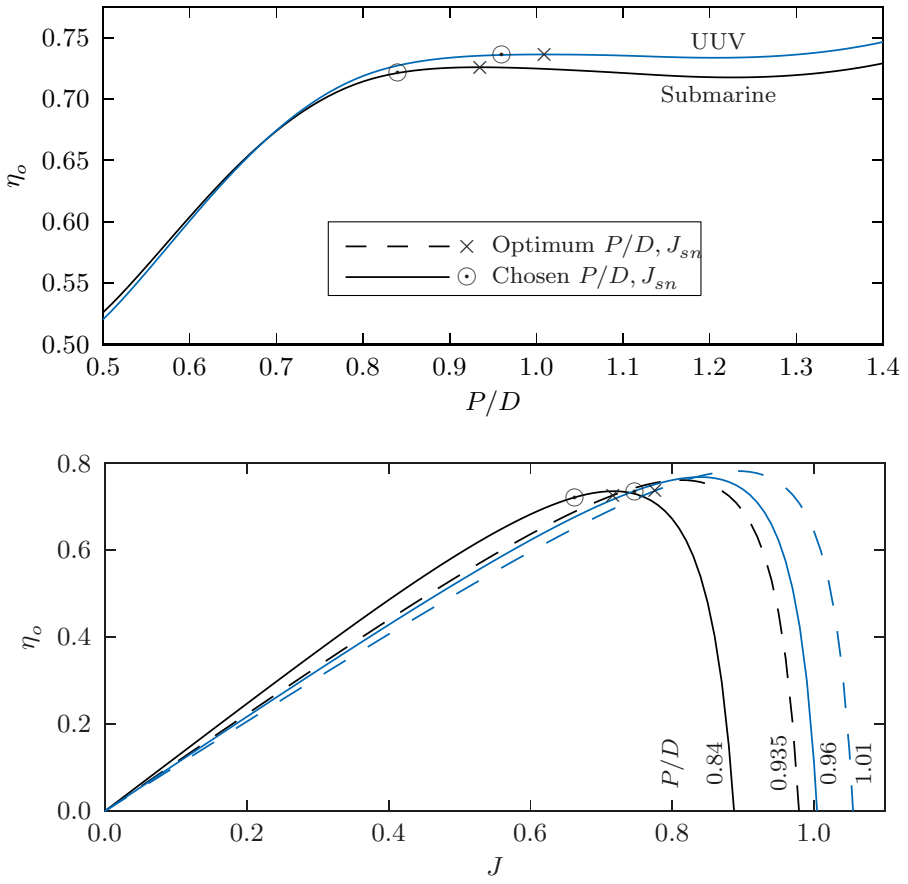


Figure 5 *Propeller open water efficiency as a function of pitch to diameter ratio and advance ratio for the BB3 submarine (Appendix A) and UUV of Appendix B. The nominal self-propulsion points are shown for optimal and actual design choices.*

using (40) through (44). Dividing by u^2, and identifying J with the nominal self-propulsion point J_{sn}, gives an expression for J_{sn} as a function of P/D which is independent of speed. The efficiency (46) as a function of P/D for this ideal self-propulsion state is then obtained and is plotted in Figure 5.

The optimum P/D and J_{sn} ratios in Figure 5 correspond to 10 m/s at 115 RPM for the submarine and 3.5 m/s at 958 RPM for the UUV, the top speeds for each vehicle. Normally adjustments have to be made in propeller geometry to match the available engine and drive train to the power and speed requirements. In that spirit, we have chosen to adjust RPM up in each case so the submarine achieves nominal self-propulsion at 10 m/s at 125 RPM and the UUV does so at 3.5 m/s at 1000 RPM. The effects of these choices are seen in Figure 5.

The self-propulsion advance ratio is only nominal because (48) does not account for any measures needed to trim the boat to achieve equilibrium. These are discussed in the next section. It is usually adequate to use the nominal self-propulsion state to design the propeller — trimming a well designed vehicle should have minimal impact on its hydrodynamic efficiency.

Of course, propeller design is much more complicated than shown here. Propeller diameter, numbers of blades, blade area ratio, vibration, strength of materials, cavitation, and more play a role. What this section does is show how the simulation interacts with the propulsion model and, within that context, that the propulsion parameters are reasonable.

7 Simulation Initialization

It is convenient to begin each simulation with the vehicle in perfect equilibrium so that subsequent motion continues as is until new commands are issued. This avoids long simulation runs to achieve equilibrium prior to beginning the maneuver of interest, and it ensures that any deviation from the equilibrium state following the beginning of a maneuver is wholly attributable to that maneuver.

Vehicle simulations are initialized with the vehicle in a true self-propulsive state at the specified speed. That is, the vehicle is in steady state flight along a straight and level flight path with:

$$\dot{y}_0, \dot{z}_0, p, q, r, \frac{d\mathcal{V}}{dt} = 0 \tag{49}$$

This is achieved by zeroing 8 equations: the $\dot{y}_0, \dot{z}_0$ equations from (7) and the righthand sides of the 6 equations of motion in (23). With u given, there are 12 parameters available for achieving the desired equilibrium. These are listed in Table 4 along with the equations they predominantly influence. With more parameters than equations, there are options for achieving equilibrium; however, a constraint on this is that, for each degree-of-freedom, there must be as many parameters available as equations that need to be zeroed. Thus, RPM must be used to zero the X equation, either y_G or ϕ can be used to zero the K equation while the other can be set arbitrarily, and v, ψ and δ_r are all needed to zero the horizontal plane equations. There are several options for the vertical plane equations but the common ones are based on either gravitational (m, x_G, θ) or planes $(\delta_b, \delta_s, \theta)$ control. Of course, the equations in Table 4 are coupled so that, in some cases (always when $\phi \neq 0$), a simultaneous numerical solution of these nonlinear equations is required to obtain solutions for the trim parameters.

Degree of Freedom	Dominant Equations	Dominant Parameters
Longitudinal	X	RPM
Roll	K	y_G, ϕ
Vertical Plane	$Z, M, \dot{z}_0$	$m, x_G, w, \theta, \delta_b, \delta_s$
Horizontal Plane	$Y, N, \dot{y}_0$	v, ψ, δ_r

Table 4 *The 8 equations that must be zeroed to establish a steady state self-propulsive state and the 12 trim parameters available to achieve this.*

The v, w parameters in Table 4 are hard to work with operationally because the vehicle cannot measure them directly. By measuring pressure (depth) and pitch angle θ, w can be estimated. However, there is no general way for the vehicle to measure v with onboard sensors. Nevertheless, it is desirable that the simulation, at least, be initialized with the values of these parameters known and fixed.

The trim method used depends on a vehicle's capabilities and operational requirements. We consider two generic methods, one for submarines which have mass and mass distribution control capabilities, and one for UUVs which do not. UUV trimming is complicated when there is some minimal level of reserve buoyancy because this severely limits low speed controllability.

Submarine Trim

Submarines generally achieve equilibrium while underway by trimming their mass and mass distribution to minimize pitch, roll, and plane deflection angles. This minimizes drag and is based on easily measured quantities. However, it uses gravitational forces, which are constant, to compensate for hydrodynamic force asymmetries (as would be generated by the sail and deck) which vary with u^2, so this trim method is speed dependent. Gravity cannot be used to trim in the horizontal plane which, fortunately, is not usually required for a submarine that has a vertical plane of symmetry, as most do. When horizontal plane trimming is required, hydrodynamic forces must be used and these must be some combination of rudder deflection and sideslip.

We assume a submarine trims by setting $\phi, \theta, \delta_b, \delta_s = 0$ and that the remaining parameters from Table 4 are used to zero the necessary 8 equations. With ϕ and θ both zero, (7c) requires that $w = 0$ as well. For the submarine of Appendix A, which has a vertical plane of symmetry ($Y_{uu}, N_{uu} = 0$) containing an axially aligned propeller ($y_P, \psi_P, \theta_P = 0$), the Y, N and $\dot{y}_0$ equations give $v, \delta_r, \psi = 0$. This means that the assumptions in (47) are correct for this submarine and, from (48), that $J_s = J_{sn}$; that is, the self-propulsion point for this submarine properly trimmed is the same as its nominal self-propulsion point and is speed invariant. Therefore, from (42):

$$RPM = \frac{60(1 - w_T)u}{J_{sn}D}. \tag{50}$$

The Z equation for this submarine determines the trimmed dimensionless mass m'_t:

$$0 = (m'_t - 1)\rho g V + Z_{uu}u^2. \tag{51}$$

This requires that $m'_t = 1$ since DSSP estimates that $Z_{uu} = 0$. Similarly, the M equation gives:

$$x_G = x_B + \frac{M_{uu}u^2 + z_P X_P}{\rho g V} \tag{52}$$

though z_P is usually zero. The K equation is used to find that y_G value that counteracts propeller torque:

$$\begin{aligned} y_G &= y_B - \frac{K_P}{\rho g V} \\ &= y_B - s_K \frac{\pi D^3}{8gV}\left[V_A^2 + (0.7\pi n D)^2\right] C_Q(\beta, P/D) \\ &= y_B - s_K \frac{\pi D^3}{8gV}(1 - w_T)^2 u^2 \left[1 + \left(\frac{0.7\pi}{J_s}\right)^2\right] C_Q\left(\tan^{-1}\frac{J_s}{0.7\pi}, P/D\right) \end{aligned} \tag{53}$$

Thus, x_G and y_G vary quadratically with speed. Instead of using y_G, some submarines negate propeller torque by building preswirl into their tailplanes, which is equivalent to a δ_ϕ trim value.

UUV Trim

UUVs generally do not have fine control over mass or mass distribution while underway. They are usually trimmed manually while sitting in the water just at the surface. Their CG is adjusted to zero the roll and pitch angles and their mass is usually adjusted so they are slightly buoyant, all at zero speed. In addition, many UUVs do not have foreplanes, as is the case for the UUV described in Appendix B. Therefore, the following trim analysis assumes:

$$m = m'_t \rho V, \quad x_G = x_B, \quad y_G = y_B, \quad \text{and} \quad \delta_b \equiv 0 \tag{54}$$

where m'_t is specified. To simplify the presentation of the expressions below, we take advantage of the following characteristics of the Appendix B UUV:

$$\begin{aligned} &K_{uu0}, K_{uv}, K_{vw}, K_{v\nu}, K_{uu\delta_s 0}, K_{uu\delta_r 0}, M_{uu}, M_{uu\delta_r\delta_r 0}, M_{u|w|}, M_{vv}, M_{|w\nu|}, \\ &\quad N_{uu}, N_{uu\delta_s\delta_s 0}, N_{vw}, Y_{uu}, Y_{uu\delta_s}, Y_{vw}, Z_{uu}, Z_{u|w|}, Z_{vv}, Z_{|w\nu|}, y_P, z_P, \theta_P, \psi_P = 0. \end{aligned} \tag{55}$$

The 6 equations of motion become:

$$\begin{aligned} 0 = {} & (1 - m'_t)\rho g V \sin\theta + \left(X_{uu} + X_{uu\delta_s\delta_s 0}\delta_s^2 + X_{uu\delta_r\delta_r 0}\delta_r^2\right) u^2 + X_{vv0} v^2 + X_{ww0} w^2 \\ & + X_P \end{aligned} \tag{56a}$$

$$0 = (m'_t - 1)\rho g V \sin\phi\cos\theta + Y_{uu\delta_r 0}\delta_r u^2 + Y_{uv0} uv + Y_{v\nu 0} v\sqrt{v^2 + w^2} \tag{56b}$$

$$0 = (m'_t - 1)\rho g V \cos\phi\cos\theta + Z_{uu\delta_s}\delta_s u^2 + Z_{uw0} uw + Z_{w\nu 0} w\sqrt{v^2 + w^2} \tag{56c}$$

$$0 = \left[(z_B - m'_t z_G)\sin\phi - y_B(1 - m'_t)\cos\phi\right]\rho g V \cos\theta + K_P \tag{56d}$$

$$\begin{aligned} 0 = {} & \left((z_B - m'_t z_G)\sin\theta + x_B(1 - m'_t)\cos\phi\cos\theta\right)\rho g V + M_{uu\delta_s 0}\delta_s u^2 + M_{uw0} uw \\ & + M_{w\nu 0} w\sqrt{v^2 + w^2} \end{aligned} \tag{56e}$$

$$\begin{aligned} 0 = {} & (m'_t - 1)\left(x_B \sin\phi\cos\theta + y_B \sin\theta\right)\rho g V + N_{uu\delta_r 0}\delta_r u^2 + N_{uv0} uv \\ & + N_{v\nu 0} v\sqrt{v^2 + w^2}. \end{aligned} \tag{56f}$$

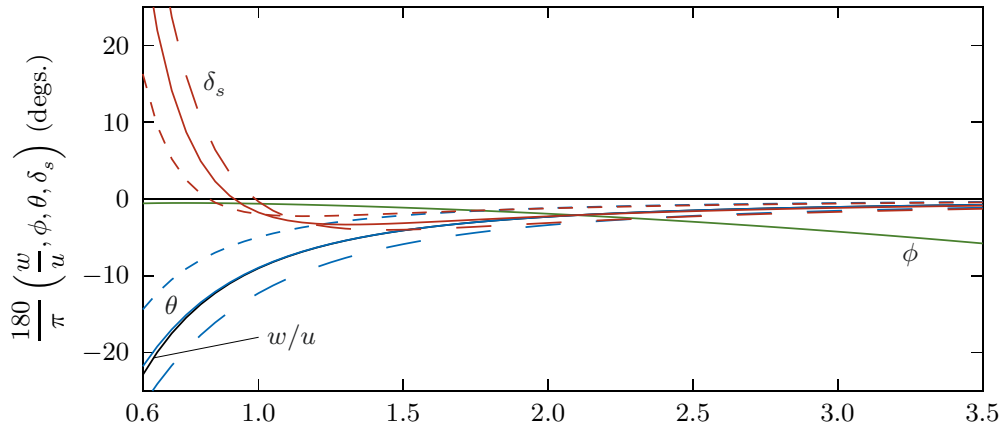
In these equations, X_P and K_P are replaced with:

$$X_P = (1 - t_D)\frac{\rho\pi D^2}{8}\left[(1 - w_T)^2 u^2 + (0.7\pi n D)^2\right] C_T\left(\tan^{-1}\frac{(1 - w_T)u}{0.7\pi n D}, P/D\right) \tag{57a}$$

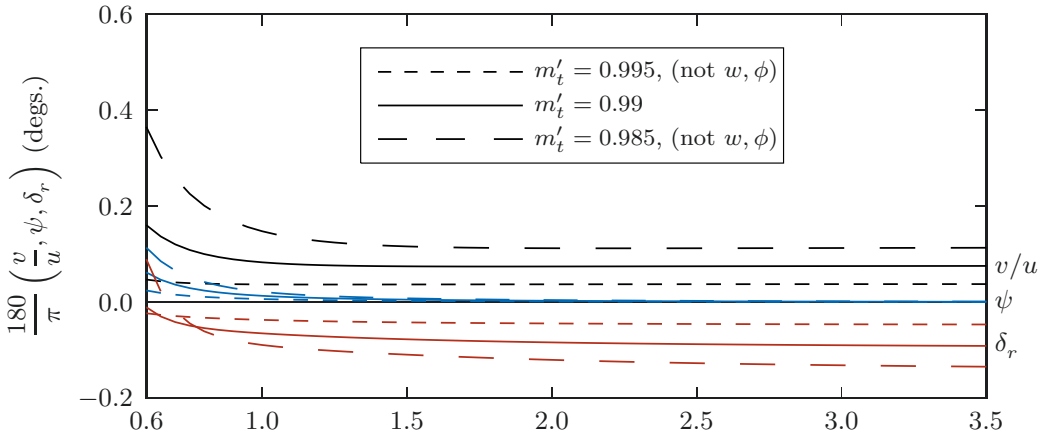
$$K_P = s_K\frac{\rho\pi D^3}{8}\left[(1 - w_T)^2 u^2 + (0.7\pi n D)^2\right] C_Q\left(\tan^{-1}\frac{(1 - w_T)u}{0.7\pi n D}, P/D\right) \tag{57b}$$

so that the 8 equations (56) and (7b & c) are expressed in terms of 8 unknowns v, w, ϕ, θ, ψ, δ_s, δ_r, and n. However, explicit expressions for δ_s, δ_r, and ψ can be obtained from (56b & c) and (7b). These are substituted into the remaining equations leaving 5 nonlinear coupled equations to be solved numerically for the unknowns v, w, ϕ, θ, and n. The solutions are then substituted back into the explicit expressions for δ_s, δ_r, and ψ.

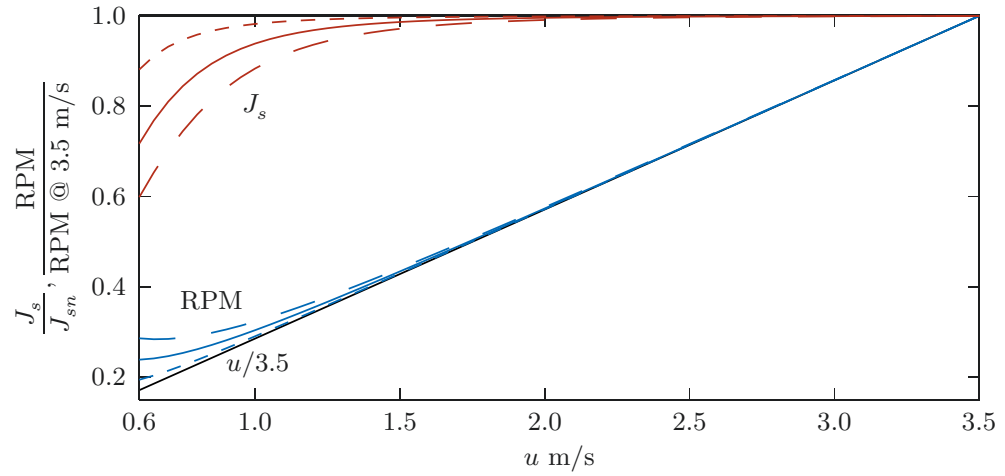
The resulting trim levels are plotted in Figure 6 and tabulated in Appendix B. They show that low levels of reserve buoyancy cause the UUV to loose control at speeds below about 0.7 m/s where the sternplanes approach their limits; lower speeds are possible for lower reserve buoyancies. These trim levels are accurate within the context of the current model. But, when



a) *The $w, \phi, \theta, \delta_s$ trim levels. The w/u curve (black) approximates θ; ϕ is insensitive to m'_t.*



b) *The v, ψ, δ_r trim levels.*



c) *Normalized J_s and RPM as a function of speed.*

Figure 6 *The accurate self-propulsion states for the UUV of Appendix B as a function of speed for net buoyancies of 0.5, 1.0, and 1.5% of the vehicle buoyancy. When $m'_t = 1.0$, the v, w, θ, ψ, δ_s, and δ_r trim levels are zero, $J_s = J_{sn}$, and normalized RPM is the line $u/3.5$.*

assessing how realistic they are, keep in mind that the large sternplane angles predicted at low speed do not account for the hydrodynamic limits on sternplane effectiveness caused by stall nor, in (56) at least, do they account for the changing inflow to the sternplanes due to increasing incidence w/u.

The Figure 6a behavior is complex. Equations (56) contain the full nonlinear model that accounts for the dependence of the critical point (32) on incidence as well as speed. At high speed, the critical point is well forward of the CB. Therefore the upwards force from the reserve buoyancy pitches the nose down which helps the UUV to maintain depth. However, the net pitch down moment is too large and must be moderated with a slightly negative sternplane deflection which also helps to negate the buoyancy. As the hydrodynamic forces decrease with decreasing speed, the reserve buoyancy and pitch restoring moment remain constant, so increasingly large incidence and sternplane deflection angles are needed. But, as incidence increases and speed continues to decrease, the critical point moves aft and crosses the CB, at $u \approx 0.9$ m/s. The impact of the reserve buoyancy on the moment balance reverses causing the moment compensation provided by the sternplanes to reverse as well. At still lower speeds, the critical point continues to move aft increasing the now unfavorable moment from the reserve buoyancy, decreasing the sternplanes moment arm, and now augmenting the reserve buoyancy with the ever increasing sternplane normal force, the total of which must be negated by incidence. The incidence and sternplane deflection angles increase quickly and become nonsensical as the sternplane critical velocity is approached. Equilibrium cannot be maintained.

Similar but much reduced effects are seen on v/u and δ_r reflecting the extent to which gravitational force and moment components impact these states at these small roll angles. The heading ψ deviates from zero to keep the vehicle net velocity on a zero degree heading; since ψ is a purely horizontal plane measurement, it must account for both v and w crossflow contributions when the vehicle is rolled.

Finally, as the hull incidence increases and the planes deflect, drag increases shifting the propeller operating point to lower less efficient advance ratios.

8 Dynamic Change Model

Underwater vehicles are controlled by physically adjusting mechanical devices such as control surfaces and propellers. Changes to these mechanisms are implemented through actuators and controllers of varying complexity and do not happen instantaneously. Following Campbell and Graham [20] and Watt [5], these changes are modelled using a generic linear, second order, ordinary differential equation (ODE):

$$\ddot{\delta} + 2\zeta\omega\dot{\delta} + \omega^2\delta = \omega^2\delta_c \tag{58}$$

where $\delta(t)$ is the physical quantity being changed with time, δ_c is the 'commanded' value to which δ is being changed, ζ is the system damping ($0 < \zeta < 1$), and ω is the system response frequency ($\omega > 0$). The general solution to (58) and its time derivatives are:

$$\delta(t) = \delta_c - \alpha\, e^{-\zeta\omega(t-t_0)} \sin\left(\sqrt{1-\zeta^2}\,\omega(t-t_0) + \beta\right) \tag{59a}$$

$$\dot{\delta}(t) = \omega\alpha\, e^{-\zeta\omega(t-t_0)} \sin\left(\sqrt{1-\zeta^2}\,\omega(t-t_0) + \beta - \cos^{-1}\zeta\right) \tag{59b}$$

$$\ddot{\delta}(t) = -\omega^2\alpha\, e^{-\zeta\omega(t-t_0)} \sin\left(\sqrt{1-\zeta^2}\,\omega(t-t_0) + \beta - 2\cos^{-1}\zeta\right). \tag{59c}$$

where t_0 is the time at which the new command δ_c is issued, and α and β are the constants of integration. The time derivatives in (59) are obtained using the fact that $\sqrt{1-\zeta^2} = \sin(\cos^{-1}\zeta)$.

The following initial conditions determine α and β and ensure C^1 continuity in $\delta(t)$ whenever a new command is issued:

$$\begin{aligned} \delta_0 \equiv \delta(t_0) &= \delta_c - \alpha\sin\beta \qquad &\Rightarrow \quad \alpha_1 &= \frac{\delta_c - \delta_0}{\sin\beta} \\ \dot{\delta}_0 \equiv \dot{\delta}(t_0) &= \omega\alpha\sin(\beta - \cos^{-1}\zeta) \quad &\Rightarrow \quad \alpha_2 &= \frac{\dot{\delta}_0}{\omega\sin(\beta - \cos^{-1}\zeta)} \end{aligned} \tag{60}$$

The phase shift β is found by setting $\alpha_1 = \alpha_2$. The problem is that these α expressions are indeterminant when $\delta_c = \delta_0 \Rightarrow \beta = 0, \pi$ or $\dot{\delta}_0 = 0 \Rightarrow \beta = \cos^{-1}\zeta$. Therefore, the solution is formulated two different ways depending on the initial condition values. To help with this define:

$$g \equiv \frac{2\zeta\dot{\delta}_0}{\omega(\delta_c - \delta_0)} \quad \text{and} \quad h \equiv \frac{1}{g} \tag{61}$$

and set:

$$\alpha = \begin{cases} \alpha_1 & \text{for } -1 \le g \le 1 \\ \alpha_2 & \text{otherwise (ie, for } -1 < h < 1). \end{cases} \tag{62}$$

Replacing the ratio $\dot{\delta}_0/(\delta_c - \delta_0)$ in the equation $\alpha_1 = \alpha_2$ with either g or h, and choosing a solution branch in which β is the principle value of $\cos^{-1}\zeta$ when $\dot{\delta}_0 = 0$, results in:

$$\beta = \cos^{-1}\left(\frac{2\zeta^2 - g}{\sqrt{g^2 + 4\zeta^2(1-g)}}\right) = \cos^{-1}\left(\frac{\text{sign}(h)(2\zeta^2 h - 1)}{\sqrt{1 + 4\zeta^2 h(h-1)}}\right). \tag{63}$$

These expressions are plotted in Figure 7 and give rise to Figure 8 which shows that:

$$0 \le \beta \le \pi. \tag{64}$$

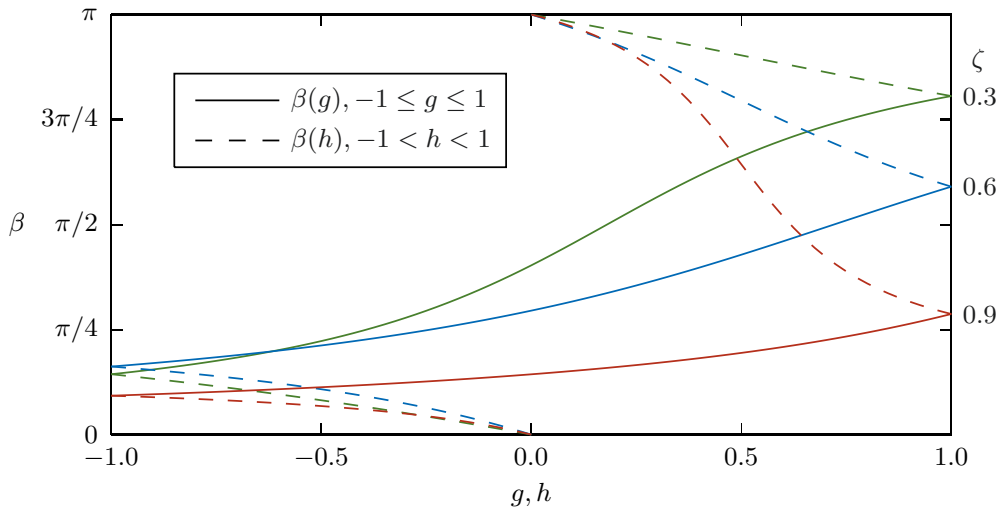


Figure 7 *The phase β of the solution $\delta(t)$ and its variation with g and h.*

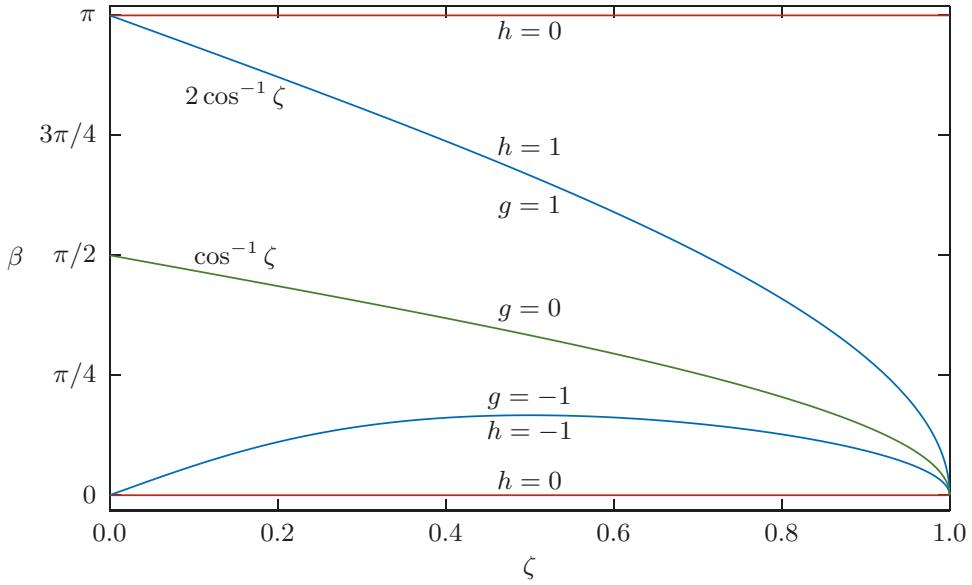


Figure 8 *The phase β of the solution $\delta(t)$ and its variation with ζ.*

From (60) and (56), $g = 2\zeta \sin(\beta - \cos^{-1}\zeta)/\sin\beta$ which expands to:

$$\left(2\zeta^2 - g\right)\sin\beta = 2\zeta\sqrt{1-\zeta^2}\cos\beta. \tag{65}$$

The coefficients here are the sides of a right-triangle with an hypotenuse that is the denominator of $\cos\beta(g)$ from (63):

$$\sqrt{g^2 + 4\zeta^2(1-g)} = \sqrt{\left(2\zeta^2 - g\right)^2 + \left(2\zeta\sqrt{1-\zeta^2}\right)^2}.$$

That is, $g^2 + 4\zeta^2(1-g) > 0$ because it is the sum of squares over the range of its parameters. The same holds for the denominator of $\cos\beta(h)$.

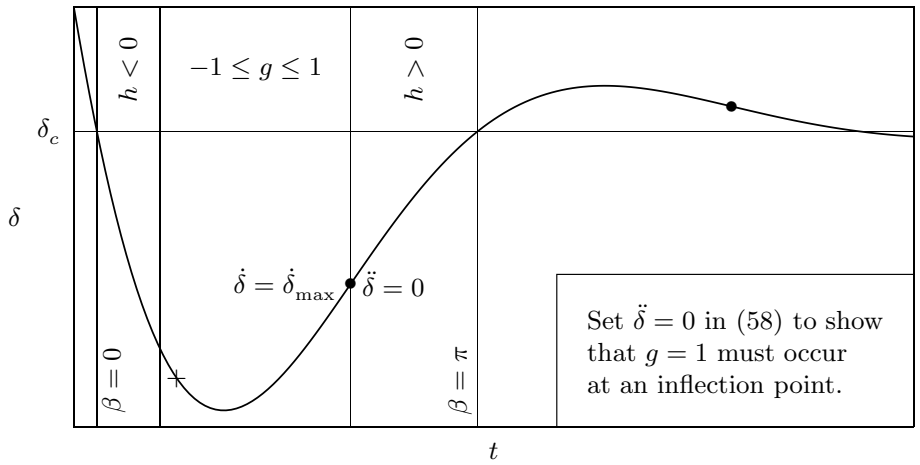


Figure 9 *A $\delta(t)$ solution with $\zeta = 0.5$ and with initial conditions somewhere between $\beta = 0$ and π. The rate $\dot{\delta}$ has extrema at the inflection points $\ddot{\delta} = 0$, shown as solid circles, but their magnitudes decrease with time because of damping. Therefore, there is a region in time prior to each inflection point where the rate magnitude is greater than at the inflection point itself. As shown in the text, this occurs if t_0 precedes the point '+' in the figure or is located in the region $h > 0$, in which case the maximum rate is just $\dot{\delta}_0$.*

Limiting the Rate

Most dynamical systems have a rate limit. This means most systems have three characteristic parameters:

- ζ System damping, assumed sub-critical: $0 < \zeta < 1$. The lower the damping the faster the system achieves δ_c, but at the expense of overshoot. A good default value is $\zeta = 0.9$.
- ω The natural response frequency, the frequency at which the system responds.
- $\dot{\delta}_{\mathrm{RL}}$ The rate limit $\dot{\delta}_{\mathrm{RL}} > 0$, the magnitude of the maximum rate at which the system can respond.

Of course, $\dot{\delta}_{\mathrm{RL}}$ does not appear in (58) so the ODE needs to be modified in some way to accommodate the rate limit.

In [5], Watt implements the rate limit by first using ω as is and checking to see if the maximum rate, $\dot{\delta}_{\max}$ say, has a magnitude below the rate limit. If true, the solution is implemented as is. If false, a 'rate limited' solution is devised by reducing ω until $|\dot{\delta}_{\max}| = \dot{\delta}_{\mathrm{RL}}$. This provides a response that is perfectly (C^∞) continuous between command changes, but it also slows the response unnaturally.

DSSP uses (58) for its dynamic change model but implements the rate limit differently because it integrates the ODE numerically in order to implement continuous command changes from the autopilots. DSSP always uses the same ω but numerically caps the rate when the limit is exceeded. In what follows, we devise an improved analytic method to do the same.

Consider Figure 9 which displays a lightly damped $\delta(t)$ response to better illustrate overshoot. The maximum rate in this response is either the initial condition itself $\dot{\delta}_0$ or it is the rate at the first inflection point where $\ddot{\delta} = 0$. This suggests there are limits on when a response may need to be rate limited, and knowing these limits simplifies the rate limiting process.

Denote inflection point times as $t = t_i$ and set (59c) to zero get the t_i:

$$\sqrt{1-\zeta^2}\omega(t_i - t_0) = 2\cos^{-1}\zeta - \beta + n\pi, \quad n = 0, 1, 2, \ldots \tag{66}$$

Since $t_i > t_0$, then $n \geq 0$. Using (65) this gives:

$$\sin\left(\sqrt{1-\zeta^2}\omega(t_i - t_0)\right) = (-1)^n(1-g)\sin\beta \tag{67a}$$

$$\sin\left(\sqrt{1-\zeta^2}\omega(t_i - t_0) + \beta\right) = (-1)^n 2\zeta\sqrt{1-\zeta^2} \tag{67b}$$

$$\sin\left(\sqrt{1-\zeta^2}\omega(t_i - t_0) + \beta - \cos^{-1}\zeta\right) = (-1)^n\sqrt{1-\zeta^2} \tag{67c}$$

The first inflection point occurs in the first half period of the response when each side of (67a) must be positive. It is selected with $n = 0$ when $g \leq 1$ and $n = 1$ when $g > 1$, so when $g = 1$ the first inflection point is $t_i = t_0$.

The rate will not need to be limited if $|\dot{\delta}_0| \geq |\dot{\delta}(t_i)|$ because the initial condition $\dot{\delta}_0$ should itself be rate limited. From (59b) and (67c) this condition becomes:

$$|\dot{\delta}_0| \geq \left|\omega\frac{\delta_c - \delta_0}{\sin\beta}\sqrt{1-\zeta^2}\right| e^{-\zeta\omega(t_i - t_0)}. \tag{68}$$

Squaring both sides and using (63) and (65) gives:

$$g^2 \geq \left(g^2 + 4\zeta^2(1-g)\right) e^{\frac{-2\zeta}{\sqrt{1-\zeta^2}}\cos^{-1}\left(\frac{2\zeta^2(1-g)+g}{\sqrt{g^2+4\zeta^2(1-g)}}\right)}. \tag{69}$$

Since the exponential term here is positive and less than or equal to one, it is also true that the response will not need to be rate limited if:

$$g^2 \geq g^2 + 4\zeta^2(1-g) \quad \Rightarrow \quad g \geq 1. \tag{70}$$

Therefore all responses that might need to be rate limited fall within the $n = 0$ branch of (66). Furthermore, when $g = 1$ the exponential term in (69) is exactly one. That is, $g = 1$ is precisely the upper limit for a possible rate limited response and it occurs when t_0 is an inflection point and $\dot{\delta}_0$ and $\delta_c - \delta_0$ have the same sign. If $g = 1$ and $|\dot{\delta}_0| = \dot{\delta}_{\mathrm{RL}}$, then the analytic solution (59) applies and is perfectly continuous. A response only needs to be 'rate limited' when $g < 1$.

Figure 9 shows that there should be a lower limit as well, when $g, h < 0$. Finding this limit requires a numerical solution of (69). This is shown as the lower g limit in Figure 10, which also shows the upper limit and the region containing all rate-limited solutions. The lower limit lacks the simplicity of the upper limit; it is lowest when $\zeta \to 1$ where $g = -0.771867$.

In summary, in a rate limited environment, a necessary but insufficient condition for the maximum rate of the linear model (59) to exceed the rate limit is:

$$-0.772 < g < 1, \quad 0 < \zeta \leq 1. \tag{71}$$

Over this range, $n = 0$ in (66) locates the first inflection point in the response and the rate there:

$$\dot{\delta}_{\max} = \dot{\delta}(t_i) = \frac{\omega(\delta_c - \delta_0)}{2\zeta}\sqrt{g^2 + 4\zeta^2(1-g)}\, e^{\frac{-\zeta}{\sqrt{1-\zeta^2}}\cos^{-1}\left(\frac{2\zeta^2(1-g)+g}{\sqrt{g^2+4\zeta^2(1-g)}}\right)} \tag{72a}$$

$$= \frac{\omega(\delta_c - \delta_0)\sqrt{1-\zeta^2}}{\sin\beta}\, e^{\frac{-\zeta}{\sqrt{1-\zeta^2}}\left(2\cos^{-1}\zeta - \beta\right)} \tag{72b}$$

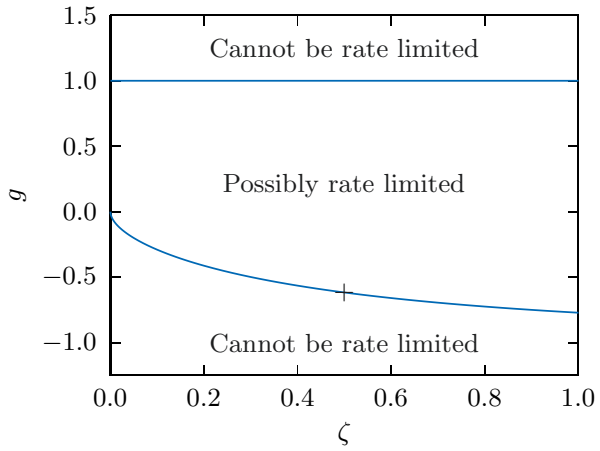


Figure 10 *The curves in g, ζ space delimiting the extent of all possible rate limited responses. The '+' symbol corresponds to that in Figure 9.*

can be compared with $\dot{\delta}_{\rm RL}$ to see if it needs to be limited. These expressions show that this rate will always have the same sign as $\delta_c - \delta_0$.

If $|\dot{\delta}_{\rm max}| > \dot{\delta}_{\rm RL}$, then the rate needs to be limited. This is done, as it is done numerically by DSSP, by limiting the rate magnitude to $\dot{\delta}_{\rm RL}$ until (58) can complete the response normally. This takes two steps. First, the normal response of (58) must be stopped prior to $t = t_i$, at $t = t_1$ say, such that $\dot{\delta}_1 \equiv \dot{\delta}(t_1) = \text{sign}(\delta_c - \delta_0)\dot{\delta}_{\rm RL}$, after which $\dot{\delta}(t) = \dot{\delta}_1 =$ constant and $\ddot{\delta} = 0$. This maintains only C^1 continuity at $t = t_1$ which can be a problem for high order ODE integrators (it isn't a problem at $t = t_0$ because the integrator must stop there anyway to implement the new command δ_c).

The second step is determining when to terminate the constant $\dot{\delta}$ ramp and switch back to (58). This happens when the magnitude of $\ddot{\delta}(t)$, which is being driven by $\delta_c - \delta(t)$ in (58), reduces to zero. If this happens at $t = t_2$ say, then (58) gives:

$$\delta_2 \equiv \delta(t_2) = \delta_c - \frac{2\zeta}{\omega}\,\text{sign}(\delta_c - \delta_0)\dot{\delta}_{\rm RL} \tag{73}$$

which, of course, corresponds to $g = 1$ for the final leg of the response. Knowing δ_2, t_2 is easily found from the constant rate between t_1 and t_2:

$$t_2 = t_1 + \frac{\delta_2 - \delta_1}{\dot{\delta}_1}. \tag{74}$$

Because δ, $\dot{\delta}$, and $\ddot{\delta}$ are all matched at $t = t_2$, the response is C^2 continuous there:

$$\delta(t) = \delta_c - \frac{\delta_c - \delta_2}{2\zeta\sqrt{1-\zeta^2}}\, e^{-\zeta\omega(t-t_2)} \sin\left(\sqrt{1-\zeta^2}\omega(t-t_2) + 2\cos^{-1}\zeta\right) \quad \text{for } t \geq t_2. \tag{75}$$

The only difficulty in all this is finding t_1 which requires a trial and error solution. However, this can be done efficiently and reliably using a pure Newton method which finds the zero in $f(t)$ by following the local slope:

$$0 = f(t_j) + \frac{df}{dt}(t_j)(t_{j+1} - t_j) \quad \Rightarrow \quad \Delta t \equiv t_{j+1} - t_j = \frac{-f(t_j)}{df/dt(t_j)} \tag{76}$$

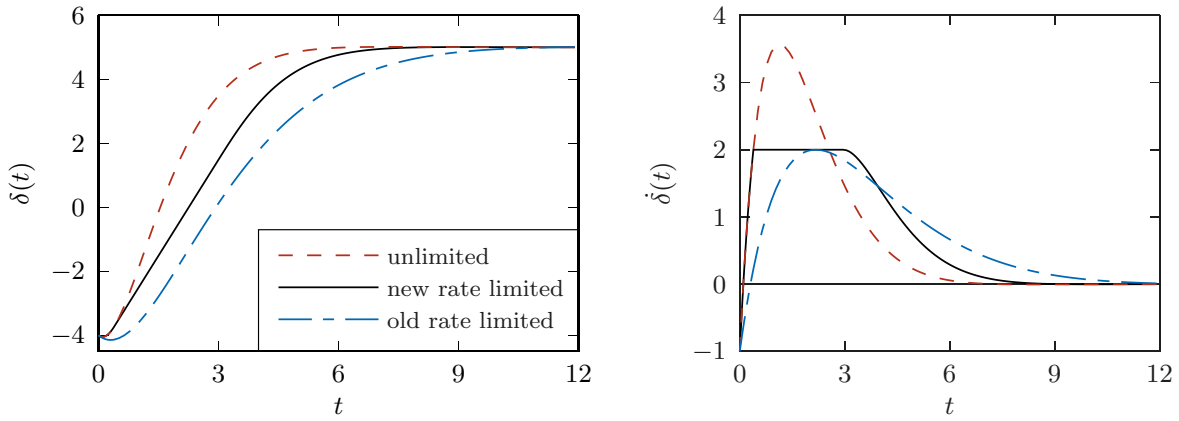


Figure 11 *The response $\delta(t)$ and its rate $\dot{\delta}(t)$ for a system with $\zeta = 0.9$ and $\omega = 1$ with initial conditions $\delta(0) = -4$ and $\dot{\delta}(0) = -1$. The rate limit is $\dot{\delta}_{\rm RL} = 2$. The new rate limited method applies the method discussed herein. The old method avoids discontinuities by reducing the frequency (to $\omega = 0.55$ in this case) which slows the response.*

From (59), set $f(t_j) = \dot{\delta}(t_j) - \dot{\delta}_1$ so that:

$$\frac{-f(t_j)}{df/dt(t_j)} = \frac{\sin\left(\sqrt{1-\zeta^2}\omega(t_j - t_0) + \beta - \cos^{-1}\zeta\right) - \dfrac{\dot{\delta}_1 \sin\beta\, e^{\zeta\omega(t_j - t_0)}}{\omega(\delta_c - \delta_0)}}{\omega \sin\left(\sqrt{1-\zeta^2}\omega(t_j - t_0) + \beta - 2\cos^{-1}\zeta\right)} \tag{77}$$

A good initial guess for t_1 is t_0. As long as the initial guess comes before t_1 and after the extremum in $\ddot{\delta}(t)$ immediately preceding t_0, the Newton iteration will march monotonically towards the correct solution, and usually get there with 10 digits of accuracy in under 10 iterations.

This new method of rate limiting the response is compared to the old method [5] in Figure 11. The new method is recommended as it avoids an unrealistic time lag and it matches the numerical scheme in use by DSSP. The only disadvantage of the new method is the discontinuities it introduces in the response. However, once the response has been calculated, the locations of the discontinuities are known and can be used to avoid numerical inefficiency in any higher level numerical integration using the response. The `dc3Module` algorithm discussed below returns the times at which discontinuities occur when it processes a change in command.

Limiting the Amplitude

Many systems operate with limits on the amplitude of the response. For example, control surfaces on submarines all have natural limits to their deflection amplitudes and, in addition, can have temporary stops put in place to prevent dangerous deflection levels at high speeds.

When applying limits, it helps to know the response extrema. The extrema occur when $\dot{\delta}(t) = 0$, at $t = t_e$ say, when (from (59b)):

$$\sqrt{1-\zeta^2}\omega(t_e - t_0) = \cos^{-1}\zeta - \beta + m\pi, \quad m = 0, 1, 2, \ldots \tag{78}$$

The only extrema of interest are when $t_e > t_0$ which means $m \geq 0$. Set $t_{e0} \equiv t_e$ when $m = 0$ and $t_{e1} \equiv t_e$ when $m = 1$. Then $t_{e0} > t_0$ only when $g, h < 0$ (Figure 8) which means the t_{e0} extremum occurs only if the response has to reverse direction before heading towards δ_c

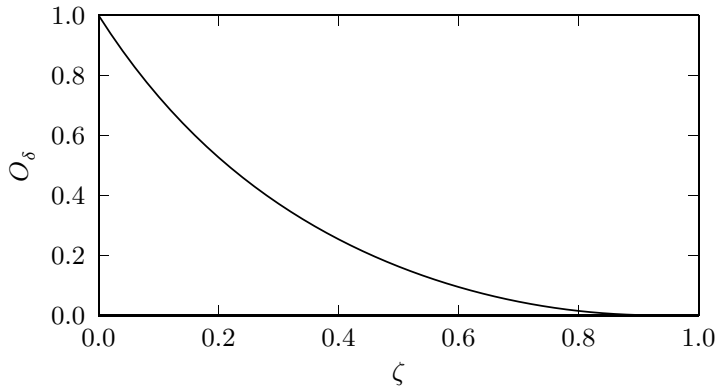


Figure 12 *Overshoot in the response as a function of* ζ.

(Figure 9). And t_{e1} gives the extremum in the first overshoot of δ_c regardless of the values of g, h. Substituting (78) and:

$$\sin\left(\sqrt{1-\zeta^2}\omega(t_e - t_0) + \beta\right) = (-1)^m\sqrt{1-\zeta^2} \tag{79}$$

into (59a) gives the response extrema. If $g, h < 0$, two extrema need to be checked against the limits, otherwise only one. All subsequent extrema ($m > 1$) will be less extreme because of damping.

The `dc3Module` algorithm allows the user to apply three kinds of limits: soft limits, hard limits, and limits that are simultaneously soft and hard.

Soft Limits: DSSP

DSSP uses only soft limits which are easily implemented and introduce no additional discontinuities. A soft limit simply reduces the magnitude of the command δ_c before applying it so that the system is not asked to exceed a soft limit. Using our under damped response model, the limit will still be temporarily exceeded due to overshoot, but the overshoot is usually small for typical damping levels. Following Campbell and Graham [20], overshoot is defined as:

$$O_\delta \equiv \frac{\delta_{e1} - \delta_c}{\delta_c - \delta_{e0}} = e^{-\zeta\pi/\sqrt{1-\zeta^2}} \tag{80}$$

and is obtained by substituting (78) and (79) into (59b). In other words, overshoot gives the amount that δ_c is exceeded as a fraction of the desired change from the previous extremum. Overshoot is plotted in Figure 12. For $\zeta > 0.8$, it is less than 1.52% of $\delta_c - \delta_{e0}$.

This overshoot estimate is based on the response (59a). If the rate has been limited then the denominator in (80) is too small and overshoot, measured as a fraction of the change in δ between the extrema preceding and following the constant rate ramp, will be even smaller than that given by (80).

Consider, for example, a typical submarine sternplane maximum deflection scenario using initial conditions $\delta_0, \dot{\delta}_0 = 0$ and system parameters $\zeta = 0.9$, $\omega = 2$ rad/s, $\dot{\delta}_{\max} = 7$ deg/s. Commanding the change $\delta_c = 25$ degrees results in a rate limited change of 7 deg/s between $\delta = 0.3$ and 18.7 degrees. The overshoot is 0.0135 degrees, about 0.05% of the commanded change. This is about the same as the default integration error DSSP uses.

Hard Limits

Hard limits alone do not limit δ_c. They can be applied by brute force using:

$$\delta_{\text{limited}} = \min(\text{upper limit}, \max(\text{lower limit}, \delta(t))) \tag{81}$$

where $\delta(t)$ is the response (59a) based on δ_c. This is probably incorrect physically as undershoot will follow overshoot if $\delta_c \approx \delta_{\text{limited}}$, and discontinuities occur every time the oscillating response crosses the limit. A good alternative relies on the physical argument that once the response has been limited, undershoot will not occur, if δ_c equals or exceeds the limit, because there is no impetus for it to do so. In other words, if a response is limited to $\delta = \delta_\ell$ at $t = t_\ell$ say, and δ_c is equal to or exceeds the limit, then the response is simply $\delta(t) = \delta_\ell$ for $t > t_\ell$. But if δ_c is within the limit, and it was overshoot that caused the limit to be hit, then the subsequent response should be determined by (58) with new initial conditions $\delta(t_\ell) = \delta_\ell$ and $\dot{\delta}(t_\ell) = 0$.

Equations (78) and (79) can be used to determine if an extremum has exceeded a limit. If it has, the time t_ℓ at which the limit is first exceeded must be found. This is easy to do in the special case that $\delta_c = \delta_\ell$ because then, from (59b), $\sin\left(\sqrt{1-\zeta^2}\omega(t_\ell - t_0) + \beta\right) = 0$. This gives an analytic solution for t_ℓ:

$$t_\ell = t_0 + \frac{\pi - \beta}{\omega\sqrt{1-\zeta^2}} \quad \text{when } \delta_c = \delta_\ell. \tag{82}$$

Otherwise a pure Newton method can again be used to get a fast and efficient numerical solution for t_ℓ. Similar to (76), if $F(t) \equiv \delta(t) - \delta_\ell$, then we solve $F(t_\ell) = 0$ by iterating t_j towards t_ℓ using:

$$\Delta t = \frac{-F(t_j)}{dF/dt(t_j)} = \frac{\sin\left(\sqrt{1-\zeta^2}\omega(t_j - t_0) + \beta\right) - \dfrac{\delta_c - \delta_\ell}{\alpha}\, e^{\zeta\omega(t_j - t_0)}}{\omega \sin\left(\sqrt{1-\zeta^2}\omega(t_j - t_0) + \beta - \cos^{-1}\zeta\right)}. \tag{83}$$

A good initial guess for finding a limit preceding t_{e0} is t_0; for t_{e1} use the inflection point between t_{e0} and t_{e1}:

$$t_i = t_{e1} - \frac{\pi - \cos^{-1}\zeta}{\omega\sqrt{1-\zeta^2}}. \tag{84}$$

These initial guesses guarantee fast monotonic convergence.

Since hard limits alone do not limit δ_c, finding t_ℓ can be complicated because the hard limit can be anywhere between the extrema of the unlimited response. No problem arises if the response is not rate limited: the initial guess is still the t_i between extrema and this works fine with (83) whether t_ℓ precedes or follows t_i. However, if the response is rate limited, then (83) is different for each segment of the response; this means the hard limit must be looked for separately between t_{e0} and t_1 (use t_1 as the initial guess), t_1 and t_2 (there is an analytic solution), and t_2 and t_{e1} (use t_2 as the initial guess).

The `dc3module` algorithm allows soft and hard limits to be applied simultaneously. The next example shows the difference between using just hard and both hard and soft limits.

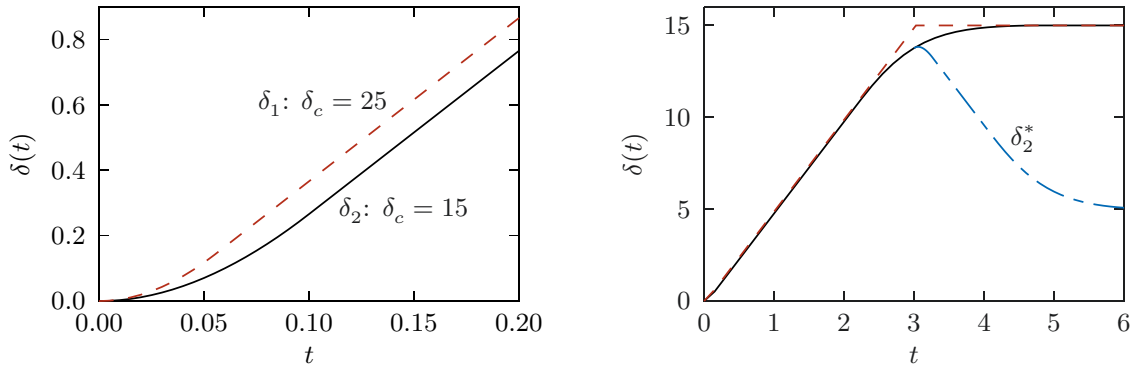


Figure 13 *Two $\delta(t)$ responses each with a hard limit $\delta_\ell = 15$. The $\delta_1(t)$ response uses $\delta_c = 25$ and $\delta_2(t)$ uses $\delta_c = 15$. In each case, $\delta_0 = \dot{\delta}_0 = 0$, $\zeta = 0.9$, $\omega = 2$ rad/s, and $\dot{\delta}_{\rm RL} = 5$ δ units/s. The δ_2^* off-shoot from δ_2 shows how a new command $\delta_c = 5$ issued at $t = 3$ modifies the response while maintaining C^1 continuity.*

An Analytic Dynamic Change Algorithm

The `dc3Module` algorithm listed in Appendix D implements the above dynamic change model, accommodating soft and/or hard limits whether rate limited or not. Figure 13 shows two responses predicted by this algorithm, each with the same hard limit. The soft limit for δ_2 has been set to match the hard limit, but not so for δ_1. The δ_2 response is slightly slower because the $\delta_c - \delta(t)$ term driving (58) is smaller, but when the limit is hit the discontinuity is less severe.

The `dc3Module` is a procedure that returns a Maple 'module'. It can be called any number of times creating as many different dynamically changing systems as are being simultaneously modelled, and returns separate, independent modules each time. All system parameters are contained within a module, some of which are exported and can therefore be accessed by the user. The $\delta(t)$, $\dot{\delta}(t)$, and $\ddot{\delta}(t)$ responses are all exported.

For example, after the `dc3Module` script is read in by Maple, the two systems generating the responses in Figure 13 are generated with the `dcMake(`ζ`,`ω`,`$\dot{\delta}_{\rm RL}$`,options)` command:

```
del1 := dcMake(0.9,2,5,dMaxS=25,dMaxH=15);  # Creates module del1
del2 := dcMake(0.9,2,5,dMaxSH=15);
```

which initializes a response $\delta(t) = 0$ for all time for each system. This is modified with `command(`t_0`,`δ_c`)`:

```
del1:-command(0,25);
del2:-command(0,25);
```

Each `command` call returns the discontinuities in the response after $t = t_0$. The `del2` response is accessed with `del2:-delta(t)`:

$$\delta_2(t) = \begin{cases} 0 & t < 0 \\ 15 - 34.412\, e^{7.2252-1.8t} \sin(0.87178t + 0.45103) & t < 0.099871 \\ 5t - 0.23385 & t < 2.1468 \\ 15 - 5.7354\, e^{3.8642-1.8t} \sin(0.87178t - 0.96946) & t < 4.7157 \\ 15 & \text{otherwise.} \end{cases} \tag{85}$$

Of course, the discontinuities also show up as the break points in this piecewise function. A subsequent `del2` command:

```
del2:-command(3,5)
```

modifies the `del2` response after $t = 3$ giving the δ_2^* curve in Figure 13:

$$\delta_2^*(t) = \begin{cases} 0 & t < 0 \\ 15 - 34.412\, e^{7.2252-1.8t} \sin(0.87178t + 0.45103) & t < 0.099871 \\ 5t - 0.23385 & t < 2.1468 \\ 15 - 5.7354\, e^{3.8642-1.8t} \sin(0.87178t - 0.96946) & t < 3 \\ 5 + 22.518\, e^{5.4-1.8t} \sin(0.87178t - 2.2153) & t < 3.2647 \\ 29.570 - 5t & t < 4.0140 \\ 5 + 5.7354\, e^{7.2252-1.8t} \sin(0.87178t - 2.5973) & \text{otherwise.} \end{cases} \tag{86}$$

Any number of additional commands can be given creating a long piecewise response. There is an option to remember only the new response if the response history is not required.

A Numerical Rate Limited Dynamic Change Algorithm

DSSP simulates underwater vehicles by numerically integrating the nonlinear ordinary differential equations describing their motion. Several vehicle system models are also based on ODEs, including the autopilots. It is convenient, then, to integrate all the ODEs in parallel, including the dynamic change ODE (58). This is handled by converting (58) to two first order ODEs. Define:

$$\begin{aligned} y_1(t) &\equiv \delta(t) \\ y_2(t) &\equiv \dot{\delta}(t) = y_1'(t) \\ y_2'(t) &\equiv \ddot{\delta}(t) = \omega^2 \left(\delta_c - y_1(t)\right) - 2\zeta\omega y_2(t) \end{aligned} \tag{87}$$

A numerical ODE integrator works by repeatedly calling a function like `F` shown here. The values $t, y_1(t), y_2(t)$ are supplied and the derivatives $y_1'(t), y_2'(t)$ are returned:

```
F(T,Y,YP)
    # T is time; Y,YP are vectors of length 2
    global δc, ζ, ω, δ̇RL
    local a,s
    a = ω [ω(δc − Y[1]) − 2ζY[2]]     # δ̈ when δ is not rate limited
    if δ̇RL > |Y[2]| then
        # Path A: Not rate limited
        YP[1] = Y[2]
        YP[2] = a
    else
        # Path B: Rate limited
        s = SIGN(1,Y[2])
        YP[1] = s*δ̇RL
        if s*a < 0 then
            # Path B1:  No longer needs to be rate limited
            # because a will decrease the rate magnitude
            YP[2] = a
        else
            # Path B2:  a is trying to increase the rate magnitude
            YP[2] = 0
        end if
    end if
end F
```

This is the best way to numerically implement and exit the rate limited state. It ensures there is C^2 continuity when exiting.

There is no good way to impose hard limits on $\delta(t)$ through this function because it is not permissible for `F` to change `Y`. A kludge is to set `YP[1]` $= 0$ when the limit is first exceeded which will limit δ to δ_ℓ within integration error, but it does not allow $\dot{\delta}$ to change. This can be fixed by assigning `YP[2]` a temporary large magnitude with a sign that drives `Y[2]` to zero, but this adds a second discontinuity when `YP[2]` is reset when `Y[2]` reaches zero. The best way to impose hard limits is to stop and restart the integration when the limit is first exceeded, using suitable boundary conditions.

9 Concluding Remarks

The above algorithms have been supplied to the contractor Dynamic Systems Analysis Ltd. (DSA) for use in the UUV Docking simulator they are developing for DRDC. This has been done using earlier drafts of this report. Feedback from DSA has improved these algorithms and is incorporated in the final version of this report. The UUV Docking simulator itself was nearing completion as this report was finalized.

Acknowledgement

The author would like to thank André Roy of Dynamic Systems Analysis Ltd. for his valuable feedback during the development of this document.

References

1. G.D. Watt, J.A. Carretero, R. Dubay, and M.R. MacKenzie, *Towards an Automated Active UUV Dock on a Slowly Moving Submarine*, P RINA Warship 2011: Naval Submarines and UUVs, Bath, June 2011.

2. G.D. Watt, A.R. Roy, J. Currie, C.B. Gillis, J. Giesbrecht, G.J Heard, M. Birsan, M.L. Seto, J.A. Carretero, R. Dubay, and T.L. Jeans, *A Concept for Docking a UUV With a Slowly Moving Submarine Under Waves*, IEEE J of Ocean Engineering, vol 41, no 2, April 2016.

3. Dynamic Systems Analysis Ltd., *Dynamic simulation of the automated docking of a UUV to a slowly moving submarine in littoral conditions; Part 1: Sensor and control system modelling; Part 2: Modelling the vehicles and the dock; Part 3: Software Manual*, DRDC Atlantic Contract Report DRDC-RDDC-2017-C247, October 2017.

4. K.A. McTaggart, *Verification and Validation of ShipMo3D Ship Motion Predictions in the Time and Frequency Domains*, ITTC Workshop on Seakeeping, Seoul, October 2010.

5. G.D. Watt, Modelling Submarine Control Surface Deflection Dynamics, DREA TM 90/203, March 1990.

6. T.I. Fossen, *Guidance and Control of Ocean Vehicles*, John Wiley & Sons, 1994.

7. M. Gertler and G.R. Hagen, *Standard Equations of Motion for Submarine Simulation*, Naval Ship Research and Development Center (NSRDC) report 2510, June 1967.

8. J. Feldman, *DTNSRDC Revised Standard Submarine Equations of Motion*, David W. Taylor Naval Ship Research and Development Center, DTNSRDC/SPD-0393/09, June 1979.

9. G.D. Watt, *Modelling and Simulating Unsteady Six Degrees-of-Freedom Submarine Rising Maneuvers*, DRDC Atlantic TR 2007-008, February 2007.

10. G.D. Watt, *Estimates for the Added Mass of a Multi-Component Deeply Submerged Vehicle Part I: Theory and Program Description*, DREA TM 88/213, October 1988.

11. F.H. Imlay, *Complete Expressions for the Gravitational and Buoyancy Force Terms in the Equations of Motion of a Submerged Body*, David Taylor Model Basin Report 1845, July 1964.

12. M. Mackay, *DSSP50 Build 090910 Documentation, Parts 1, 2, and 3*, DRDC Atlantic TMs 2009-226, 2009-227, 2009-228, December 2009.

13. H. Lamb, *Hydrodynamics*, 6^{th} edition, Dover Publications, 1945.

14. F.H. Imlay, *The Complete Expressions for "Added Mass" for a Rigid Body Moving in an Ideal Fluid*, David Taylor Model Basin Report 1528, July 1961.

15. J.N. Newman, *Marine Hydrodynamics*, MIT Press, 1977.

16. W.E. Cummins, *The impulse response function and ship motions*, David Taylor Model Basin Technical Report 1661, 1962.

17. W.P.A. van Lammeren, J.D. van Manen, and M.W.C. Oosterveld, *The Wageningen B-Screw Series*, SNAME Transactions, vol 77, pp 269-317, 1969.

18. J.P. Comstock, Editor, *Principles of Naval Architecture*, Society of Naval Architects and Marine Engineers, revised edition, December 1970.

19. S. Toxopeus, F. Quadvlieg, and M. Kerkvliet, *SHWG Collaborative Exercise: Part 6: Generic Submarine Hydrodynamics BB2,* September 2017.

20. W. Campbell and R. Graham, private communication, June 1988.

21. G.D. Watt, *BB3: A Generic BB2 Based Submarine Design Using Sternplanes, Rudder, and Bowplanes,* DRDC Reference Document DRDC-RDDC-2019-D135, October 2019.

Appendix A: BB3 Data

This appendix lists the file `bb3Data.ini` generated for the BB3 submarine [21] used by the DSA UUV docking simulator [3]. The format was defined by DSA and allows this data to be read directly into their simulation.

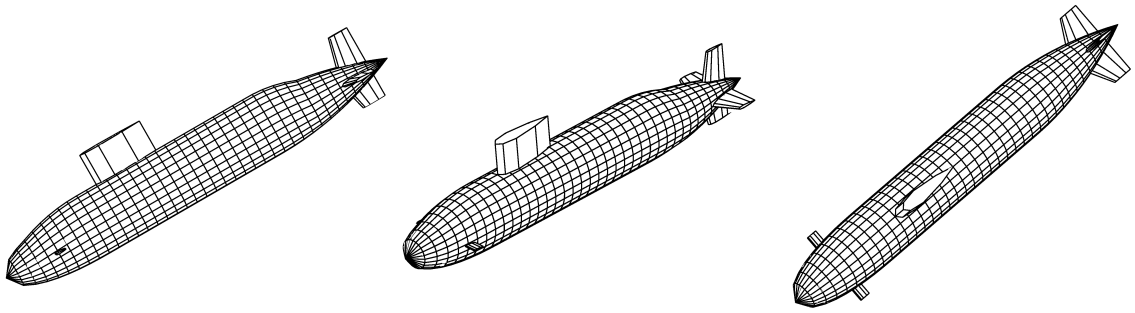


Figure A1 *The BB3 geometry with bowplanes, symmetrical sternplanes, and an asymmetrical rudder, as modelled by DSSP.*

```
// bb3Data.ini written for vehicle "bb3" on 2019-03-19 16:52:40

//////////////////////////
// Miscellaneous Constants
//
// Fluid density, Gravitational constant, SI units
$rho 1028.0
$g 9.81
//
// Conversions for degrees to radians, knots to m/s
$d2r .1745329e-1
$k2mps .5144444

////////////////////////////
// Vehicle Geometry and Mass
//
// Hull length, diameter, volume in m, m, m^3
$ell 70.20000
$dee 9.600000
$vol 4364.328
//
// Body axes origin relative to vehicle nose, m
$xRef -32.34300
$yRef 0.
$zRef 0.
//
// Centers of buoyancy and z-component of gravity in body axes, m
$xB -.84988e-1
$yB 0.
$zB -.582790
```

```
$zG -.182790
//
// Moments of Inertia in body axes, SI units normalized by rho
$Ix 58912.13
$Iy 1153880.
$Iz 1260327.
$Ixy -0.
$Ixz -5785.048
$Iyz -0.

////////////////////////////////////////
// Initial Equilibrium Trim for Simulation
// - for starting a simulation with the vehicle in perfect trim.
//
// In all cases: u is given; dx_0/dt, dy_0/dt, p, q, r = 0
//
// Need: v, w, phi, theta, psi, mtp, xG, yG, delta_b, delta_s, delta_r, rpm
// where mtp = m/(rho*vol)
//
// There are 3 equilibrium modes, determined by the value of iniMode:
//
// iniMode = 1 ==> Generally a submarine; it can trim mass and
//             mass distribution:
//             w, phi, theta, delta_b, delta_s = 0;
//             rpm is linear in u, as given by propulsion coefficients;
//             mtp, xG, yG,, v, delta_r have the form:
//                 p = p0 + p2*u^2  and  p0, p2 are given herein;
//             psi = -arctan(v/u).
//
// iniMode = 2 ==> Generally a UUV without foreplanes; it has been
//             manually trimmed at zero forward speed:
//             mtp = constant (given herein), xG = xB, yG = yB;
//             v, w, phi, theta, psi, delta_s, delta_r, rpm
//             are tabulated as a function of u.
//
// iniMode = 3 ==> Generally a UUV with foreplanes; it has been
//             manually trimmed at zero forward speed:
//             w = 0, mtp = constant (given herein), xG = xB, yG = yB;
//             v, phi, theta, psi, delta_b, delta_s, delta_r, rpm
//             are tabulated as a function of u.
$iniMode 1
// Parameter form: p = p0 + p2*u^2
//                 psi = -arctan(v/u)
$mtp0 1.
$mtp2 0.
$xG0 -.84988e-1
$xG2 .4093456e-4
$yG0 0.
$yG2 .5505476e-4
$v0 0.
```

```
$v2 0.
$delta_r0 0.
$delta_r2 0.

////////////////////////////
// Control Surface Parameters
//
// Number of control surfaces, virtual depth control weights
$NCS 6
$kDb -1.000000
$kDs 1.000000
// CS1
$iCS 1
$dsspCmpnt 2
$type SBOWPLANE
$dsspLabel #STBDBOWPLANE
$esamLabel STBD BOWPLN
$deltaMin -20.00000    // degrees
$deltaMax 20.00000    // degrees
$deldotMax 5.000000   // degrees/s
$zeta .900000
$omega 2.500000       // radians/s
$kdb 1.000000
$kdr 0.
$kds 0.
$kdphi 0.
$CprFlag false
// CS2
$iCS 2
$dsspCmpnt 3
$type SBOWPLANE
$dsspLabel #PORTBOWPLANE
$esamLabel PORT BOWPLN
$deltaMin -20.00000    // degrees
$deltaMax 20.00000    // degrees
$deldotMax 5.000000   // degrees/s
$zeta .900000
$omega 2.500000       // radians/s
$kdb -1.000000
$kdr 0.
$kds 0.
$kdphi 0.
$CprFlag false
// CS3
$iCS 3
$dsspCmpnt 4
$type RUDDER
$dsspLabel #TOPRUDDER
$esamLabel TOP RUDDER
$deltaMin -30.00000    // degrees
```

```
$deltaMax 30.00000    // degrees
$deldotMax 6.000000   // degrees/s
$zeta .900000
$omega 2.000000       // radians/s
$kdb 0.
$kdr -1.000000
$kds 0.
$kdphi -0.
$CprFlag false
// CS4
$iCS 4
$dsspCmpnt 5
$type RUDDER
$dsspLabel #BOTRUDDER
$esamLabel BOT RUDDER
$deltaMin -30.00000    // degrees
$deltaMax 30.00000    // degrees
$deldotMax 6.000000   // degrees/s
$zeta .900000
$omega 2.000000       // radians/s
$kdb 0.
$kdr 1.000000
$kds 0.
$kdphi -0.
$CprFlag false
// CS5
$iCS 5
$dsspCmpnt 6
$type STERNPLANE
$dsspLabel #STBDSTERNPLANE
$esamLabel STBD STRNPLN
$deltaMin -25.00000    // degrees
$deltaMax 25.00000    // degrees
$deldotMax 6.000000   // degrees/s
$zeta .900000
$omega 2.000000       // radians/s
$kdb 0.
$kdr 0.
$kds 1.000000
$kdphi -0.
$CprFlag true
  $u0 1.460000   // m/s
  $u1 1.660000   // m/s
  $u2 2.060000   // m/s
  $u3 2.260000   // m/s
  $g0 -.500000
  $g1 0.
  $g2 0.
  $g3 1.000000
// CS6
```

```
$iCS 6
$dsspCmpnt 7
$type STERNPLANE
$dsspLabel #PORTSTERNPLANE
$esamLabel PORT STRNPLN
$deltaMin -25.00000    // degrees
$deltaMax 25.00000    // degrees
$deldotMax 6.000000   // degrees/s
$zeta .900000
$omega 2.000000       // radians/s
$kdb 0.
$kdr 0.
$kds -1.000000
$kdphi -0.
$CprFlag true
  $u0 1.460000   // m/s
  $u1 1.660000   // m/s
  $u2 2.060000   // m/s
  $u3 2.260000   // m/s
  $g0 -.500000
  $g1 0.
  $g2 0.
  $g3 1.000000

//////////////////////////////////////////
// Vehicle Total and Component Added Masses
// - SI units normalized by rho
//
// Xudot = Xudot
$XudotHull -161.8639
$XudotSail -14.61194
$XudotCS1 -.9382121e-1
$XudotCS2 -.9382121e-1
$XudotCS3 -1.206910
$XudotCS4 -.5910010
$XudotCS5 -1.159550
$XudotCS6 -1.159550
$Xudot -180.7805
// Xvdot = Yudot
$XvdotHull 0.
$XvdotSail 0.
$XvdotCS1 .2013420e-1
$XvdotCS2 -.2013420e-1
$XvdotCS3 0.
$XvdotCS4 0.
$XvdotCS5 -.3722405
$XvdotCS6 .3722405
$Xvdot 0.
// Xwdot = Zudot
$XwdotHull 0.
```

```
$XwdotSail -.8309681
$XwdotCS1 .2852345e-1
$XwdotCS2 .2852345e-1
$XwdotCS3 .3253299
$XwdotCS4 -.2071885
$XwdotCS5 0.
$XwdotCS6 0.
$Xwdot -.6557625
// Xpdot = Kudot
$XpdotHull 0.
$XpdotSail 0.
$XpdotCS1 .9471376e-1
$XpdotCS2 -.9471376e-1
$XpdotCS3 0.
$XpdotCS4 0.
$XpdotCS5 0.
$XpdotCS6 0.
$Xpdot 0.
// Xqdot = Mudot
$XqdotHull 65.79692
$XqdotSail 93.19834
$XqdotCS1 -.6095680
$XqdotCS2 -.6095680
$XqdotCS3 14.57135
$XqdotCS4 -7.868527
$XqdotCS5 0.
$XqdotCS6 0.
$Xqdot 164.4777
// Xrdot = Nudot
$XrdotHull 0.
$XrdotSail 0.
$XrdotCS1 .6593534
$XrdotCS2 -.6593534
$XrdotCS3 0.
$XrdotCS4 0.
$XrdotCS5 14.96234
$XrdotCS6 -14.96234
$Xrdot 0.
// Yvdot = Yvdot
$YvdotHull -4267.064
$YvdotSail -428.2660
$YvdotCS1 -.2336881e-1
$YvdotCS2 -.2336881e-1
$YvdotCS3 -73.07088
$YvdotCS4 -38.88529
$YvdotCS5 -.6924330
$YvdotCS6 -.6924330
$Yvdot -4808.718
// Ywdot = Zvdot
$YwdotHull 0.
```

```
$YwdotSail 0.
$YwdotCS1 .3310207
$YwdotCS2 -.3310207
$YwdotCS3 0.
$YwdotCS4 0.
$YwdotCS5 0.
$YwdotCS6 0.
$Ywdot 0.
// Ypdot = Kvdot
$YpdotHull -1734.540
$YpdotSail -2638.679
$YpdotCS1 1.111065
$YpdotCS2 1.111065
$YpdotCS3 -273.2516
$YpdotCS4 100.5144
$YpdotCS5 0.
$YpdotCS6 0.
$Ypdot -4543.733
// Yqdot = Mvdot
$YqdotHull 0.
$YqdotSail 0.
$YqdotCS1 -7.223745
$YqdotCS2 7.223745
$YqdotCS3 0.
$YqdotCS4 0.
$YqdotCS5 0.
$YqdotCS6 0.
$Yqdot 0.
// Yrdot = Nvdot
$YrdotHull 237.6538
$YrdotSail -2863.063
$YrdotCS1 -.5573540
$YrdotCS2 -.5573540
$YrdotCS3 2309.116
$YrdotCS4 1219.537
$YrdotCS5 22.99844
$YrdotCS6 22.99844
$Yrdot 948.1272
// Zwdot = Zwdot
$ZwdotHull -3746.704
$ZwdotSail -11.61807
$ZwdotCS1 -8.682959
$ZwdotCS2 -8.682959
$ZwdotCS3 -.5213443
$ZwdotCS4 -.4966089
$ZwdotCS5 -74.75030
$ZwdotCS6 -74.75030
$Zwdot -3926.207
// Zpdot = Kwdot
$ZpdotHull 0.
```

```
$ZpdotSail 0.
$ZpdotCS1 -29.23376
$ZpdotCS2 29.23376
$ZpdotCS3 0.
$ZpdotCS4 0.
$ZpdotCS5 -231.1781
$ZpdotCS6 231.1781
$Zpdot 0.
// Zqdot = Mwdot
$ZqdotHull -208.6726
$ZqdotSail 83.72210
$ZqdotCS1 189.4227
$ZqdotCS2 189.4227
$ZqdotCS3 -17.83290
$ZqdotCS4 -16.27498
$ZqdotCS5 -2336.269
$ZqdotCS6 -2336.269
$Zqdot -4452.751
// Zrdot = Nwdot
$ZrdotHull 0.
$ZrdotSail 0.
$ZrdotCS1 7.093817
$ZrdotCS2 -7.093817
$ZrdotCS3 0.
$ZrdotCS4 0.
$ZrdotCS5 0.
$ZrdotCS6 0.
$Zrdot 0.
// Kpdot = Kpdot
$KpdotHull -911.4968
$KpdotSail -16493.72
$KpdotCS1 -99.22213
$KpdotCS2 -99.22213
$KpdotCS3 -1104.656
$KpdotCS4 -268.3430
$KpdotCS5 -760.1915
$KpdotCS6 -760.1915
$Kpdot -20497.13
// Kqdot = Mpdot
$KqdotHull 0.
$KqdotSail 0.
$KqdotCS1 637.7835
$KqdotCS2 -637.7835
$KqdotCS3 0.
$KqdotCS4 0.
$KqdotCS5 -7227.872
$KqdotCS6 7227.872
$Kqdot 0.
// Krdot = Npdot
$KrdotHull 96.57962
```

```
$KrdotSail -17640.23
$KrdotCS1 23.78262
$KrdotCS2 23.78262
$KrdotCS3 8642.043
$KrdotCS4 -3151.411
$KrdotCS5 0.
$KrdotCS6 0.
$Krdot -12005.37
// Mqdot = Mqdot
$MqdotHull -855259.1
$MqdotSail -1206.521
$MqdotCS1 -4132.721
$MqdotCS2 -4132.721
$MqdotCS3 -622.6103
$MqdotCS4 -535.7484
$MqdotCS5 -73054.77
$MqdotCS6 -73054.77
$Mqdot -1011999.
// Mrdot = Nqdot
$MrdotHull 0.
$MrdotSail 0.
$MrdotCS1 -154.8002
$MrdotCS2 154.8002
$MrdotCS3 0.
$MrdotCS4 0.
$MrdotCS5 0.
$MrdotCS6 0.
$Mrdot 0.
// Nrdot = Nrdot
$NrdotHull -978122.6
$NrdotSail -20190.00
$NrdotCS1 -13.72402
$NrdotCS2 -13.72402
$NrdotCS3 -72999.70
$NrdotCS4 -38263.51
$NrdotCS5 -771.0173
$NrdotCS6 -771.0173
$Nrdot -1111145.

//////////////////////
// Viscous Coefficients
// - SI units normalized by rho
//
// X
$Xuu -2.998094
$Xuudbdb -8.510144
$Xuudrdr0 -3.532005
$Xuudsds0 -30.53768
$Xvv0 63.27067
$Xvr -1118.232
```

```
$Xww0 46.81362
$Xwq 1431.581
$Xpr 2421.114
$Xqq 35525.31
$Xrr 26011.55
// Y
$Yuu -0.
$Yuudb -0.
$Yuudr0 34.06934
$Yuuds -0.
$Yuv0 -174.5281
$Yup -908.7873
$Yur0 1129.912
$Yu1r1dr 643.7188
$Yvw -289.3298
$Yvq -2725.600
$Yvnu0 -299.4753
$Yvnu1r1v1 -1867.867
$Ywp -1998.587
$Ywr -1336.190
$Ypq -18118.38
$Yp1p1 -576.8796
$Yqr -13569.23
$Yr1r1 104735.0
// Z
$Zuu -0.
$Zuudb -12.05286
$Zuuds0 -35.82177
$Zuw0 -110.1265
$Zuq0 -1330.125
$Zu1w1 0.
$Zu1q1ds -368.4676
$Zvv 293.3478
$Zvp 2234.833
$Zvr -1769.272
$Zwnu0 -200.8512
$Zwnu1q1w1 -2539.178
$Zpp 0.
$Zpr -22158.22
$Zq1q1 -91942.86
$Zrr -15553.66
$Z1wnu1 -0.
// K
$Kuu0 0.
$Kuudb -0.
$Kuudr0 57.12006
$Kuuds0 -0.
$Kuv -598.7583
$Kup -6437.645
$Kur 143.9649
```

```
$Kvw 0.
$Kvq 3246.557
$Kvnu -373.0479
$Kwp 0.
$Kwr 990.0016
$Kpq 25278.63
$Kp1p1 -3252.849
$Kqr 3641.554
// M
$Muu 1.752575
$Muudb 283.2878
$Muudrdr0 5.121939
$Muuds0 -1226.617
$Muw0 1516.574
$Muq0 -87623.75
$Mu1w1 0.
$Mu1q1ds -12617.15
$Mvv 3208.703
$Mvp 23864.00
$Mvr -44725.28
$Mwnu0 -2353.824
$Mpp -2018.540
$Mpr -458956.9
$Mqnu -38581.92
$Mq1q1 -2739043.
$Mrr -345206.1
$M1wnu1 -0.
// N
$Nuu 0.
$Nuudbdb 0.
$Nuudr0 -1063.975
$Nuudsds0 -0.
$Nuv0 -2730.439
$Nup -5622.743
$Nur0 -64493.04
$Nu1r1dr -20106.27
$Nvw 3362.495
$Nvq 68954.71
$Nvnu0 2271.919
$Nwp 22492.11
$Nwr 15528.76
$Npq 478233.6
$Nqr 370452.3
$Nrnu -23822.74
$Nr1r1 -2384569.

//////////////////////
// Propulsion Parameters
//
//  Geometry m, degrees
```

```
$DP 5.000000
$PoD .83
$xP -36.51300
$yP 0.
$zP 0.
$psiP 0.
$thetaP 0.
// wake fraction, thrust deduction, K sign from prop handedness
$wT .313423
$tD .77199e-1
$sK -1
// Control: max rpm, max drpm/dt, damping, response frequency (rad/s)
$rpmMax 125.
$rpmdotMax 10.
$zetaP .900000
$omegaP .8000000
// rpm = [r1mps*u | r1kts*U], u in m/s, U in knots, nominally
// Jsn = J at nominal self-propulsion
// - These formulas are exact only if iniMode = 1 and the vehicle is in
//   perfect equilibrium as defined in "Initial Equilibrium" above.
// - If iniMode = 2,3, these formulae are approximately correct at
//   moderate to high speeds; see tabulated rpm vs u data above for a
//   more accurate relationship.
$r1mps 12.55247
$r1kts 6.457548
$Jsn .6563585

//////////////////////
// AutoDepth and AutoHeading parameters
// - see Section 11 of DSSP50 documentation, Part 2, Input Reference
//
// AutoDepth
$thetaL 10.00000
$ellz 60.00000
$DCC .100000
$DCE 25.00000
$Dk1 .500000
$Dk2 1.000000
$DKP 1.000000
$DKI 0.
$DKD 1.000000
// AutoHeading
$HCC .50000e-1
$HCE 30.00000
$Hk1 .500000
$Hk2 .100000
$HKP 1.000000
$HKI 0.
$HKD 1.000000
```

Appendix B: UUV Data

This appendix lists the file `uuvData.ini` generated for the DSA UUV docking simulator [3]. The format was defined by DSA and allows this data to be read directly into their simulation.

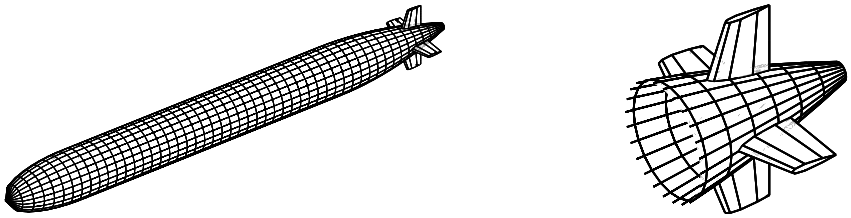


Figure B1 *The UUV geometry consisting of a hull and symmetrical tail, as modelled by DSSP.*

```
// uuvData.ini written for vehicle "uuv" on 2019-03-20 09:32:57

//////////////////////////
// Miscellaneous Constants
//
// Fluid density, Gravitational constant, SI units
$rho 1028.0
$g 9.81
//
// Conversions for degrees to radians, knots to m/s
$d2r .1745329e-1
$k2mps .5144444

////////////////////////////
// Vehicle Geometry and Mass
//
// Hull length, diameter, volume in m, m, m^3
$ell 3.200000
$dee .320000
$vol .2168123
//
// Body axes origin relative to vehicle nose, m
$xRef -1.501000
$yRef 0.
$zRef 0.
//
// Centers of buoyancy and z-component of gravity in body axes, m
$xB -.5333e-2
$yB 0.
$zB 0.
$zG .15000e-1
//
// Moments of Inertia in body axes, SI units normalized by rho
$Ix .2602067e-2
$Iy .1384583
$Iz .1384583
$Ixy -0.
$Ixz -0.
$Iyz -0.

////////////////////////////////////////
// Initial Equilibrium Trim for Simulation
// - for starting a simulation with the vehicle in perfect trim.
//
// In all cases: u is given; dx_0/dt, dy_0/dt, p, q, r = 0
//
```

```
// Need: v, w, phi, theta, psi, mtp, xG, yG, delta_b, delta_s, delta_r, rpm
// where mtp = m/(rho*vol)
//
// There are 3 equilibrium modes, determined by the value of iniMode:
//
// iniMode = 1 ==> Generally a submarine; it can trim mass and
//             mass distribution:
//             w, phi, theta, delta_b, delta_s = 0;
//             rpm is linear in u, as given by propulsion coefficients;
//             mtp, xG, yG,, v, delta_r have the form:
//                 p = p0 + p2*u^2  and  p0, p2 are given herein;
//             psi = -arctan(v/u).
//
// iniMode = 2 ==> Generally a UUV without foreplanes; it has been
//             manually trimmed at zero forward speed:
//             mtp = constant (given herein), xG = xB, yG = yB;
//             v, w, phi, theta, psi, delta_s, delta_r, rpm
//             are tabulated as a function of u.
//
// iniMode = 3 ==> Generally a UUV with foreplanes; it has been
//             manually trimmed at zero forward speed:
//             w = 0, mtp = constant (given herein), xG = xB, yG = yB;
//             v, phi, theta, psi, delta_b, delta_s, delta_r, rpm
//             are tabulated as a function of u.
$iniMode 2
$mtp .99
// Tabulated states as a function of u:
// u(m/s) rpm v(m/s) w(m/s) phi(rad) theta(rad) psi(rad) delta_s(rad) delta_r(rad)
 .60 237.8728 .1683744e-2 -.2399276 -.9939949e-2 -.3804094 .1085072e-2 .5818385 -.1803713e-3
 .65 240.5814 .1554728e-2 -.2263248 -.9360203e-2 -.3350696 .8190815e-3 .3831701 -.5172143e-3
 .70 245.6828 .1473315e-2 -.2137353 -.9084892e-2 -.2963510 .6400836e-3 .2465686 -.7203089e-3
 .75 252.5960 .1425430e-2 -.2021517 -.9030388e-2 -.2632846 .5150975e-3 .1519244 -.8504777e-3
 .80 260.9013 .1401932e-2 -.1915283 -.9144506e-2 -.2349931 .4249091e-3 .8596421e-1 -.9393840e-3
 .85 270.2901 .1396659e-2 -.1817991 -.9393393e-2 -.2107124 .3578913e-3 .3982028e-1 -.1004123e-2
 .90 280.5317 .1405308e-2 -.1728892 -.9754258e-2 -.1897930 .3067659e-3 .7500308e-2 -.1054204e-2
 .95 291.4512 .1424787e-2 -.1647226 -.1021123e-1 -.1716913 .2668377e-3 -.1508942e-1 -.1095045e-2
1.00 302.9141 .1452811e-2 -.1572265 -.1075291e-1 -.1559562 .2349947e-3 -.3077534e-1 -.1129799e-2
1.05 314.8160 .1487659e-2 -.1503331 -.1137087e-1 -.1422146 .2091229e-3 -.4152889e-1 -.1160339e-2
1.10 327.0751 .1528008e-2 -.1439809 -.1205875e-1 -.1301589 .1877535e-3 -.4873848e-1 -.1187796e-2
1.15 339.6268 .1572830e-2 -.1381146 -.1281164e-1 -.1195347 .1698445e-3 -.5339084e-1 -.1212872e-2
1.20 352.4196 .1621314e-2 -.1326849 -.1362567e-1 -.1101315 .1546420e-3 -.5619310e-1 -.1236013e-2
1.25 365.4124 .1672814e-2 -.1276482 -.1449776e-1 -.1017744 .1415906e-3 -.5765591e-1 -.1257516e-2
1.30 378.5720 .1726814e-2 -.1229657 -.1542542e-1 -.9431754e-1 .1302743e-3 -.5815052e-1 -.1277584e-2
1.35 391.8712 .1782894e-2 -.1186031 -.1640661e-1 -.8763893e-1 .1203763e-3 -.5794844e-1 -.1296367e-2
1.40 405.2882 .1840713e-2 -.1145302 -.1743967e-1 -.8163594e-1 .1116516e-3 -.5724922e-1 -.1313978e-2
1.45 418.8049 .1899991e-2 -.1107201 -.1852317e-1 -.7622193e-1 .1039085e-3 -.5620014e-1 -.1330512e-2
1.50 432.4065 .1960499e-2 -.1071491 -.1965596e-1 -.7132343e-1 .9699459e-4 -.5491016e-1 -.1346046e-2
1.55 446.0805 .2022046e-2 -.1037959 -.2083704e-1 -.6687787e-1 .9078753e-4 -.5346008e-1 -.1360651e-2
1.60 459.8167 .2084474e-2 -.1006418 -.2206556e-1 -.6283173e-1 .8518793e-4 -.5190980e-1 -.1374389e-2
1.65 473.6065 .2147652e-2 -.9766987e-1 -.2334081e-1 -.5913908e-1 .8011421e-4 -.5030367e-1 -.1387318e-2
1.70 487.4427 .2211468e-2 -.9486516e-1 -.2466217e-1 -.5576028e-1 .7549878e-4 -.4867443e-1 -.1399490e-2
1.75 501.3191 .2275830e-2 -.9221418e-1 -.2602910e-1 -.5266108e-1 .7128514e-4 -.4704608e-1 -.1410955e-2
1.80 515.2306 .2340659e-2 -.8970482e-1 -.2744115e-1 -.4981169e-1 .6742571e-4 -.4543607e-1 -.1421759e-2
1.85 529.1729 .2405888e-2 -.8732619e-1 -.2889791e-1 -.4718616e-1 .6388008e-4 -.4385690e-1 -.1431946e-2
1.90 543.1420 .2471462e-2 -.8506845e-1 -.3039904e-1 -.4476180e-1 .6061374e-4 -.4231734e-1 -.1441555e-2
1.95 557.1348 .2537331e-2 -.8292273e-1 -.3194423e-1 -.4251870e-1 .5759701e-4 -.4082334e-1 -.1450625e-2
2.00 571.1486 .2603456e-2 -.8088095e-1 -.3353322e-1 -.4043932e-1 .5480422e-4 -.3937873e-1 -.1459190e-2
2.05 585.1808 .2669800e-2 -.7893580e-1 -.3516577e-1 -.3850820e-1 .5221306e-4 -.3798573e-1 -.1467283e-2
2.10 599.2295 .2736334e-2 -.7708062e-1 -.3684167e-1 -.3671164e-1 .4980402e-4 -.3664536e-1 -.1474935e-2
2.15 613.2928 .2803032e-2 -.7530935e-1 -.3856075e-1 -.3503748e-1 .4756001e-4 -.3535779e-1 -.1482175e-2
2.20 627.3692 .2869872e-2 -.7361643e-1 -.4032285e-1 -.3347489e-1 .4546593e-4 -.3412249e-1 -.1489028e-2
2.25 641.4572 .2936833e-2 -.7199679e-1 -.4212783e-1 -.3201421e-1 .4350845e-4 -.3293847e-1 -.1495521e-2
2.30 655.5557 .3003899e-2 -.7044578e-1 -.4397556e-1 -.3064681e-1 .4167571e-4 -.3180441e-1 -.1501675e-2
2.35 669.6637 .3071057e-2 -.6895913e-1 -.4586595e-1 -.2936493e-1 .3995717e-4 -.3071875e-1 -.1507512e-2
```

```
 2.40 683.7800 .3138292e-2 -.6753291e-1 -.4779889e-1 -.2816160e-1 .3834338e-4 -.2967978e-1 -.1513052e-2
 2.45 697.9041 .3205595e-2 -.6616349e-1 -.4977430e-1 -.2703057e-1 .3682589e-4 -.2868570e-1 -.1518314e-2
 2.50 712.0350 .3272956e-2 -.6484753e-1 -.5179212e-1 -.2596617e-1 .3539708e-4 -.2773465e-1 -.1523314e-2
 2.55 726.1722 .3340366e-2 -.6358196e-1 -.5385228e-1 -.2496328e-1 .3405012e-4 -.2682477e-1 -.1528069e-2
 2.60 740.3151 .3407818e-2 -.6236390e-1 -.5595473e-1 -.2401726e-1 .3277880e-4 -.2595423e-1 -.1532594e-2
 2.65 754.4631 .3475307e-2 -.6119071e-1 -.5809943e-1 -.2312390e-1 .3157751e-4 -.2512120e-1 -.1536902e-2
 2.70 768.6159 .3542825e-2 -.6005993e-1 -.6028634e-1 -.2227938e-1 .3044117e-4 -.2432391e-1 -.1541006e-2
 2.75 782.7729 .3610369e-2 -.5896928e-1 -.6251543e-1 -.2148020e-1 .2936514e-4 -.2356066e-1 -.1544919e-2
 2.80 796.9339 .3677934e-2 -.5791663e-1 -.6478667e-1 -.2072319e-1 .2834521e-4 -.2282980e-1 -.1548652e-2
 2.85 811.0985 .3745517e-2 -.5690001e-1 -.6710006e-1 -.2000543e-1 .2737752e-4 -.2212974e-1 -.1552214e-2
 2.90 825.2663 .3813114e-2 -.5591755e-1 -.6945558e-1 -.1932427e-1 .2645855e-4 -.2145898e-1 -.1555616e-2
 2.95 839.4372 .3880722e-2 -.5496755e-1 -.7185322e-1 -.1867726e-1 .2558507e-4 -.2081606e-1 -.1558867e-2
 3.00 853.6109 .3948339e-2 -.5404838e-1 -.7429298e-1 -.1806215e-1 .2475411e-4 -.2019961e-1 -.1561976e-2
 3.05 867.7871 .4015963e-2 -.5315853e-1 -.7677486e-1 -.1747690e-1 .2396295e-4 -.1960833e-1 -.1564950e-2
 3.10 881.9657 .4083591e-2 -.5229660e-1 -.7929888e-1 -.1691959e-1 .2320908e-4 -.1904097e-1 -.1567796e-2
 3.15 896.1464 .4151223e-2 -.5146125e-1 -.8186505e-1 -.1638849e-1 .2249020e-4 -.1849635e-1 -.1570522e-2
 3.20 910.3292 .4218856e-2 -.5065124e-1 -.8447338e-1 -.1588197e-1 .2180415e-4 -.1797336e-1 -.1573134e-2
 3.25 924.5138 .4286489e-2 -.4986540e-1 -.8712389e-1 -.1539855e-1 .2114898e-4 -.1747093e-1 -.1575638e-2
 3.30 938.7002 .4354122e-2 -.4910263e-1 -.8981660e-1 -.1493684e-1 .2052285e-4 -.1698807e-1 -.1578040e-2
 3.35 952.8881 .4421752e-2 -.4836189e-1 -.9255156e-1 -.1449557e-1 .1992407e-4 -.1652383e-1 -.1580345e-2
 3.40 967.0776 .4489380e-2 -.4764220e-1 -.9532878e-1 -.1407354e-1 .1935107e-4 -.1607732e-1 -.1582558e-2
 3.45 981.2684 .4557005e-2 -.4694265e-1 -.9814831e-1 -.1366966e-1 .1880238e-4 -.1564767e-1 -.1584684e-2
 3.50 995.4605 .4624625e-2 -.4626236e-1 -.1010102 -.1328290e-1 .1827666e-4 -.1523410e-1 -.1586727e-2

////////////////////////////
// Control Surface Parameters
//
// Number of control surfaces, virtual depth control weights
$NCS 4
$kDb 0.
$kDs 1.000000
// CS1
$iCS 1
$dsspCmpnt 1
$type RUDDER
$dsspLabel #TOPRUDDER
$esamLabel TOP RUDDER
$deltaMin -30.00000    // degrees
$deltaMax 30.00000    // degrees
$deldotMax 100.0000   // degrees/s
$zeta .800000
$omega 8.000000       // radians/s
$kdb 0.
$kdr -1.000000
$kds 0.
$kdphi -1.000000
$CprFlag false
// CS2
$iCS 2
$dsspCmpnt 2
$type RUDDER
$dsspLabel #BOTRUDDER
$esamLabel BOT RUDDER
$deltaMin -30.00000    // degrees
$deltaMax 30.00000    // degrees
$deldotMax 100.0000   // degrees/s
$zeta .800000
$omega 8.000000       // radians/s
$kdb 0.
$kdr 1.000000
$kds 0.
$kdphi -1.000000
$CprFlag false
// CS3
```

```
$iCS 3
$dsspCmpnt 3
$type STERNPLANE
$dsspLabel #STBDSTERNPLANE
$esamLabel STBD STRNPLN
$deltaMin -30.00000    // degrees
$deltaMax 30.00000    // degrees
$deldotMax 100.0000   // degrees/s
$zeta .800000
$omega 8.000000       // radians/s
$kdb 0.
$kdr 0.
$kds 1.000000
$kdphi -1.000000
$CprFlag false
// CS4
$iCS 4
$dsspCmpnt 4
$type STERNPLANE
$dsspLabel #PORTSTERNPLANE
$esamLabel PORT STRNPLN
$deltaMin -30.00000    // degrees
$deltaMax 30.00000    // degrees
$deldotMax 100.0000   // degrees/s
$zeta .800000
$omega 8.000000       // radians/s
$kdb 0.
$kdr 0.
$kds -1.000000
$kdphi -1.000000
$CprFlag false

//////////////////////////////////////////
// Vehicle Total and Component Added Masses
// - SI units normalized by rho
//
// Xudot = Xudot
$XudotHull -.4220917e-2
$XudotCS1 -.2598339e-4
$XudotCS2 -.2598339e-4
$XudotCS3 -.2598339e-4
$XudotCS4 -.2598339e-4
$Xudot -.4324852e-2
// Xvdot = Yudot
$XvdotHull 0.
$XvdotCS1 0.
$XvdotCS2 0.
$XvdotCS3 -.5434573e-5
$XvdotCS4 .5434573e-5
$Xvdot 0.
// Xwdot = Zudot
$XwdotHull 0.
$XwdotCS1 .5434573e-5
$XwdotCS2 -.5434573e-5
$XwdotCS3 0.
$XwdotCS4 0.
$Xwdot 0.
// Xpdot = Kudot
$XpdotHull 0.
$XpdotCS1 0.
$XpdotCS2 0.
$XpdotCS3 0.
$XpdotCS4 0.
$Xpdot 0.
```

```
// Xqdot = Mudot
$XqdotHull 0.
$XqdotCS1 .1034945e-4
$XqdotCS2 -.1034945e-4
$XqdotCS3 0.
$XqdotCS4 0.
$Xqdot 0.
// Xrdot = Nudot
$XrdotHull 0.
$XrdotCS1 0.
$XrdotCS2 0.
$XrdotCS3 .1034945e-4
$XrdotCS4 -.1034945e-4
$Xrdot 0.
// Yvdot = Yvdot
$YvdotHull -.2076851
$YvdotCS1 -.1799342e-2
$YvdotCS2 -.1799342e-2
$YvdotCS3 -.9399501e-5
$YvdotCS4 -.9399501e-5
$Yvdot -.2113026
// Ywdot = Zvdot
$YwdotHull 0.
$YwdotCS1 0.
$YwdotCS2 0.
$YwdotCS3 0.
$YwdotCS4 0.
$Ywdot 0.
// Ypdot = Kvdot
$YpdotHull 0.
$YpdotCS1 -.1904896e-3
$YpdotCS2 .1904896e-3
$YpdotCS3 0.
$YpdotCS4 0.
$Ypdot 0.
// Yqdot = Mvdot
$YqdotHull 0.
$YqdotCS1 0.
$YqdotCS2 0.
$YqdotCS3 0.
$YqdotCS4 0.
$Yqdot 0.
// Yrdot = Nvdot
$YrdotHull -.7988052e-4
$YrdotCS1 .2601564e-2
$YrdotCS2 .2601564e-2
$YrdotCS3 .1421345e-4
$YrdotCS4 .1421345e-4
$Yrdot .5151675e-2
// Zwdot = Zwdot
$ZwdotHull -.2076851
$ZwdotCS1 -.9399501e-5
$ZwdotCS2 -.9399501e-5
$ZwdotCS3 -.1799342e-2
$ZwdotCS4 -.1799342e-2
$Zwdot -.2113026
// Zpdot = Kwdot
$ZpdotHull 0.
$ZpdotCS1 0.
$ZpdotCS2 0.
$ZpdotCS3 -.1904896e-3
$ZpdotCS4 .1904896e-3
$Zpdot 0.
// Zqdot = Mwdot
```

```
$ZqdotHull .7988052e-4
$ZqdotCS1 -.1421345e-4
$ZqdotCS2 -.1421345e-4
$ZqdotCS3 -.2601564e-2
$ZqdotCS4 -.2601564e-2
$Zqdot -.5151675e-2
// Zrdot = Nwdot
$ZrdotHull 0.
$ZrdotCS1 0.
$ZrdotCS2 0.
$ZrdotCS3 0.
$ZrdotCS4 0.
$Zrdot 0.
// Kpdot = Kpdot
$KpdotHull 0.
$KpdotCS1 -.2105541e-4
$KpdotCS2 -.2105541e-4
$KpdotCS3 -.2105541e-4
$KpdotCS4 -.2105541e-4
$Kpdot -.8418807e-4
// Kqdot = Mpdot
$KqdotHull 0.
$KqdotCS1 0.
$KqdotCS2 0.
$KqdotCS3 -.2754819e-3
$KqdotCS4 .2754819e-3
$Kqdot 0.
// Krdot = Npdot
$KrdotHull 0.
$KrdotCS1 .2754819e-3
$KrdotCS2 -.2754819e-3
$KrdotCS3 0.
$KrdotCS4 0.
$Krdot 0.
// Mqdot = Mqdot
$MqdotHull -.1213535
$MqdotCS1 -.2169294e-4
$MqdotCS2 -.2169294e-4
$MqdotCS3 -.3762005e-2
$MqdotCS4 -.3762005e-2
$Mqdot -.1289209
// Mrdot = Nqdot
$MrdotHull 0.
$MrdotCS1 0.
$MrdotCS2 0.
$MrdotCS3 0.
$MrdotCS4 0.
$Mrdot 0.
// Nrdot = Nrdot
$NrdotHull -.1213535
$NrdotCS1 -.3762005e-2
$NrdotCS2 -.3762005e-2
$NrdotCS3 -.2169294e-4
$NrdotCS4 -.2169294e-4
$Nrdot -.1289209

//////////////////////
// Viscous Coefficients
// - SI units normalized by rho
//
// X
$Xuu -.7030554e-2
$Xuudbdb 0.
$Xuudrdr0 -.3659249e-2
```

```
$Xuudsds0 -.3659249e-2
$Xvv0 .4243074e-1
$Xvr -.7522974e-1
$Xww0 .4243074e-1
$Xwq .7522974e-1
$Xpr 0.
$Xqq .6075261e-1
$Xrr .6075261e-1
// Y
$Yuu 0.
$Yuudb 0.
$Yuudr0 .3760183e-1
$Yuuds -0.
$Yuv0 -.8374700e-1
$Yup 0.
$Yur0 .9332220e-1
$Yu1r1dr .2031534e-2
$Yvw 0.
$Yvq 0.
$Yvnu0 -.2704349
$Yvnu1r1v1 -.2014511e-1
$Ywp 0.
$Ywr 0.
$Ypq .6806307e-3
$Yp1p1 0.
$Yqr -0.
$Yr1r1 .1263848
// Z
$Zuu 0.
$Zuudb 0.
$Zuuds0 -.3760183e-1
$Zuw0 -.8374700e-1
$Zuq0 -.9332220e-1
$Zu1w1 0.
$Zu1q1ds -.2031534e-2
$Zvv -0.
$Zvp 0.
$Zvr -0.
$Zwnu0 -.2704349
$Zwnu1q1w1 -.2014511e-1
$Zpp 0.
$Zpr .6806307e-3
$Zq1q1 -.1263848
$Zrr 0.
$Z1wnu1 -0.
// K
$Kuu0 0.
$Kuudb 0.
$Kuudr0 0.
$Kuuds0 0.
$Kuv 0.
$Kup -.1650563e-2
$Kur -0.
$Kvw 0.
$Kvq .4188537e-3
$Kvnu -0.
$Kwp 0.
$Kwr .4188537e-3
$Kpq 0.
$Kp1p1 -.1677722e-6
$Kqr 0.
// M
$Muu -0.
$Muudb 0.
```

```
$Muudrdr0 0.
$Muuds0 -.5382057e-1
$Muw0 .6603778e-1
$Muq0 -.1496527
$Mu1w1 -0.
$Mu1q1ds -.2907806e-2
$Mvv 0.
$Mvp -.4415029e-3
$Mvr -0.
$Mwnu0 -.1590808
$Mpp 0.
$Mpr .1578736e-2
$Mqnu -.8963322e-1
$Mq1q1 -.1667665
$Mrr -0.
$M1wnu1 0.
// N
$Nuu 0.
$Nuudbdb 0.
$Nuudr0 -.5382057e-1
$Nuudsds0 0.
$Nuv0 -.6603778e-1
$Nup -0.
$Nur0 -.1496527
$Nu1r1dr -.2907806e-2
$Nvw -0.
$Nvq 0.
$Nvnu0 .1590808
$Nwp -.4415029e-3
$Nwr -0.
$Npq -.1578736e-2
$Nqr -0.
$Nrnu -.8963322e-1
$Nr1r1 -.1667665

////////////////////////
// Propulsion Parameters
//
//  Geometry m, degrees
$DP .220000
$PoD .96
$xP -1.649000
$yP 0.
$zP 0.
$psiP 0.
$thetaP 0.
// wake fraction, thrust deduction, K sign from prop handedness
$wT .221417
$tD .36919e-1
$sK -1
// Control: max rpm, max drpm/dt, damping, response frequency (rad/s)
$rpmMax 1000.
$rpmdotMax 250.0000
$zetaP .800000
$omegaP 2.500000
// rpm = [r1mps*u | r1kts*U], u in m/s, U in knots, nominally
// Jsn = J at nominal self-propulsion
// - These formulas are exact only if iniMode = 1 and the vehicle is in
//   perfect equilibrium as defined in "Initial Equilibrium" above.
// - If iniMode = 2,3, these formulae are approximately correct at
//   moderate to high speeds; see tabulated rpm vs u data above for a
//   more accurate relationship.
$r1mps 284.4173
$r1kts 146.3169
```

```
$Jsn .7465820

//////////////////////
// AutoDepth and AutoHeading parameters
// - see Section 11 of DSSP50 documentation, Part 2, Input Reference
//
// AutoDepth
$thetaL 20.00000
$ellz 2.000000
$DCC 3.000000
$DCE 30.00000
$Dk1 .150000
$Dk2 1.000000
$DKP .300000
$DKI 0.
$DKD .500000
// AutoHeading
$HCC .100000
$HCE 30.00000
$Hk1 .100000
$Hk2 1.000000
$HKP .500000
$HKI 0.
$HKD .500000
```

Appendix C: Wageningen B4-70 Family Propeller Loads

This appendix lists file `ctcqB470.f90`, Fortran 90 code for predicting the 2 quadrant performance (positive vehicle forward speed only) of the Wageningen B-Screw Series B4-70 family of propellers, as discussed van Lammeren et al [17].

```
! ctcqB470.f90
! Subroutines for predicting the 2 quadrant performance (positive vehicle forward speed only)
! of the Wageningen B-Screw Series B4-70 family of propellers, as discussed by:
!     "The Wageningen B-Screw Series," van Lammeren et al, SNAME Transactions, vol 77, pp 269-317, 1969.
! The thrust and torque coefficients C_T and C_Q presented there have been digitized and least squares fitted
! with 2-D cubic spline.  The splines are managed using software adapted from:
!     "A Practical Guide to Splines", de Boor, Springer Verlag, New York, 1978.
!
! This code can be compiled as a Linux library using: gfortran -shared -fpic -o ctcqB470.so ctcqB470.f90
! or the same command works with "gfortran" replaced with "ifort", the intel compiler.  This produces
! the library ctcqB470.so which can be linked into other code or run from Maple.
!
! (c) 2015, George D. Watt, Defence Research and Development Canada - Atlantic
!


PROGRAM ctcqB470
! Test program
IMPLICIT NONE
REAL(8) :: beta,pod,ct,cq
INTEGER :: i,j,flagtx,flagty,flagqx,flagqy
DO j=0,2,1
   pod = 0.5_8 + j*0.45_8
   DO i=0,3,1
      beta = 0.0_8 + i*1.0_8
      CALL ctB470(beta,pod,ct,flagtx,flagty)
      CALL cqB470(beta,pod,cq,flagqx,flagqy)
      WRITE(*,*) beta,pod,ct,cq,flagtx,flagty,flagqx,flagqy
   ENDDO
ENDDO
END PROGRAM ctcqB470


SUBROUTINE indexSearch(id,x,vec,n,iLo,flag)
!
! Return the index iLo giving the beginning of the interval in vec which contains the value x.
! The search starts at the interval indentified in the last call with this id, so sequential calls are efficient.
! A binary search based on SUBROUTINE INTERV from de Boor "A Practical Guide to Splines", Springer Verlag, New York, 1978.
!
!   id = integer identifying source of Call; 1 <= id <= numIds; speeds sequential call searching based on source (id).
!    x = value to bracket with vec breakpoints.
!  vec = vector of continuously increasing x breakpoints.
!    n = number of entries in vec; n > 1.
!  iLo = returned index such that vec(iLo) <= x <= vec(iLo+1).
! flag = -1,0,1 signaling that x < vec(1), vec(1) <= x <= vec(n), x > vec(n), respectively.
!
! The numIDs parameter allows any number of saves for calls from independent sources.  This should be at least 2
! to accomodate spline surfaces with two independent dimensions.  An additional 2 per additional spline surface
! is generally warranted, but not in the case of two surfaces with the same breakpoints and which are always
! called in tandem, such as thrust and torque for a single propeller operating point.
!
IMPLICIT NONE
INTEGER, INTENT(IN) :: id,n
REAL(8), INTENT(IN) :: x,vec(n)
INTEGER, INTENT(OUT) :: iLo,flag
REAL(8), PARAMETER :: eps100 = 100*EPSILON(1.0_8)
INTEGER, PARAMETER :: numIds = 4
INTEGER, SAVE :: iLoSave(numIds) = 2
INTEGER :: iHi,di,iNew
REAL(8) :: dx
!
! Use id to initialize iLo to the value returned with the last call with this id.
IF (id < 1 .OR. id > numIds) THEN
   WRITE(*,"('indexSearch ERROR: require 1 <= id <=',I2,' but got id =',I3)") numIds,id
   STOP
ENDIF
IF (n < 2) THEN
   WRITE(*,"('indexSearch ERROR: require n > 1 but got n =',I5)") n
   STOP
ENDIF
!
! Check x is within range of vec limits.  If not, return with flag = +/- 1.   If yes, find interval.
flag = 0
```

```
IF (x <= vec(2)) THEN
   dx = MAX(ABS(vec(1)),ABS(vec(n)))*eps100   ! dx > round-off error in x.
   IF (x < vec(1) - dx) flag = -1
   iLo = 1
ELSEIF (x >= vec(n-1)) THEN
   dx = MAX(ABS(vec(1)),ABS(vec(n)))*eps100   ! dx > round-off error in x.
   IF (x > vec(n) + dx) flag = 1
   iLo = n - 1
ELSE
   iLo = iLoSave(id)
   iHi = iLo + 1
   di = 1
   ! Binary expansion to bracket x
   IF (x <= vec(iHi)) THEN
      DO
         IF (x >= vec(iLo)) EXIT
         iHi = iLo
         iLo = MAX(2,iLo - di)
         di = 2*di
      ENDDO
   ELSE
      DO
         iLo = iHi
         iHi = MIN(n-1,iHi + di)
         IF (x <= vec(iHi)) EXIT
         di = 2*di
      ENDDO
   ENDIF
   ! Now vec(iLo) <= x <= vec(iHi).
   !
   ! Binary contraction to find interval
   DO
      iNew = (iLo + iHi)/2
      IF (iNew == iLo) EXIT
      IF (x < vec(iNew)) THEN
         iHi = iNew
      ELSE
         iLo = iNew
      ENDIF
   ENDDO
ENDIF
iLoSave(id) = iLo
END SUBROUTINE indexSearch


REAL(8) FUNCTION cubicPatch(X,Y,K)
!
! Calculate and return the value of the cubic spline patch surface of interest.
!
! X = x - x_vec(iLo), the local patch x coordinate.
! Y = y - y_vec(iLo), the local patch y coordinate.
! K = reference to first element of a 16 element array containing the patch coefficients in sequential order.
!
IMPLICIT NONE
REAL(8), INTENT(IN) :: X,Y,K(16)
cubicPatch = X*(X*(X*(Y*(Y*(Y*K(16) + K(15)) + K(14)) + K(13)) &
                   + Y*(Y*(Y*K(12) + K(11)) + K(10)) + K( 9)) &
                   + Y*(Y*(Y*K( 8) + K( 7)) + K( 6)) + K( 5)) &
                   + Y*(Y*(Y*K( 4) + K( 3)) + K( 2)) + K( 1)
END FUNCTION cubicPatch


SUBROUTINE ctB470(beta,pod,ct,flagx,flagy)
!
! Open water two quadrant thrust coefficient C_T as a function of beta and P/D for MARIN B 4-70 propeller series.
! A cubic spline surface least-squares fitted to digitized data from Figure 36 from
! "The Wageningen B-Screw Series," van Lammeren et al, SNAME Transactions, vol 77, pp 269-317, 1969.
! Rn = (c_0.75r*sqrt(V_A^2 + (0.75*pi*n*D)^2))/nu = 1*10^6, corrected from data acquired at Rn = 0.35*10^6.
!
!  beta = hydrodynamic pitch angle; tan(beta) = V_A/(0.7*pi*n*D); 0 <= beta <= pi.
!   pod = P/D, the pitch to diameter ratio; 0.5 <= pod <= 1.4.
!    ct = C_T, unconventional thrust coefficient used for four quadrant data.
! flagx = -1,0,1 signalling whether beta < 0,     0 <= beta <= pi, beta > pi  respectively.
! flagy = -1,0,1 signalling whether  pod < 0.5, 0.5 <=  pod <= 1.4, pod > 1.4 respectively.
!
! The flags must be 0 for the spline to be valid.
!
IMPLICIT NONE
REAL(8), INTENT(IN) :: beta,pod
REAL(8), INTENT(OUT) :: ct
INTEGER, INTENT(OUT) :: flagx,flagy
! Let x,y represent beta,P/D respectively.
```

```
INTEGER, PARAMETER :: nx = 22, ny = 3    ! no. patches in x,y directions
REAL(8), PARAMETER :: bx(nx+1) = &       ! x breakpoints
  (/0._8,.174532925199433_8,.349065850398866_8,.523598775598299_8,.698131700797732_8,.872664625997165_8,1.04719755119660_8,&
    1.17809724509617_8,1.30899693899575_8,1.43989663289532_8,1.57079632679490_8,1.70169602069447_8,1.83259571459405_8,&
    1.96349540849362_8,2.09439510239320_8,2.22529479629277_8,2.35619449019235_8,2.48709418409192_8,2.61799387799150_8,&
    2.74889357189107_8,2.87979326579064_8,3.01069295969022_8,3.14159265358979_8/)
REAL(8), PARAMETER :: by(ny+1) = &       ! y breakpoints
  (/.5_8,.8_8,1.1_8,1.4_8/)
REAL(8), PARAMETER :: K(0:3,0:3,ny,nx) = RESHAPE(&      ! spline coefficients
  (/.100485064886274_8,.277830755667736_8,0._8,-.07369566410565_8,-.308547392788365_8,-.21933993402587_8,0._8,.586826624591656_8,&
    0._8,0._8,0._8,0._8,-1.52285437829239_8,1.4433448778629_8,0._8,-5.70481224149363_8,.181844508655737_8,.257932926358942_8,&
    -.06632609769516_8,.03346924463994_8,-.358505054132156_8,-.06089674538572_8,.528143962132726_8,-1.21174869543475_8,0._8,0._8,&
    0._8,0._8,-1.24388084545368_8,-.0969544273386_8,-5.13433101734486_8,16.486090074454_8,.254158707376216_8,.22717396379496_8,&
    -.03620377751913_8,.04022641946572_8,-.361958335932802_8,-.07118251587402_8,-.562429863758858_8,.624922070843053_8,0._8,0._8,&
    0._8,0._8,-1.28993253320641_8,1.2736912823559_8,9.70315004966421_8,-10.7812778329589_8,.0385370133984755_8,.247222369454902_8,&
    0._8,-.00160516990787_8,-.447713784216258_8,-.087439536258854_8,0._8,.065491068980226_8,-.797364687888335_8,.75573361081501_8,&
    0._8,-2.98703270466427_8,.65524946865931_8,-1.7613892927072_8,0._8,13.8850723158686_8,.11266038464743_8,.24678897357959_8,&
    -.00144465291712_8,-.09037123355577_8,-.472177386231428_8,-.069756947633621_8,.0589419620823852_8,.294836370820704_8,&
    -.6512944876697_8,-.0507652194435_8,-2.68832943419814_8,8.63209657738746_8,.50172963337542_8,1.9875802325759_8,&
    12.4965650842828_8,-40.2923893164705_8,.184127034652816_8,.221521948769488_8,-.08277876311728_8,.09197640346365_8,&
    -.479839111922104_8,.045214049736754_8,.324294695820758_8,-.360327439800846_8,-.67540709499134_8,.66690319593316_8,&
    5.08055748545083_8,-5.64506387272245_8,1.13480004918906_8,-1.3934258323004_8,-23.7665853005404_8,26.4073170005978_8,&
    -.060409216187028_8,.245617691985066_8,0._8,-.00734398639129_8,-.666166446334092_8,.015396300800029_8,0._8,.29171042848072_8,&
    -.45427686838683_8,-.16652766619842_8,0._8,4.28317415901825_8,.38817858327501_8,-2.74910048367461_8,0._8,-2.45899174667438_8,&
    .013077803775927_8,.24363481565935_8,-.006609587752l7_8,.009818531710014_8,-.653671374525078_8,.09415811649058_8,&
    .262539385632819_8,-.374122597855606_8,-.388589465952902_8,.98992935673661_8,3.85485674311666_8,-12.4649491346464_8,&
    -.5029443389873_8,-3.4130282552755_8,-2.21309257200542_8,24.5584875172203_8,.085838485932228_8,.242320066569854_8,&
    .00222709078684_8,-.0024745453187_8,-.61189670501326_8,.150668646448505_8,-.0741709524373603_8,.0824121693748295_8,&
    -.0812271786870289_8,-.0626928637479574_8,-7.36359747806471_8,8.1817749756267_8,-1.062951984086l5_8,1.88990783116718_8,&
    19.8895461934857_8,-22.0994957705372_8,-.188451478184928_8,.228616326379588_8,0._8,.160968615282615_8,-.789265200177612_8,&
    -.29395998943805_8,0._8,1.56210474236006_8,-.251027037470331_8,-1.60595331344709_8,0._8,2.99564909125304_8,.4847621309967_8,&
    2.4350401185092_8,0._8,-13.4548663260357_8,-.115520427658417_8,.272077852505971_8,.144871753754l9_8,-.304615155720778_8,&
    -.835276368965295_8,.127808291000088_8,1.40589426812443_8,-2.48092773731165_8,-.651930506040662_8,-.797128058807l9_8,&
    2.69608418212877_8,.393844859916041_8,.851992775746264_8,-1.19777378952102_8,-12.1093796934364_8,20.9139750810422_8,&
    -.029083223273214_8,.276754812713944_8,-.129281886394365_8,.143646540438181_8,-.737388446441794_8,.301494362799496_8,&
    -.826940695456711_8,.918822994951833_8,-.63778753607439_8,.926860562644663_8,3.050544556049l6_8,-3.3894939511661_8,&
    -.03250620633003_8,-2.81662833369562_8,6.71319787951909_8,-7.4591087550236_8,-.33127368801099_8,.141336572195341_8,0._8,&
    .45332618314923_8,-.832590069620032_8,-.632016757104036_8,0._8,1.37820754075188_8,.002793820775805_8,-.330969288863302_8,0._8,&
    -4.04930244289729_8,-.507182918287585_8,1.32862048763013_8,0._8,2.77495792726689_8,-.276632909407348_8,.263734641645895_8,&
    .40799356483442_8,-.61443077290138_8,-.9849834931509_8,-.259900721099869_8,1.240386786677l3_8,-.432221608358368_8,&
    -.205828131841462_8,-1.42428094844506_8,-3.64437219860902_8,11.3443766052433_8,-.03367290796256_8,2.07785912799211_8,&
    2.49746213454861_8,-18.2038969121533_8,-.1773827269469_8,.342634471862905_8,-.144994130777_8,.16110458975221_8,&
    -.962988882105967_8,.367631516648366_8,.851387339154258_8,-.945985932393732_8,-.654807745908326_8,-.547922584193605_8,&
    6.56556674611567_8,-7.29507416235146_8,.3229512059145_8,-1.33871575756828_8,-13.8860450864166_8,15.4289389849117_8,&
    -.479199440710873_8,.028010650761592_8,0._8,.58527324786891_8,-.877963867772581_8,-.626130549931136_8,0._8,.218324497147142_8,&
    -.262766534244309_8,.364694771695l9_8,0._8,-2.59633786984355_8,-.30309969122226_8,7.25478019661176_8,0._8,-20.9831495381748_8,&
    -.45499386778992_8,.186034427686676_8,.52674592308219_8,-.4410806199207_8,-1.0599082713289_8,-.567182935700083_8,&
    .196492047433037_8,1.86414562635863_8,-.223459225221578_8,-.336316453162072_8,-2.33670408285626_8,1.81283847092147_8,&
    1.30678933023215_8,1.58932982130356_8,-18.8848345843764_8,35.4691628181502_8,-.363685583144523_8,.382990214156911_8,&
    .12977336515337_8,-.14419262794822_8,-1.16204693585871_8,.0540316118750295_8,1.87422311115508_8,-2.0824701235056_8,&
    -.485710889913307_8,-1.24887251573022_8,-.705149459035332_8,.783499398929558_8,1.04162056012152_8,-.164896968407433_8,&
    13.0374119520014_8,-14.4860132800087_8,-.642048821966636_8,-.031589910582345_8,0._8,.43273055787637_8,-.997385525276304_8,&
    .164151668070051_8,0._8,-2.60551825070932_8,-.42146916145515_8,4.16328879987571_8,0._8,-13.5830892762276_8,3.0673072050197_8,&
    -17.9752922028984_8,0._8,85.1387351608899_8,-.639842070078658_8,.085247340044991_8,.389457502089_8,.1280709199105_8,&
    -1.01848901762442_8,-.539338259620077_8,-2.3449664256384_8,5.73830308675712_8,.460774068049601_8,.49585469529422_8,&
    -12.2247803486115_8,20.3844486940053_8,-.02653460650571_8,5.01216629053751_8,76.6248616448332_8,-164.480795148284_8,&
    -.575758778039786_8,.353500989673448_8,.5047213300081_8,-.5608014777868_8,-1.2364032904758_8,-.396976281580028_8,&
    2.81950635244321_8,-3.13278483604815_8,.059680360001l87_8,-1.335212366487l03_8,6.12122347600715_8,-6.80135941778805_8,&
    3.93237135968343_8,6.57726858737965_8,-71.4078539887062_8,79.3420599874623_8,-.772948284059378_8,.020916983235931_8,0._8,&
    .0498867503342_8,-.950053452594835_8,.33009279879179_8,0._8,-1.78507502406815_8,.7830595612553_8,-2.8955919414688_8,0._8,&
    19.8508138384431_8,4.04762814382164_8,9.34120088926294_8,0._8,-60.2777726388454_8,-.765326246829549_8,.034386405827051_8,&
    .044898075301_8,.8595760023193_8,-.899222638607074_8,-.151877457705467_8,-1.60656752166146_8,2.61993729788718_8,&
    .45035395245249_8,2.46412779490906_8,17.8657324546048_8,-44.2070085178098_8,5.22248854935203_8,-6.93379772322396_8,&
    -54.2499953749737_8,115.053202898708_8,-.727760946241982_8,.293410771632935_8,.8185164773881_8,-.9094627526535_8,&
    -1.01863864582563_8,-.408434900274092_8,.751376046436644_8,-.834862273818663_8,1.60391898185827_8,1.24767496785925_8,&
    -21.9205752114424_8,24.3561946793823_8,1.36628612690702_8,-8.41943016552789_8,49.2978872339543_8,-54.7754302599601_8,&
    -.87481392593505_8,.0354625119087_8,0._8,.0211604302968_8,-.536983894390061_8,.0522054627019731_8,0._8,.313325833415787_8,&
    2.3725594163896_8,.77268906970875_8,0._8,-3.82021212368015_8,7.50969937312845_8,-15.2438524153813_8,0._8,64.2803914787578_8,&
    -.863603840744396_8,.041175828089845_8,.0190443872674_8,.7031062268987_8,-.51286245807724_8,.136803437724995_8,&
    .281993250075011_8,-3.03921381383479_8,2.50122040996264_8,-.258768203686052_8,-3.43819091131119_8,.974278607023231_8,&
    4.67211421844044_8,2.11185328386453_8,57.8523523308548_8,-48.8262197183633_8,-.830553229337415_8,.242441141711832_8,&
    .6518399914758_8,-.7242666571954_8,-.528500807226768_8,-.51458834196626_8,-2.45329918237675_8,2.72588798041847_8,&
    2.14045828922969_8,-2.05862752656914_8,-2.56134016497244_8,2.84593351663437_8,9.1940739809824_8,23.6401853584155_8,&
    13.9087545842589_8,-15.454171760281_8,-.887608051925642_8,.02134510194589_8,0._8,.1408928591574_8,.470180718696282_8,&
    -.52910289151669_8,0._8,2.61747806860127_8,5.32161146404313_8,-5.21355777536281_8,0._8,21.4226385812677_8,-22.8365941968045_8,&
    20.1503345520079_8,0._8,-99.9580972765264_8,-.877400414144596_8,.059386173919433_8,-.126803573242_8,.2124542330626_8,&
    .382121759093478_8,.177616187005112_8,2.35573026174078_8,-5.29402052631284_8,4.33595537312853_8,.570554641570653_8,&
    19.2803747231312_8,-18.199733039207_8,-19.4903624576678_8,-6.83835171261565_8,-89.962287548817_8,109.340684069687_8,&
    -.8424359760846_8,.192830960790436_8,.318012382998_8,-.3533470922201_8,.504483784541429_8,.161668801945951_8,&
    -2.40888821194014_8,2.67654245771099_8,5.75096269862379_8,7.22485155486917_8,2.90061498783l4_8,-3.22290554203849_8,&
    -26.6862753809736_8,-31.293739543126_8,8.44432811391672_8,-9.3825867932537_8,-.78609806754159_8,-.092051400807677_8,0._8,&
    .6263919576715_8,.689478724417576_8,-.858197507772487_8,0._8,3.08764675087503_8,-3.6462981061632_8,2.69946009913248_8,0._8,&
```

-17.8308144275782_8,7.021768758698 92_8,-1.10867658717825_8,0._8,31.8438299198803_8,-.7968009049267 39_8,.077074427764538_8,&
.5637527619046_8,-.5471353442518_8,.515385934359463_8,-.0245328850370921_8,2.77888207578748_8,-4.43813024038856_8,&
-3.31789206596816_8,-2.11485979630723_8,-16.0477329848078_8,24.7382531872725_8,7.54894919037975_8,7.48915749117497_8,&
28.6594469278357_8,-45.1359471576866_8,-.737713482321036_8,.26759954195834_8,.0713309520776_8,-.0792566134197_8,&
.638295939178871_8,.444501195530954_8,-1.21543514056324_8,1.35048348951409_8,-4.72871313743822_8,-5.06417122663689_8,&
6.21669488373368_8,-6.907438759707 68_8,11.1563760879859_8,12.498119915333_8,-11.9629055140416_8,13.2921172378742_8,&
-.742574504535045_8,-.160621357274498_8,0._8,.7964612936913_8,.095828444996301_8,-.208471127910165_8,0._8,.0564567202220725_8,&
-.88885596272004_8,2.26408382144834_8,0._8,-5.3257716602741_8,3.64030035272954_8,-7.64192149238367_8,0._8,-10.8185254343874_8,&
-.769256456787705_8,.054423192023026_8,.7168151643225_8,-.8054387588729_8,.034811438069225_8,-.193227813450166_8,&
.05081104820016 66_8,-.2818474857140 67_8,-.353426651113122_8,.826125473174848_8,-4.79319449425681_8,7.01340818685043_8,&
1.05562371828583_8,-10.5629233596678_8,-9.73667289089902_8,42.646263711495_8,-.71016298088161_8,.267043825719855_8,&
-.0080797186636_8,.0089774651816_8,-.026193793742108_8,-.23884000567303_8,-.202851688942459_8,.225390765492595_8,&
-.347614492598228_8,-.156171012925793_8,1.51887287392055_8,-1.68763652656244_8,-1.83810472958785_8,-4.89043589212293_8,&
28.6449644493894_8,-31.8277382772037_8,-.737095955100778_8,-.166256001242443_8,0._8,.6883305974491_8,-.0502531875995548_8,&
-.0085621507482204_8,0._8,-1.89394457252551_8,.540686642890062_8,-.736891731027152_8,0._8,-9.57419666368484_8,&
.53640417590731_8,13.1366113096916_8,0._8,-4.18980797075006_8,-.768387829342361_8,.019593260069703_8,.6194975377045_8,&
-.6265070313801_8,-.003451961083096_8,-.519927185329891_8,-1.7045501152727_8,3.746455109129 03_8,.06111581366328_8,&
-3.32192483022123_8,-8.616776997306 65_8,23.7605567842335_8,4.36426275360466_8,12.0053631575824_8,-3.77082717371935_8,&
-57.1094979155756_8,-.723670762775581_8,.222134884218788_8,.055641209462_8,-.0618235660689_8,-.211685339110166_8,&
-.531114375028846_8,1.66725948294362_8,-1.85251053660423_8,-1.06943653198729_8,-2.07664069686611_8,12.7677241084928_8,&
-14.1863601205726_8,6.084540811526 62_8,-5.6766975838438_8,-55.1693752976944_8,61.2993058864612_8,-.7200501911603 61_8,&
-.15053871355304_8,0._8,.2669651094191_8,.219378041648061_8,.473796903513823_8,0._8,-4.61583708098712_8,.75133207020167_8,&
4.42184346692361_8,0._8,-11.2195304062937_8,1.02959983274_8,-29.929520438105_8,0._8,146.169670278011_8,-.758003747271931_8,&
-.078458134008967_8,.2402685984776_8,.1429408602783_8,.236889511515557_8,-.772479108352619_8,-4.15425337288792_8,&
7.03128687079046_8,1.77495778930862_8,1.39257025722251_8,-10.0975773656723_8,1.33370939650901_8,-4.00267520118474_8,&
9.5362905369552_8,131.552703250222_8,-237.595243991453_8,-.756057610384406_8,.104297057351681_8,.3689153727276_8,&
-.4099059696975_8,-.17889227903887_8,-1.36658367697257_8,2.17390481082313_8,-2.41544978980333_8,1.31995705727086_8,&
-4.30587462512288_8,-8.897238908808 12_8,9.8858210098143_8,4.2828836646541_8,24.3171966094217_8,-82.283016342096_8,&
91.425573713306_8,-.676150479323908_8,-.079881596646984_8,0._8,-.2016417050243_8,.469002062693884_8,.092930073194849_8,0._8,&
-.039369829633968_8,1.15565497903308_8,-7.33145172480387_8,0._8,46.1811648840831_8,-1.1422114299351_8,28.5778078876296_8,0._8,&
-191.860252542545_8,-.705559284353636_8,-.134324857002657_8,-.1814775345215_8,.5532771965616_8,.49581809925218_8,&
.0823002191931331_8,-.0354328466713857_8,-4.83293979330249_8,.20311091346224_8,5.13746279389584_8,41.563048395673_8,&
-91.969724734911_8,2.2509041177076_8,-23.2244602988471_8,-172.674227288269_8,453.373828356187_8,-.747251235254483_8,&
-.093826534644994_8,.3164719423835_8,-.3516354915372_8,.3868298343907_8,-1.24385323300058_8,-4.38507866064303_8,&
4.87230962293713_8,3.00184153940015_8,5.2434661528835_8,-41.2097038657458_8,45.7885598508034_8,-8.01602106228056_8,&
-4.4180630156796_8,235.362218232302_8,-261.513575813555_8,-.597518318303642_8,-.129241475123125_8,0._8,.15417729359747_8,&
.712837375776605_8,-.357420445354716_8,0._8,2.18841006860645_8,.70710959939779_8,3.8910271896311_8,0._8,-29.1621801038525_8,&
-1.91130413657775_8,-15.0158554073587_8,0._8,98.1139004729479_8,-.63212797391343_8,-.087613605851018_8,.138759564238_8,&
-.6383433657244_8,.664698314022555_8,.233450273168247_8,1.96956906174554_8,-5.60524319751208_8,1.08703889348353_8,&
-3.98276143840797_8,-26.2459620934611_8,86.0697613267822_8,-3.76698544601842_8,11.4748977203296_8,88.3025104256221_8,&
-295.312924288335_8,-.663158965762115_8,-.176710576054694_8,-.4357494649143_8,.4841660721269_8,.760653045197612_8,&
-.0982239531114145_8,-3.07514981601433_8,3.41683312890446_8,-.14604257062756_8,3.5084968637404_8,51.2168231006448_8,&
-56.9075812229193_8,-.3507391473962_8,-15.2780855821243_8,-177.479121433904_8,197.199023815366_8,-.496378915137434_8,&
-.143035528223779_8,0._8,.16101612192325_8,.799709195852101_8,-.110629787394084_8,0._8,-.402765288245033_8,-.04345777988784_8,&
-2.0056854397615_8,0._8,9.3670585137565_8,.17989034759_8,3.9845844736444_8,0._8,-25.7423896292569_8,-.534942138312619_8,&
-.099561175303813_8,.1449145097312_8,-.5596523561233_8,.755645596851249_8,-.2193764152211 53_8,-.362488759420855_8,&
1.74744607769514_8,-.392252831945396_8,.523420358950673_8,8.43035266237431_8,-29.8993528550171_8,.68022116969442_8,&
-2.9658607262442_8,-23.1681506663032_8,97.5066005476117_8,-.56687879864349_8,-.163718605619159_8,-.35877261078_8,.3986362342_8,&
.704389728035078_8,.03494077010488_8,1.21021271050506_8,-1.34468078944959_8,-.28377751173163_8,-2.4911933144763_8,&
-18.4790649071479_8,20.5322943412631_8,.3380076066381_8,9.46003102181 53_8,64.5877898265753_8,-71.7642109183702_8,&
-.392038381894832_8,-.182946664114283_8,0._8,.21105788523533_8,.79757909326186_8,-.430892671886239_8,0._8,.726258219119779_8,&
.02718499441151_8,-.4409427760092_8,0._8,-.7419542543831_8,-.9998019303195_8,1.0268489533556_8,0._8,-13.6939041830573_8,&
-.441223818227748_8,-.125961035100187_8,.18995209671198_8,-.624528966951_8,.687920263612182_8,-.234802952724827_8,&
.653632397207164_8,-1.06793841452972_8,-.125130603259706_8,-.641270424690782_8,-.667758828940526_8,8.39139963958729_8,&
-1.06148265725683_8,-2.67050517607362_8,-12.3245137647658_8,24.5803280262932_8,-.478778722161572_8,-.180612598150411_8,&
-.37212397354417_8,.4134710817157_8,.647471956351342_8,-.130966886322589_8,-.307512175869681_8,.341680195410392_8,&
-.15104223500319_8,1.22375218063132_8,6.88450084669219_8,-7.64944538520762_8,-2.30817159219648_8,-3.42852486782571_8,&
9.7977814588943_8,-10.8864238432355_8,-.289412201836696_8,-.244602667083781_8,0._8,.26269719734255_8,.753302100128742_8,&
-.493546882459255_8,0._8,-.171908997253456_8,-.36543630550075_8,-.03770013498288_8,0._8,-6.11953785194061_8,1.13209207627285_8,&
14.3224961825115_8,0._8,48.9476744299631_8,-.35570017763357_8,-.173674423800831_8,.23642747760844_8,-.5654054354147_8,&
.600596492465096_8,-.539962311718293_8,-.154718097528312_8,2.3924567158316_8,-.541973867998183_8,-1.689975355000653_8,&
-5.5075840667479_8,18.0440718833714_8,6.7504281406347_8,27.5383682785898_8,44.052906986969_8,-177.400672899527_8,&
-.401789978545397_8,-.184477404798275_8,-.27243741426499_8,.30270823807218_8,.489279501499719_8,.0131701430400206_8,&
1.99849294672095_8,-2.2205477185786_8,-1.05745909965637_8,-.122626386544228_8,10.7320806282891_8,-11.9245340314339_8,&
14.1868820847544_8,6.071930787905 68_8,-115.607698622623_8,128.452998469585_8,-.194527635602071_8,-.277729415383345_8,0._8,&
.24512398541813_8,.715825374748854_8,.232819554405809_8,0._8,.742115275864555_8,.0791352132459_8,5.58673096352191_8,0._8,&
13.1021689479915_8,-1.299409887141_8,-26.7112627402722_8,0._8,-107.125936890381_8,-.271228112610777_8,-.211545939320087_8,&
.220611586876 42_8,-.34095073617819_8,.805708353518926_8,.433190678888 16_8,.667903748277688_8,-2.00275413471665_8,&
2.10891306389829_8,9.12431657947582_8,11.7919520531922_8,-51.6210094570008_8,-12.2051890052615_8,-55.6352657006486_8,&
-96.4133432013386_8,344.153097929054_8,-.324042521464845_8,-.171235685962757_8,-.08624407568404_8,.09582675076003_8,&
.941702532893337_8,.293189311482572_8,-1.13457497296679_8,1.26063885885205_8,4.51371646719027_8,2.26181525800391_8,&
-34.666956458114_8,38.5188405090161_8,-28.280835959497_8,-20.5619351806315_8,213.32444493483_8,-237.027161038705_8,&
-.102384839237016_8,-.211437797518362_8,0._8,.326492699992 55_8,.669747812963324_8,.322351486402698_8,0._8,-1.33446688456906_8,&
-.43114185618549_8,-4.90275738559884_8,0._8,-28.9660880949668_8,-9.2868441244788_8,12.7874824873326_8,0._8,144.327272011847_8,&
-.157000875592721_8,-.123284768520163_8,.29384342999335_8,-.71571107600696_8,.730422653000753_8,-.037954572431746_8,&
-1.20102019611242_8,2.17380828556202_8,-2.68405345042898_8,-12.7236011712337_8,-26.0694792854685_8,83.5275962064855_8,&
-1.5537630339611_8,51.7558459305191_8,129.894544810659_8,-375.702517456478_8,-.18686459650162_8,-.140220701046277_8,&
-.35029653841298_8,.38921837601442_8,.669637287331539_8,-.171638452996643_8,.75540726089354_8,-.839341400992789_8,&
-6.59214184377893_8,-5.81283780538075_8,49.1053571716706_8,-54.5615079685238_8,15.5195318068326_8,28.2528931036811_8,&
-208.237720900181_8,231.375245444648_8,-.042932304363427_8,-.224568070476193_8,0._8,-.02079860416277_8,.079492442906042_8,&
-.303857221493471_8,0._8,-1.49874456138466_8,-4.07807701574579_8,.11887524441784_8,0._8,27.7110990881654_8,18.9105401433164_8,&
3.49013481454473_8,0._8,-97.816095117067_8,-.110864287818676_8,-.230183693599957_8,-.01871874374645_8,.157387774447 39_8,&

```
    -.052130826699375_8,-.708518253067127_8,-1.34887010524616_8,4.72859836283924_8,-3.29421476704023_8,7.600871998224_8,&
    24.9399891793494_8,-64.0104375335662_8,17.3165460195202_8,-22.9202108670746_8,-88.0344856053648_8,198.763707778338_8,&
    -.177354612925819_8,-.198920240747256_8,.12293025325616_8,-.13658917028463_8,-.25841245629508_8,-.241118758248506_8,&
    2.90686842130916_8,-3.2298538014547_8,-.4976359548386_8,5.28204737176907_8,-32.6694046008636_8,36.2993384454043_8,&
    7.883999164933_8,-22.0747011301303_8,90.8528513951545_8,-100.947612661284_8/),&
  (/4,4,ny,nx/))
INTEGER, PARAMETER :: idctx = 1,idcty = 2  ! These may be the same as torque, if spline breakpoints the same.
INTEGER :: i,j
REAL(8) :: cubicPatch
!
! Find the patch (i,j coordinates) for this x value.
CALL indexSearch(idctx,beta,bx,nx+1,i,flagx)
CALL indexSearch(idcty, pod,by,ny+1,j,flagy)
!
! Calculate CT
ct = cubicPatch(beta-bx(i),pod-by(j),K(0,0,j,i))
END SUBROUTINE ctB470


SUBROUTINE cqB470(beta,pod,cq,flagx,flagy)
!
! Open water two quadrant torque coefficient C_Q as a function of beta and P/D for MARIN B 4-70 propeller series.
! A cubic spline surface least-squares fitted to digitized data from Figure 36 from
! "The Wageningen B-Screw Series," van Lammeren et al, SNAME Transactions, vol 77, pp 269-317, 1969.
! Rn = (c_0.75r*sqrt(V_A^2 + (0.75*pi*n*D)^2))/nu = 1*10^6, corrected from data acquired at Rn = 0.35*10^6.
!
!  beta = hydrodynamic pitch angle; tan(beta) = V_A/(0.7*pi*n*D); 0 <= beta <= pi.
!   pod = P/D, the pitch to diameter ratio; 0.5 <= pod <= 1.4.
!    cq = C_Q, unconventional torque coefficient used for four quadrant data.
! flagx = -1,0,1 signalling whether beta < 0,     0 <= beta <= pi, beta > pi  respectively.
! flagy = -1,0,1 signalling whether  pod < 0.5, 0.5 <=  pod <= 1.4, pod > 1.4 respectively.
!
! The flags must be 0 for the spline to be valid.
!
IMPLICIT NONE
REAL(8), INTENT(IN) :: beta,pod
REAL(8), INTENT(OUT) :: cq
INTEGER, INTENT(OUT) :: flagx,flagy
! Let x,y represent beta,P/D respectively.
INTEGER, PARAMETER :: nx = 22, ny = 3    ! no. patches in x,y directions
REAL(8), PARAMETER :: bx(nx+1) = &       ! x breakpoints
  (/0._8,.174532925199433_8,.349065850398866_8,.523598775598299_8,.698131700797732_8,.872664625997165_8,1.04719755119660_8,&
    1.17809724509617_8,1.30899693899575_8,1.43989662289532_8,1.57079632679490_8,1.70169602069447_8,1.83259571459405_8,&
    1.96349540849362_8,2.09439510239320_8,2.22529479629277_8,2.35619449019235_8,2.48709418409192_8,2.61799387799150_8,&
    2.74889357189107_8,2.87979326579064_8,3.01069295969022_8,3.14159265358979_8/)
REAL(8), PARAMETER :: by(ny+1) = &       ! y breakpoints
  (/.5_8,.8_8,1.1_8,1.4_8/)
REAL(8), PARAMETER :: K(0:3,0:3,ny,nx) = RESHAPE(&      ! spline coefficients
  (/.0091609031507366_8,.0393396217833733_8,0._8,.055385973880935_8,-.0229849143317415_8,-.0799430498966495_8,0._8,&
    .103322794718126_8,0._8,0._8,0._8,-.120287955272734_8,.229979671981379_8,0._8,-1.86962759411516_8,.0224582109805333_8,&
    .0542938347311834_8,.049847376492833_8,-.020619497660513_8,-.0441781138433462_8,-.0520458953226924_8,.0929905152463328_8,&
    -.159852988684676_8,0._8,0._8,0._8,0._8,-.101773998719425_8,-.274819778429512_8,-1.68266483470361_8,3.2078890401618_8,&
    .0426758988474224_8,.0786349962586002_8,.031289828598382_8,-.034766476220423_8,-.055738766762488_8,-.0394118931198268_8,&
    -.0508771745698721_8,.0565301939665275_8,0._8,0._8,0._8,0._8,-.249046763287328_8,-.418288638408375_8,1.20443530144159_8,&
    -1.33826144604629_8,.00450975864894347_8,.0266096320549396_8,0._8,.063479184540439_8,-.0339774563017212_8,-.0589263056119471_8,&
    0._8,-.0675335453863081_8,-.062982626100026_8,.120417074661928_8,0._8,-.978934719103545_8,.076935372859342_8,&
    -.712393803335052_8,0._8,4.26708816787092_8,.0142065862475668_8,.043749011880828_8,.05713126608639_8,-.031464118499099_8,&
    -.0534787537107351_8,-.0771603628661688_8,-.0607801908476544_8,.133300676028177_8,-.0532887411172577_8,-.143895299495935_8,&
    -.881041247193166_8,1.67964677368392_8,-.021571387608655_8,.439720001989991_8,3.8403793510839_8,-7.52006310785042_8,&
    .0316235725601239_8,.0695324595379463_8,.02881355943721_8,-.032015066041343_8,-.0784979614941395_8,-.0776372948472605_8,&
    .0591904175776693_8,-.0657671306418583_8,-.130400580323957_8,-.219015418917353_8,.630640849122162_8,-.700712054580222_8,&
    .252937050674114_8,.713530573521191_8,-2.92767744598074_8,3.2529749399787_8,-.0029299538673112_8,.0162056689605864_8,0._8,&
    .044558603017322_8,-.0489317838104538_8,-.0819950857157062_8,0._8,-.0192974092854081_8,-.022699359070665_8,-.252591448508091_8,&
    0._8,1.25530742096367_8,-.013740622368511_8,.228639771144373_8,0._8,-2.47663338989415_8,.0031348291023324_8,&
    .0282364917752502_8,.0401027427155892_8,.00298521099144_8,-.0740513395758693_8,-.0872053862226569_8,-.0173676683568359_8,&
    .0323853392899148_8,-.0645834932570697_8,.0863415551521429_8,1.12977667886734_8,-2.2578490620085_8,-.012017792552387_8,&
    -.44005124412701_8,-2.2289700509049_8,5.84598218526494_8,.0152956241760833_8,.0531041443723116_8,.042789432607886_8,&
    -.047543814008762_8,-.100901641433986_8,-.0888819456286146_8,.0117791370040491_8,-.0130879300045054_8,.002036949712383_8,&
    .154588315730277_8,-.902287476940119_8,1.00254164104463_8,-.1867989513697786_8,-.199018084648684_8,3.03241391583272_8,&
    -3.36934879536989_8,-.0122346763260151_8,-.00458396779831959_8,0._8,-.0662622085320869_8,-.0581110447675906_8,&
    -.149271837348654_8,0._8,.192559841229089_8,-.02989393211878_8,-.132875944283807_8,0._8,-.0414547895907484_8,&
    -.0038511878149051_8,.0333458570404648_8,0._8,-.654366120078404_8,-.0118207870351442_8,.0133068285053515_8,.059635987678883_8,&
    -.029059882518443_8,-.0976934802589988_8,-.0972806802166697_8,.173303857106205_8,-.221516260794082_8,-.0708759947228664_8,&
    -.144068737473238_8,-.0373093106317282_8,.803100052365616_8,-.011515315944828_8,-.143332995380468_8,-.58892950806963_8,&
    .779606884046521_8,-.0032461164204395_8,.041242252832697_8,.0334820934122802_8,-.0372023260136452_8,-.117261276225919_8,&
    -.0531077563674966_8,-.0260607776085096_8,.0289564195650036_8,-.0957707525078672_8,.0503826902863053_8,.685480736497107_8,&
    -.761645262774596_8,-.086469484416055_8,-.28619684153002_8,.112716687572471_8,-.125240763969341_8,-.0233080633389402_8,&
    -.0345071651362348_8,0._8,.09512846797293_8,-.0688979372730667_8,-.192606973158111_8,0._8,.118289994119378_8,&
    -.0319104093432716_8,-.115416094366219_8,0._8,-.384080088856739_8,-.017160801771511_8,.56897540006063_8,0._8,&
    -1.60603477679599_8,-.0310917442445406_8,-.0088224787835102_8,.085615621175649_8,-.0391130693838_8,-.123486199379274_8,&
    -.160668674745706_8,.106460994707517_8,.130063093180336_8,-.0769054000522535_8,-.219117718357358_8,-.345672079970712_8,&
    1.21130126230044_8,.110168879273088_8,.13534601032568_8,-1.44543129911777_8,.729134716584325_8,-.0270891355911317_8,&
```

```
.0319863577484802_8,.05041383393109_8,-.056015371034547_8,-.158593608763496_8,-.061675042762729_8,.223517778569721_8,&
-.248353087299703_8,-.141046068674771_8,-.0994696255189516_8,.744499056099601_8,-.827221173444007_8,.040370502798134_8,&
-.53504639566635_8,-.78921005419044_8,.876900060211995_8,-.0363963052501274_8,-.0686141973739873_8,0._8,.095535610652667_8,&
-.0816050151940752_8,-.18089884480098_8,0._8,-.162547099657008_8,-.0408957841391555_8,.182498728451073_8,0._8,&
-1.22499793155528_8,-.0721533120707_8,.216953055905463_8,0._8,2.73668195201693_8,-.0544011029746998_8,-.0428195824976986_8,&
.08598204958743_8,.0243620425016_8,-.140263440325099_8,-.224786561708139_8,-.146292389691222_8,.619519139206719_8,&
-.0192211097557875_8,-.148250713068671_8,-1.10249813840005_8,1.59307530715008_8,.066823017405418_8,.955857182948953_8,&
2.46301375681589_8,-7.378120125273_8,-.0588508181136185_8,.0153473987301115_8,.10790788783884_8,-.11989765315427_8,&
-.204138707151239_8,-.145291827937309_8,.411274835594821_8,-.456972039549783_8,-.119908122839356_8,-.379619263178064_8,&
.331269638035626_8,-.368077375594956_8,.376042167021172_8,.441573003213879_8,-4.17729435593316_8,4.641438173258304_8,&
-.0522684327220478_8,-.093474325131976_8,0._8,.04439999909368_8,-.102474083588608_8,-.0973684669272202_8,0._8,&
-.340059745872199_8,-.078675169994951_8,.296095082885124_8,0._8,.207925387722672_8,1.18086147753364_8,-4.3706280290309_8,0._8,&
10.8056259519682_8,-.0791119302861084_8,-.081486325376574_8,.03995999918435_8,.141790035767771_8,-.14086623680532_8,&
-.18918459831252_8,-.306053771284945_8,.501356151417439_8,.015767340339087_8,.352239437569843_8,.187132848950418_8,&
-2.27009935665918_8,.16142496952739_8,-1.4531090219983_8,9.72506335676943_8,-.598182738998419_8,-.0961330970067933_8,&
-.0192270162088062_8,.16757103137526_8,-.1861900348614_8,-.211629839626516_8,-.237450700201059_8,.145166764990671_8,&
-.161296405545202_8,.076987095385739_8,-.14841217935821_8,-1.85595657204388_8,2.06217396893739_8,.58459703108484_8,&
4.22041965253667_8,9.18669889167563_8,-10.2074432129702_8,-.0643817459915045_8,-.110949335022093_8,0._8,.02768530007968_8,&
-.0623699676562861_8,-.244519547319959_8,0._8,.269829438402975_8,.385048047846735_8,-1.42024653056021_8,0._8,&
4.45128477623965_8,-.20018561357377_8,8.72818207853094_8,0._8,-26.9590205879417_8,-.0969190433959783_8,-.10347430400045_8,&
.02491677007175_8,.16717818073853_8,-.128440437015396_8,-.171665598951006_8,.242846494562611_8,-.123703569329536_8,&
.07915877763719_8,-.218399640975398_8,4.00615629861492_8,-2.50500516895404_8,1.6903754541113_8,1.449246511978521_8,&
-24.263118529142_8,14.1635099032887_8,-.121205014409756_8,-.0433861331581567_8,.17537713273638_8,-.1948634808182_8,&
-.161423928562013_8,-.0593576659326325_8,.131513282165921_8,-.146125869073191_8,.30655781265822_8,1.50894274257672_8,&
1.75165164655709_8,-1.94627960728492_8,.3238835098131_8,-9.284476923811177_8,-11.5159596161822_8,12.7955106846414_8,&
-.0663972638796057_8,-.147715700325679_8,0._8,.07881023300368_8,.028144996307455_8,-.16767409321741_8,0._8,.0493664623073129_8,&
.306435341226777_8,2.00730255657795_8,0._8,-6.13549785214247_8,-1.22552424017403_8,-6.22439366122159_8,0._8,36.5625049971488_8,&
-.108584097686207_8,-.126436937414538_8,.07092920970334_8,.13983060843563_8,-.02082433717546_8,-.154345148394334_8,&
.0444298160765865_8,-.0514486406723305_8,.742967666192379_8,.350718136499023_8,-5.52194806692688_8,3.05699216370041_8,&
-2.10565470361809_8,3.64748268800658_8,32.9062544974273_8,-43.1024662762113_8,-.136356123609551_8,-.0461251473150888_8,&
.19677675729536_8,-.21864084143928_8,-.0645183115450305_8,-.141578391729918_8,-.00187396052839369_8,.0020821783648845_8,&
.43374656953881_8,-2.13706281945715_8,-2.77065511959551_8,3.0785056884383_8,.7863864180947_8,11.753569491884_8,&
-5.8859651511694_8,6.53996127908024_8,-.0602111718654437_8,-.14923048558224_8,0._8,.06214936156146_8,.0453725006549062_8,&
.0378765753882813_8,0._8,.322562819494153_8,-.17482690248949_8,-.437011118316553_8,0._8,8.22256428484305_8,.1166283156743_8,&
3.92579000766329_8,0._8,-46.4002861020675_8,-.103302284777953_8,-.132450157960497_8,.05593442540535_8,.0888008788144_8,&
.0654446693977315_8,.124968536651592_8,.290306537544735_8,-1.46677731193024_8,-.0839210022938_8,1.783081238589887_8,&
7.40030785635742_8,-13.8693067619173_8,.04155759321818_8,-8.602287239888889_8,-41.7602574918538_8,98.568601142765_8,&
-.135605610151675_8,-.0749132654375649_8,.13585521633828_8,-.15095024037587_8,.089459831350144_8,-.0968774150426241_8,&
-1.02979304319247_8,1.14421449243593_8,.74255979378354_8,2.4785531266867_8,-5.08206822936975_8,5.64674247707762_8,&
-3.63619952216132_8,-7.04491942645557_8,46.9514835366402_8,-52.1683150407097_8,-.057005947913039_8,-.142955245225887_8,0._8,&
.14119151850477_8,.0055981086617741_8,.125269385990265_8,0._8,.0900560116329225_8,-.129027070023838_8,1.10464301263577_8,0._8,&
-9.99878545799766_8,.34208567575901_8,-7.31720568452605_8,0._8,49.7135377045979_8,-.0960803504811738_8,-.104833535229471_8,&
.12707236665432_8,-.11976408639571_8,.0456104367729372_8,.14958450913105_8,.0810504104696052_8,-.0309142707348329_8,&
-.0676013735989_8,-1.59502906102237_8,-8.99890691219675_8,24.8384923911754_8,-.51081051157508_8,6.10544949571162_8,&
44.7421839341332_8,-127.651704316518_8,-.11932752838385_8,-.0609264185638741_8,.01928468889814_8,-.02142743210906_8,&
.096945641144702_8,.189867902314479_8,.053227566808153_8,-.0591417408979726_8,-.685372419441703_8,-.287980262723397_8,&
13.3557362398616_8,-14.8397069331788_8,1.90102487466466_8,-1.51520030926749_8,-70.1443499507321_8,77.9381666119229_8,&
-.0577167266621389_8,-.124041748016982_8,0._8,.09315736465483_8,-.010596462348631_8,.0383292221628294_8,0._8,&
.0278642173038367_8,.005309660709306_8,-1.76881694027761_8,0._8,9.52367514659114_8,.20031573681496_8,7.36910346769365_8,0._8,&
-47.0030295835069_8,-.0924140022215501_8,-.0988892595600515_8,.08384162818938_8,.01547615105479_8,.00165463816742239_8,&
.0458525608348766_8,.0250777955734869_8,-.0900445789179228_8,-.268196192416123_8,.802575349301637_8,8.57130763193142_8,&
-25.2902146711949_8,1.14196497836855_8,-5.32171451985167_8,-42.3027266251528_8,118.389300062363_8,-.11411717747408_8,&
-.0444057218617774_8,.09777016413865_8,-.10863351570962_8,.0152362043887224_8,.0365872018711737_8,-.0559623254525178_8,&
.0621803616140089_8,.061158303125888_8,-.882998032762065_8,-14.1898855721429_8,15.7665395246035_8,-1.06528367216726_8,&
1.26176052189237_8,64.2476434309649_8,-71.3862704788532_8,-.0585635263833423_8,-.132804275187574_8,0._8,.15456583594113_8,&
.0010906716866987_8,-.0459431781772574_8,0._8,.105003896305595_8,.083973466605198_8,1.1250232244262_8,0._8,-8.9343714079094_8,&
-.115722791279544_8,-1.90171294188001_8,0._8,16.1815586705403_8,-.0942315313692011_8,-.0910714994833429_8,.13910925234705_8,&
-.16411263469024_8,-.0098571765662283_8,-.0175921261747361_8,.0945035066750601_8,-.625301271420327_8,.18025240591961_8,&
-1.28725705570891_8,-8.04093426711771_8,21.2011547462514_8,-.24933458973919_8,2.46730789916538_8,14.5634028034829_8,&
-44.0116376073447_8,-.113464189639644_8,-.0519163594416218_8,-.00859211887419_8,.0095467987491_8,-.023512633146253_8,&
-.129721365453224_8,-.468267637603244_8,.520297375114695_8,-.357177616684895_8,-.387505834492428_8,11.0401050045063_8,&
-12.2667833383416_8,.613249816925786_8,-.677792572726076_8,-25.0470710431173_8,27.8300789368026_8,-.0572414533510338_8,&
-.123806664816417_8,0._8,-.0515168911132_8,.0171262375321876_8,.150831200627509_8,0._8,-1.40220915890008_8,.038529232739176_8,&
.378222298497593_8,0._8,-2.5798881775341_8,.10217818919954_8,-3.10045232099066_8,0._8,33.3662417529189_8,-.0929924967358995_8,&
-.109897104215721_8,.04636520200192_8,.0185966179551_8,.0245159504301378_8,-.227765272275441_8,-1.26198824301003_8,&
2.66276549807996_8,.082338941495054_8,-.31834750943642_8,-2.32189935978147_8,3.91782507378814_8,.0729310202317_8,&
5.90843295229472_8,30.0296175776283_8,-73.4439270926531_8,-.121286651135695_8,-.0770568961668481_8,.06310215816147_8,&
-.07011350906831_8,-.085497904675249_8,-.266011533599988_8,1.13450070526186_8,-1.26055633917989_8,-.116354976723907_8,&
-.653674355382368_8,1.20414320662974_8,-1.33793689625433_8,2.5651404564053_8,4.09634183385801_8,-36.0699168057662_8,&
40.0776853397366_8,-.0541102674881317_8,-.104536270206895_8,0._8,-.10139937112076_8,.0324655540875386_8,.0904733279117735_8,&
0._8,-.362457687435632_8,.078654513807381_8,-.839322480806462_8,0._8,10.5230043185738_8,-.02009977682789_8,4.58776563701052_8,&
0._8,-37.25643091975_8,-.0882089315704578_8,-.131914100409363_8,-.09125943400865_8,.26955302780788_8,.0498211949003058_8,&
-.00739024769588394_8,-.326211918692164_8,-.0868758485280045_8,.110978886167067_8,2.00188868520759_8,9.470703886371635_8,&
-24.9235376518426_8,.35030627944164_8,-5.4714707113198_8,-33.5307878277722_8,94.9612124345271_8,-.128718579003273_8,&
-.113890443306588_8,.15133829101841_8,-.16815365668713_8,.015899399999003_8,-.226573878013686_8,-.404400182367307_8,&
.449333535963683_8,.890973324934163_8,.954955851241068_8,-12.960479999943_8,14.4005333332693_8,-1.74495310272179_8,&
.0495839493354434_8,51.9343033633041_8,-57.7047815147802_8,-.0485578949597201_8,-.096784861971818_8,0._8,-.05209973502544_8,&
.0520204044979305_8,.106569590858467_8,0._8,.477321762567295_8,.070761349905163_8,.96228887189732_8,0._8,-4.10756189098209_8,&
.05815333520697_8,-4.79975427712333_8,0._8,13.5396882061938_8,-.0790000463969498_8,-.110851790428565_8,-.04688976152287_8,&
.04411435804474_8,.0968826097547865_8,.235446466751475_8,.429589586310588_8,-1.73043857998851_8,.248543840417809_8,&
-.146752838667902_8,-3.69680570188287_8,12.3676432681894_8,-1.01620136636259_8,-1.14403846145085_8,12.1857193855662_8,&
-30.147905012146_8,-.115284574395406_8,-.127074770670352_8,-.00718683928264_8,.0079853769807_8,.159457770888524_8,&
```

```
    .025981801941076_8,-1.12780513567918_8,1.25311681742128_8,.205731843889127_8,.974427422613105_8,7.43407323948751_8,&
    -8.26008137720797_8,-1.07669159542479_8,-1.97254118338706_8,-14.9473951253595_8,16.6082168059547_8,-.040405052878053_8,&
    -.077111892431161_8,0._8,-.02963184083622_8,.0735386480648621_8,.111768749571657_8,0._8,.0979612736934459_8,.093598111238228_8,&
    -.922570225109415_8,0._8,1.20946123407677_8,-.5448139798167_8,3.53090471065699_8,0._8,-9.45081689634891_8,-.064338680309977_8,&
    -.0851124894568409_8,-.02666865675257_8,-.03810296349077_8,.109714227326081_8,.13821829346871_8,.0881651463239706_8,&
    -.0423257688152605_8,-.150517502974597_8,-.596015691908802_8,1.08851511066691_8,.528588654780121_8,.259285377179014_8,&
    .979184148644123_8,-8.50573520670114_8,8.10520459108335_8,-.0933013862690418_8,-.111401483651015_8,-.0609613238943_8,&
    .06773480432699_8,.157971782777897_8,.179689423683202_8,.0500719543904051_8,-.0556355048781948_8,-.217083956907914_8,&
    .19981231128316_8,1.56424489997113_8,-1.73804988885667_8,.00636497712721_8,-1.93585173579057_8,-1.21105107473779_8,&
    1.3456123052642_8,-.030397068111893_8,-.070369813652241_8,0._8,-.01728247776158_8,.0700368551806491_8,.0517437246134023_8,0._8,&
    -.0712140991642157_8,-.120349838332883_8,.464012812330489_8,0._8,-2.50186588242192_8,.1309120789658_8,-2.14638706913634_8,0._8,&
    12.1123519632201_8,-.0519746391071266_8,-.0750360826477896_8,-.01555422998542_8,-.01640675608425_8,.0836371918872328_8,&
    .0325159178389398_8,-.0640926892478426_8,.512699890856419_8,-.048696373459087_8,-.211490975922959_8,-2.25167929417691_8,&
    3.71149505467803_8,-.185970538768122_8,1.12394796093211_8,10.9011167668866_8,-26.7480919479832_8,-.0763283770144494_8,&
    -.0887984447818797_8,-.03032031046125_8,.03368923384583_8,.101466522259775_8,.132489274821639_8,.397337212523055_8,&
    -.441485791692217_8,-.214584436235745_8,-.560394887667481_8,1.08866625503085_8,-1.2096291722564_8,.410115875936282_8,&
    .442633195113291_8,-13.1721659862902_8,14.6357399847655_8,-.0229978004303076_8,-.060460039616603_8,0._8,-.042306009174271_8,&
    .0452587705065596_8,.0628887069708187_8,0._8,-.103575418857983_8,-.06894078513937_8,-.378871418688695_8,0._8,&
    2.25464361074669_8,.350264090508_8,3.7528603337751_8,0._8,-7.85463145848531_8,-.0422780745629926_8,-.0718826620935888_8,&
    -.038075408256826_8,.05430684605241_8,.0613288462886404_8,.0349233438791077_8,-.0932178769720916_8,.109403034357821_8,&
    -.121726833255817_8,.229882356212969_8,2.0291792496703_8,-6.79245609048888_8,1.26404714126138_8,1.63210983998203_8,&
    -7.06916831263113_8,24.3749000871024_8,-.0658033750907889_8,-.0800650586136165_8,.01080075319033_8,-.01200083687814_8,&
    .0663701224525594_8,.00853143697249156_8,.0052448539498443_8,-.00582761549979439_8,-.0535323083648_8,-.386573238416497_8,&
    -4.08403123176894_8,4.53781247974289_8,1.77557724747156_8,3.97183187592212_8,14.8682417657595_8,-16.5202686286214_8,&
    -.0174691048051731_8,-.0503023801530372_8,0._8,-.034848686150972_8,.0452151568734806_8,.156613145592343_8,0._8,&
    .0829279334913579_8,.068607601554934_8,1.09487338812769_8,0._8,-.829862950082191_8,-.14981824119754_8,-6.97733457282278_8,0._8,&
    1.76027101864264_8,-.0335007333771594_8,-.0597115254137432_8,-.031363817535862_8,.00691198599081_8,.0944381547554488_8,&
    .179003687634864_8,.0746351401421493_8,-.415885826490273_8,.374663318340991_8,.870810391604719_8,-.74687665507347_8,&
    2.77954479021506_8,-2.19549129554088_8,-6.50206139778466_8,1.58424391677708_8,-8.05189253623654_8,-.0540503109577752_8,&
    -.0766635797178122_8,-.02514303014415_8,.02793670016017_8,.143627506343509_8,.111495598567958_8,-.299662103699083_8,&
    .332957892998937_8,.643735246201794_8,1.17316149191949_8,1.7547136561202_8,-1.9496818401338_8,-4.22092886084586_8,&
    -7.72552603250724_8,-5.66245936583563_8,6.2916215175969_8,-.0107109138530332_8,-.0266910868659074_8,0._8,-.034264756263884_8,&
    .0554752996865216_8,.0845860991177138_8,0._8,-.043844373626309_8,.009774115814797_8,-1.64511949132469_8,0._8,&
    -.138606137521004_8,-1.05426117446478_8,4.89704610634206_8,0._8,-.203796047503158_8,-.0196433883319297_8,-.0359425710571216_8,&
    -.030838280637487_8,-.017960430797557_8,.0796673313339233_8,.0727481182385021_8,-.0394599362636957_8,-.102103712538538_8,&
    -.487504097295662_8,-1.68254314845432_8,-.124745523768968_8,-.382426014702398_8,.40935016415527_8,4.84202117351431_8,&
    -.18341644275198_8,9.9083154639932_8,-.0336865365379846_8,-.0592948557549989_8,-.047002668355301_8,.052225187061443_8,&
    .0951835723032361_8,.0215041540950102_8,-.131353277548318_8,.145948086164779_8,-1.01381964136841_8,-1.86064548668652_8,&
    -.468928937000939_8,.521032152223698_8,2.11297355389022_8,7.4072164831433_8,8.73406747483996_8,-9.70451941649041_8,&
    -.0056463722295071_8,-.0328237344576762_8,0._8,-.042836050525131_8,.0038407159353497_8,-.0943764900851141_8,0._8,&
    -.0906073462316808_8,-.40423327926796_8,.277946017672671_8,0._8,-.218636658228935_8,1.9104608621802_8,.900561958586765_8,0._8,&
    .63401363905321_8,-.016650065930988_8,-.0443894680994282_8,-.038552445472611_8,-.015654875107172_8,-.0269186294384398_8,&
    -.11884047356761_8,-.0815466116085073_8,.307106317556393_8,-.326752663738356_8,.218914119951081_8,-.196772992405802_8,&
    3.50856036918926_8,2.19774781801046_8,1.07174564112883_8,.57061227514656_8,-19.2071075089341_8,-.0338593080812551_8,&
    -.0717477516619743_8,-.052641833069072_8,.058490925632301_8,-.0616180959794816_8,-.0848497347925581_8,.194849074192258_8,&
    -.216498971324721_8,-.18405686710149_8,1.04816162418836_8,2.96093133986401_8,-3.28992371096024_8,2.05203471237198_8,&
    -3.7718060211926_8,-16.7157844828923_8,18.5730938698811_8/),&
  (/4,4,ny,nx/))
INTEGER, PARAMETER :: idcqx = 1,idcqy = 2  ! These may be the same as thrust, if spline breakpoints the same.
INTEGER :: i,j
REAL(8) :: cubicPatch
!
! Find the patch (i,j coordinates) for this x value.
CALL indexSearch(idcqx,beta,bx,nx+1,i,flagx)
CALL indexSearch(idcqy,pod,by,ny+1,j,flagy)
!
! Calculate CQ
cq = cubicPatch(beta-bx(i),pod-by(j),K(0,0,j,i))
END SUBROUTINE cqB470
```

Appendix D: Dynamic Change Module

This appendix lists `dc3Module.mpl`, Maple code for creating a module for modelling dynamic change in a system. Read this script into Maple, call `dcMake` to instantiate a dynamic change model, and call `command` to issue system change commands. Many different systems can be instantiated simultaneously and independently. See the file header for a detailed explanation.

```
# Dynamic Change Module, version 3.01
# OBJECTIVE: Analytically model control system dynamic change with limits
#
# George Watt
# DRDC Atlantic
# (c) 2019 Defence Research and Development Canada
# September 2019
#
# Instantiate a dynamic change module for state d by calling procedure dcMake
# with characteristic parameters.  Do this for as many different control
# systems as necessary, creating an independent module for each.  Once
# created, analytical solutions for responses d(t) to any number of commanded
# changes are generated using the command(t,dc) procedure, which automatically
# matches d and d' values for the new response to the old at time t.
#
# The dynamic response is governed by the second order ordinary differential
# equation:
#    d''(t) + 2 zeta omega d'(t) = omega^2(dc - d(t)).
# This ODE governs the response unless there are nonlinear effects which limit
# the rate d', or if hard stops are encountered.
#
# The dcMake procedure call is:
#
#    dcMake(ZETA,OMEGA,DDRL,{dIni,dIniD,dRange,dHistory,Verbose,
#                            dMinS,dMinSH,dMinH,dMaxS,dMaxSH,dMaxH})
#
# The first three parameters must always be present and entered in the correct
# order.
#    ZETA, dimensionless damping 0 < zeta < 1.
#    OMEGA, rad/s, response frequency, omega > 0.
#    DDRL, d unit/s, the rate limit; ie, maximum allowable |d'|, DDRL > 0.
#
# The remaining keyword parameters in dcMake (in the {}'s) have default
# values and don't need to be specified.  They can be changed using the syntax
# keyword=value, as in dMinS = 20, and given in any order after the first
# three required parameters (do not enclose with {}'s).
#
#    dIni,     d unit, initial value for d (default = 0).
#    dIniD,    d/s, initial value for d'(t) (default = 0).
#    dRange,   d unit, nominal range (> 0) of d for estimating error
#                      (default = 100).
#    dHistory, boolean, remember response time history (default = true).
#    Verbose,  boolean, write out useful info and numbers (default = false).
```

```
#    dMinS,    d unit, soft minimum d limit (default = no limit).
#    dMinSH,   d unit, soft & hard minimum d limit (default = no limit).
#    dMinH,    d unit, hard minimum d limit (default = no limit).
#    dMaxS,    d unit, soft maximum d limit (default = no limit).
#    dMaxSH,   d unit, soft & hard maximum d limit (default = no limit).
#    dMaxH,    d unit, hard maximum d limit (default = no limit).
#
# dIni and dIniD specify the initial conditions when procedure command() is
# first called.  On subsequent command() calls, the current response provides
# these initial conditions.
#
# dRange gives an order of magnitude estimate of the range of d and is only
# used to convert relative d error (specified internally) to absolute error.
#
# If dHistory is true, the response d(t) is a piecewise function containing
# the entire response history.  If dHistory is false, d(t) is still a
# piecewise function but contains only the latest response and its initial
# conditions.  If this module is being used in a numerical integration where
# command() calls are frequent, it is better to set dHistory false.
#
# if Verbose is true, useful information and numbers are written out when
# dcMake and command are called, making it possible to follow what has
# happened.
#
# If the rate limit is not exceeded and if hard stops are not encountered, the
# response is determined completely by the above ODE, is C^1 continuous at the
# time a new command is issued, and is C^infinity continuous otherwise.
#
# Soft limits only cap the commanded change (dc in the call command(t,dc)) but
# allow the response to temporarily overshoot the limits thereby avoiding
# discontinuities.  Hard limits are rigidly enforced producing a discontinuity
# in d' when the limit is encountered (there is C^0 continuity at the hard
# limit).  Following a hard limit, a new solution is generated with dIniD=0.
#
# If the rate is limited, d' will be continuous when this occurs but not d''
# (C^1 continuity).  While the rate is limited, d' is constant with |d'| =
# DDRL and with d'' = 0.  As soon as the ODE predicts d'' = 0, rate limiting
# stops which creates a second discontinuity but with both d' and d''
# continuous (C^2 continuity).  The rate can be limited only once for each
# command() call.
#
# The times at which discontinuities occur in the latest response are listed
# in chronological order in the parameter tDiscs.  These are both the rate and
# hard limit discontinuities.  tDiscs is returned by command() and it can be
# accessed directly since it is exported by the module.  It is an empty list
# if there are no discontinuites.
#
# The call "m := dcMake(...)" (where m is any name) instantiates and returns a
# new module named m in which d = dIni and d' = dIniD are constant for all
# time.  A command is issued to m at t = t0 using m:-command(t0,dc).  The
```

```
# response to this command matches d and d' to dIni and dIniD at t = t0 and
# can be obtained for any t by calling procedures:
#    m:-delta(t)   for d
#    m:-deltaD(t)  for d'
#    m:-deltaDD(t) for d''.
# Any number of subsequent calls to command can be made "m:-command(ti,dc_i);"
# with no restriction on ti.  If dHistory = true, the m:-delta* procedures
# provide a complete history of all the merged responses.
#
# Module parameters and procedures that have been "exported" are available
# using the prefix "m:-" as in, for example, m:-tDiscs.  However, these
# parameters are associated only with the response to the latest command().
#
# All calculations use 15 significant figures.  Numerical error is controlled
# using a relative error Err = 1e-10.  This number is hardwired into the code
# and it is inadvisable to change it.  In most cases, numerical relative error
# will be O(Err).  The exception is when zeta > 1 - sqrt(Err).
#
# Maple does numerical computation several different ways and it takes time
# for it to figure out how to handle each computation.  This delay is avoided
# and speed maximized by specifying a priori (using option hfloat) that all
# computations using numeric quantities are to be done using hardware floating
# point (HFloat) numbers.
#
dcMake := proc(ZETA::And(positive,numeric),
               OMEGA::And(positive,numeric),
                DDRL::And(positive,numeric),
               {dIni::numeric:=HFloat(0.),
               dIniD::numeric:=HFloat(0.),
              dRange::And(positive,numeric):=HFloat(100.),
            dHistory::truefalse := true,
             Verbose::truefalse := false,
               dMinS::numeric:=HFloat(-1.e99),
              dMinSH::numeric:=HFloat(-1.e99),
               dMinH::numeric:=HFloat(-1.e99),
               dMaxS::numeric:=HFloat( 1.e99),
              dMaxSH::numeric:=HFloat( 1.e99),
               dMaxH::numeric:=HFloat( 1.e99)})
  option hfloat;
  # Err is acceptable relative error -- change at own risk.
  local Err :=  1.e-10,
        pi :=  evalhf(Pi);

  # Ensure 'option hfloat' takes precedence over sFloats with more than
  # evalhf(Digits) sig figs.
  UseHardwareFloats := true;

  # Warn if hardware floats have less than 15 significant figures.
  if evalhf(Digits) < 15 then
    WARNING("Module was tested with hardware floats having 15 sig figs;\n"
```

```
          "HFloats on this machine only have %1 sig figs",evalhf(Digits));
end if;

return module()
  option hfloat; # fast -- all floating point calcs use hardware floats

  # t is always a variable and never given an explicit value.
  local t,z2,rtz,oz,ortz,acz,odc0,z2dd0,fdf,FdF,dErr,tErr,ddErr,iniDel,
        beta,alpha,T0,Te0,Te1,Ti,T1,T2,Ts,dc,dc0,d1,d2,dd0,dd1,noHardLimits,
        rateLimChk,gRegion,g,h,response,deltaUp,hardCapit,applyLimits,nLims;
  export delta,deltaD,deltaDD,dsMin,dhMin,dsMax,dhMax,command,newton,tDiscs,
         softLimited,hardLimited,rateLimited;

  # Allow limits to be changed using these exported variables
  dsMin := max(dMinS,dMinSH);
  dhMin := max(dMinH,dMinSH);
  dsMax := min(dMaxS,dMaxSH);
  dhMax := min(dMaxH,dMaxSH);

  if OMEGA < Err then
    error "require omega > %1 ",Err
  end if;

  # Absolute error based on change over a half period
  dErr := dRange*Err;             # max acceptable d error
  tErr := pi*Err/OMEGA;           # max acceptable time error
  ddErr := dErr*OMEGA/pi;         # max acceptable d' error

  if DDRL < ddErr then
    error "DDRL < %1 is too small.",ddErr
  end if;

  if dIni < max(dsMin,dhMin) then
    error "Initial delta < greatest minimum"
  end if;
  if dIni > min(dsMax,dhMax) then
    error "Initial delta > smallest maximum"
  end if;
  if abs(dIniD) > DDRL then
    error "Initial rate magnitude > rate limit"
  end if;

  z2 := ZETA + ZETA;
  rtz := sqrt((1.0 - ZETA)*(1.0 + ZETA)); # maintain sig figs for zeta ~ 1
  oz := OMEGA*ZETA;
  ortz := OMEGA*rtz;
  acz := arccos(ZETA);

  ########################################################
  # Initialize response prior to calling command()
```

```
iniDel := proc(D1::numeric,DD1::numeric,DDD1::numeric)
  option hfloat;
  delta   := unapply(piecewise(t<infinity,D1,0.0),t);
  deltaD  := unapply(piecewise(t<infinity,DD1,0.0),t);
  deltaDD := unapply(piecewise(t<infinity,DDD1,0.0),t);
end proc;

# Initialize response based on initial conditions
iniDel(dIni,dIniD,0.0);

# Newton zero finding function to find ts (t_\ell in report)
# FdF is (d - ds)/d'; dcs = dc - ds, ts0 = ts - t0
FdF := (dcs,alpha,ts0,beta)
         ->  (dcs*exp(oz*ts0)/alpha - sin(ortz*ts0 + beta))
             /(OMEGA*sin(ortz*ts0 + beta - acz));

# Newton zero finding function to find t1
# fdf is (d' - d_1')/d''; dd1 = signum(dc - d0)*DDRL, t1t0 = t1 - t0
fdf := (dd1,alpha,t1t0,beta)
      -> (dd1*exp(oz*t1t0)/(OMEGA*alpha) - sin(ortz*t1t0 + beta - acz))
         /(OMEGA*sin(ortz*t1t0 + beta - acz - acz));

if Verbose then
  printf("\nzeta,omega,ddrl =  %g,%g,%g\n",ZETA,OMEGA,DDRL);
  printf("dIni,dIniD = %g,%g\n",dIni,dIniD);
  printf("Err,dRange = %g,%g\n",Err,dRange);
  printf("tErr,dErr,ddErr = %g,%g,%g\n",tErr,dErr,ddErr);
  printf("dsMin,dsMax = %g,%g\n",dsMin,dsMax);
  printf("dhMin,dhMax = %g,%g\n",dhMin,dhMax);
  printf("dHistory = %a\n",dHistory);
end if;


#########################################################################
# command(TNOT,DC) an exported procedure to set the new command
#                  to DC at t = T0.  The exported response is:
#   delta(t)   = new response
#   deltaD(t)  = new delta'(t)
#   deltaDD(t) = new delta''(t)
#
# Input parameters are:
#   TNOT = time (s) at which commanded change is initiated
#          (new delta matches old delta and delta' at t = T0).
#   DC   = commanded delta (d units) to change to.
#
command := proc(TNOT::numeric,DC::numeric)
  option hfloat;
  #
  # Apply soft limits to DC.
  if DC < dsMin then
```

```
    dc := dsMin;
    softLimited := true;
  elif DC > dsMax then
    dc := dsMax;
    softLimited := true;
  else
    dc := DC;
    softLimited := false;
  end if;

  # Initialize these parameters for the new response
  rateLimited := false;
  hardLimited := false;
  noHardLimits := evalb(dhMin = -1.0e99 and dhMax = 1.0e99);
  tDiscs := [];
  if not dHistory then
    # Forget previous response except initial conditions for new response.
    iniDel(delta(TNOT),deltaD(TNOT),deltaDD(TNOT));
  end if;
  #
  # New response, ignoring any limitations
  response(TNOT,delta(TNOT),deltaD(TNOT));
  #
  # Check for limits being exceeded and apply if necessary
  nLims := 0;
  applyLimits();

  if Verbose then
    print('softLimited' = softLimited,'dc'=dc);
    print('hardLimited' = hardLimited);
    print('rateLimited' = rateLimited);
    print('delta'(t) = delta(t));
  end if;

  # Return list of discontinuities in the new reponse
  tDiscs;
end proc:


#####################################################################
# response(Tnot,D0,DD0) calculate new response using dc and the
#                       given initial conditions.
# Tnot = time new response begins
#   D0 = d(Tnot) for new response
#  DD0 = d'(Tnot) for new response
#
response := proc(Tnot,D0,DD0)
  option hfloat;
  local a,b,bdenom;
  #
```

```
T0 := Tnot;
dc0 := dc - D0;
odc0 := OMEGA*dc0;
z2dd0 := z2*DD0;
dd1 := signum(dc0)*DDRL; # only needed if dc0 <> 0
#
# Default values
rateLimChk,gRegion,g,h := false$4; # make unused g,h non-numeric
#
if abs(z2dd0) < ddErr and abs(odc0) < ddErr then
  # Trivial solution: d = dc
  h,beta,alpha := 0.0,0.0,0.0;
elif abs(z2dd0) < abs(odc0)*(1.0 + Err) then
  # |g| < 1 + Err
  gRegion := true;
  g := z2dd0/odc0;
  if abs(g) < Err then
    # g = 0, possibly ratelimited
    rateLimChk := true;
    g := 0.0;
    beta := acz;
    alpha := dc0/rtz;
    Te0 := T0;
    Te1 := Te0 + pi/ortz;
    Ti := T0 + acz/ortz;
  elif g > 1.0 - Err then
    # exactly rate limited -- no limit needed
    g := 1.0;
    beta := acz + acz;
    alpha := dc0/(z2*rtz);
    Te0 := T0 - acz/ortz;
    Te1 := Te0 + pi/ortz;
    Ti := T0;
  else
    # general case
    rateLimChk := evalb(g > -0.772);
    bdenom := sqrt(g*g + z2*z2*(1.0 - g));
    beta := arccos((z2*ZETA - g)/bdenom);
    #alpha := dc0/sin(beta);
    alpha := dc0*bdenom/(z2*rtz);
    Te0 := T0 + (acz - beta)/ortz;
    Te1 := Te0 + pi/ortz;
    Ti := T0 + (acz + acz - beta)/ortz;
  end if;
else
  # |h| < 1, cannot be rate limited
  h := odc0/z2dd0;
  if abs(h) < Err then
    # h = 0
    h,beta := 0.0,0.0;
```

```
      alpha := -DD0/ortz;
      Te0 := T0 + acz/ortz;
      Te1 := Te0 + pi/ortz;
      Ti := T0 + 2.0*acz/ortz;
    else
      # general case
      bdenom := sqrt(z2*z2*h*(h - 1.0) + 1.0);
      if h > 0 then
        a := pi;
      else
        a := 0.0;
        bdenom := -bdenom;  # give bdenom the sign of h
      end if;
      beta := arccos((z2*ZETA*h - 1.0)/bdenom);
      #alpha := DD0/(omega*sin(beta - acz));
      alpha := DD0*bdenom/ortz;
      Te0 := T0 + (acz - beta)/ortz;
      Te1 := Te0 + pi/ortz;
      Ti := T0 + (acz + acz - beta + a)/ortz;
    end if;
  end if;
  # Update Deltas with new, unlimited response
  a := ortz*(t - T0) + beta;
  b := alpha*exp(-oz*(t - T0));
  deltaUp(T0, dc - b*sin(a),
             OMEGA*b*sin(a - acz),
          -OMEGA^2*b*sin(a - acz - acz));

  if Verbose then
    printf("\nresponse(T0,D0,DD0) (%g,%g,%g) called with dc = %g\n",
           Tnot,D0,DD0,dc);
    if gRegion then
      printf("   g,Te0,Ti,Te1 = %g, %g, %g, %g\n",g,Te0,Ti,Te1);
    else
      printf("   h,Te0,Ti,Te1 = %g, %g, %g, %g\n",h,Te0,Ti,Te1);
    end if;
  end if;

end proc:


######################################################################
# deltaUp(T,D,DD,DDD) update delta(t), deltaD(t), deltaDD(t) at
#                     time T with new response D, DD, and DDD.
#
#   T = time following which new response applies.
#   D = new response d
#  DD = new response d'
# DDD = new response d''
#
```

```
# Return: true if T precedes a previous time break,
#         false if T follows all previous time breaks.
#
deltaUp := proc(T,D,DD,DDD)
  option hfloat;
  local d,dd,ddd,n,i;
  #
  # Break old response into components
  d,dd,ddd := [op(delta(t))],[op(deltaD(t))],[op(deltaDD(t))];
  n := nops(d);
  # The d[i] are the break points in the old response.
  for i from 1 to n-2 by 2 do
    if T < rhs(d[i]) + tErr then break end if;
  end do;
  # After i the old response is discarded.
  if i = 1 then
    # the first call to command uses this branch -- it should return false
    delta   := unapply(piecewise(t<T,  d[2],  D),t);
    deltaD  := unapply(piecewise(t<T, dd[2], DD),t);
    deltaDD := unapply(piecewise(t<T,ddd[2],DDD),t);
    if rhs(d[i]) = infinity then false else true end if;
  elif i < n then
    delta   := unapply(piecewise(  d[1..i-1][],t<T,  d[i+1],  D),t);
    deltaD  := unapply(piecewise( dd[1..i-1][],t<T, dd[i+1], DD),t);
    deltaDD := unapply(piecewise(ddd[1..i-1][],t<T,ddd[i+1],DDD),t);
    true;
  else
    delta   := unapply(piecewise(  d[1..-2][],t<T,  d[-1],  D),t);
    deltaD  := unapply(piecewise( dd[1..-2][],t<T, dd[-1], DD),t);
    deltaDD := unapply(piecewise(ddd[1..-2][],t<T,ddd[-1],DDD),t);
    false;
  end if;
end proc:


#####################################################################
# applyLimits() check to see if the rate or delta limits are
#               exceeded; if so, break and restart the response.
#
# Return:  true if response gets changed (discontinuities added),
#         false if response is not changed.
#
applyLimits := proc()
  option hfloat;
  local a,b;
  #
  if Verbose then
    nLims := nLims + 1;
    printf("\napplyLimits[%d]\n",nLims);
  end if;
```

```
#
if Te0 > T0 + tErr then
  # First extremum that might need to be capped precedes Ti
  if hardCapit(Te0,T0) then
    thisproc();
    return true;
  end if;
end if;
if rateLimChk and abs(deltaD(Ti)) > DDRL + ddErr then
  # The rate limit is being exceeded.
  # Response is a straight line between (T1,d1) and (T2,d2).
  if Verbose then
    printf("   rate limited = true, T1 iteration:\n")
  end if;
  a := unapply(fdf(dd1,alpha,t - T0,beta),t);
  T1 := newton(a,T0,ERR=tErr,verbose=Verbose);
  d1 := delta(T1);
  if hardCapit(T1,T1) then
    # hard limit precedes T1
    return true;
  end if;
  d2 := dc - z2*dd1/OMEGA;
  T2 := T1 + (d2 - d1)/dd1;
  if Verbose then
    printf("   constant rate ramp T1,d1, T2,d2 = %g,%g, %g,%g\n",
           T1,d1,T2,d2);
  end if;
  # Check if hard limit occurs during constant deltaD ramp
  if d2 < dhMin or d2 > dhMax then
    if dd1 < 0 then a := dhMin else a := dhMax end if;
    Ts := T1 + (a - d1)/dd1;
    if Ts > T1 + tErr then
      # the constant deltaD ramp
      deltaUp(T1,d1 + dd1*(t-T1),dd1,0.0);
      tDiscs := [tDiscs[],T1];
      rateLimited := true;
    end if;
    deltaUp(Ts,a,0.0,0.0);
    hardLimited := true:
    tDiscs := [tDiscs[],Ts];
    if Verbose then
      printf("   hard capped on ramp at Ts,ds = %g,%g\n",Ts,a);
    end if;
    return true;
  end if;
  deltaUp(T1,d1 + dd1*(t-T1),dd1,0.0);
  response(T2,d2,dd1);  # resets T0 to T2
  rateLimited := true;
  tDiscs := [tDiscs[],T1,T2];
  thisproc();
```

```
    return true;
  elif Verbose then
    printf("   rate limited = false\n");
  end if;
  # Test the overshoot extremum and cap if necessary
  hardCapit(Te1,Ti);
end proc:


#####################################################################
# hardCapit(T,Tj) check to see if the current response needs to be
#                 capped.  If so, cap it and restart the response.
#   T = time at wich to check if delta exceeds the cap (eg, Te).
#  Tj = initial guess for newton solution for cap time (eg, T0,Ti).
#
# Return:  true if the response gets capped,
#         false if the response is not capped.
#
hardCapit := proc(T,Tj)
  option hfloat;
  local a,b,c,cc;
  #
  if noHardLimits then
    return false;
  elif deltaD(Tj) < 0 then
    # response is decreasing
    a := dhMin;
    b := delta(T) < dhMin - dErr;  # is cap exceeded?
    c := dc < dhMin + dErr;        # does dc exceed cap?
    cc := c and dc > dhMin - dErr; # does dc equal cap?
  else
    # response is increasing
    a := dhMax;
    b := delta(T) > dhMax + dErr;
    c := dc > dhMax - dErr;
    cc := c and dc < dhMax + dErr;
  end if;
  if b then
    # cap is being exceeded
    if Verbose then printf("   hardCapit(%g,%g) = true\n",T,Tj) end if;
    if cc then
      # analytical solution for Ts
      Ts := T0 + (pi - beta)/ortz;
      if Verbose then
        printf("   analytical hard cap: Ts,ds = %g,%g\n",Ts,a);
      end if;
    else
      # numerical solution
      b := unapply(FdF(dc - a,alpha,t - T0,beta),t);
      Ts := newton(b,Tj,ERR=tErr,verbose=Verbose);
```

```
      if Verbose then
        printf("   numerical hard cap: Ts,ds = %g,%g\n",Ts,a);
      end if;
    end if;
    if c then
      # just stay at cap
      deltaUp(Ts,a,0.0,0.0);
    else
      # need new response
      response(Ts,a,0.0);  # new T0 = Ts begins post-cap response
    end if;
    hardLimited := true:
    tDiscs := [tDiscs[],Ts];
    true;
  else
    if Verbose then printf("   hardCapit(%g,%g) = false\n",T,Tj) end if;
    false;
  end if;
end proc:


#############################################################################
# Newton's method.  Solve F(x) = 0 using Newton's method.
#
#     newton(FDF,X0,{ERR=Err,verbose=false,count=100})
#
# FDF = a procedure in one variable, x, returning F(x)/(dF(x)/dx).
# X0  = initial guess for x.
#
# Options
#
# ERR     = desired level of error in x (default = Err).  Will be
#           increased to sqrt(ERR) with WARNING if converge is poor.
# verbose = true/false (default false), print count,dx,x each iteration.
# count   = <posint> (default 100), max # of iterations allowed.
#
# Return x, the solution to F(x) = 0.
#
# - A pure Newton method.  Requires user to know F well enough to know
# that the initial guess X0 will converge.  True for the usage herein, but
# in extreme cases round-off error may prevent convergence.
#
newton := proc(FDF::procedure,X0::numeric,
               {ERR::And(positive,numeric) := Err,
                        verbose::truefalse := false,
                             count::posint := 100})
  option hfloat;
  local x,dx,c,fmt;
  x := X0;
  dx := ERR + ERR;
```

```
    c := 0; # counter
    if verbose then
      fmt := "      %5d      %9.2g   %19.15f\n";
      printf(cat("      iteration       dx         x\n",fmt),c,0,x)
    end if;
    while abs(dx) > ERR do
      if c = count then
        if abs(dx) > sqrt(ERR) then
          error "from newton(FDF,%1)\n       %2 iterations "
                "without convergence: |dx| = %3\n",X0,c,abs(dx);
        else
          WARNING("from newton(FDF,%1)\n         "
                  "%2 iterations with reduced convergence: |dx| = %3\n",
                  X0,c,abs(dx));
          return x;
        end if;
      end if;
      dx := -FDF(x);
      x := x + dx;
      c := c + 1;
      if verbose then printf(fmt,c,dx,x) end if;
    end do;
    x;
  end proc:

  end module;
end proc:
```

DOCUMENT CONTROL DATA

*Security markings for the title, authors, abstract and keywords must be entered when the document is sensitive

1. ORIGINATOR (Name and address of the organization preparing the document. A DRDC Centre sponsoring a contractor's report, or tasking agency, is entered in Section 8.)

DRDC – Atlantic Research Centre
Defence Research and Development Canada
9 Grove Street
P.O. Box 1012
Dartmouth, Nova Scotia B2Y 3Z7
Canada

2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)

CAN UNCLASSIFIED

2b. CONTROLLED GOODS

NON-CONTROLLED GOODS
DMC A

3. TITLE (The document title and sub-title as indicated on the title page.)

Deeply Submerged Hydrodynamic, Control, Propulsion, and Dynamic Change Models for Underwater Vehicle Simulation

4. AUTHORS (Last name, followed by initials – ranks, titles, etc., not to be used)

Watt, G. D.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total pages, including Annexes, excluding DCD, covering and verso pages.)	6b. NO. OF REFS (Total references cited.)
November 2019	84	21

7. DOCUMENT CATEGORY (e.g., Scientific Report, Contract Report, Scientific Letter.)

Reference Document

8. SPONSORING CENTRE (The name and address of the department project office or laboratory sponsoring the research and development.)

DRDC – Atlantic Research Centre
Defence Research and Development Canada
9 Grove Street
P.O. Box 1012
Dartmouth, Nova Scotia B2Y 3Z7
Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
01ef - More Navy	

10a. DRDC PUBLICATION NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC-RDDC-2019-D160	

11a. FUTURE DISTRIBUTION WITHIN CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

Public release

11b. FUTURE DISTRIBUTION OUTSIDE CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

12. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Use semi-colon as a delimiter.)

Underwater vehicle hydrodynamics; Submarines; Vehicle control; Propulsion; Vehicle Dynamics

13. ABSTRACT (When available in the document, the French version of the abstract must be included here.)

In 2011, a DRDC Technology Investment Fund project was initiated to develop a concept for reliably docking unmanned underwater vehicles with a slowly moving submerged submarine. A contractor, Dynamic Systems Analysis Ltd. (DSA), began developing a simulation to evaluate the concept using their own and additional algorithms from DRDC and other contractors. This report documents the deeply submerged hydrodynamics, control, propulsion, and dynamic change models DRDC provided DSA. The report has evolved over a multi-year period to incorporate feedback from DSA.

En 2011, on a lancé un projet du Fonds d'investissement technologique de RDDC en vue de concevoir un moyen d'amarrer avec fiabilité des véhicules sous-marins sans équipage (VSSE) à un sous-marin immergé se déplaçant lentement. Un entrepreneur, à savoir Dynamic Systems Analysis ltée (DSA), a entrepris l'élaboration d'une simulation visant à évaluer ce moyen à l'aide de ses propres algorithmes ainsi que d'algorithmes additionnels provenant de RDDC et d'autres entrepreneurs. Le présent rapport documente les modèles relatifs à l'hydrodynamique en immersion profonde, à la manoeuvre, à la propulsion et aux changements dynamiques que RDDC a fournis à DSA. On l'a adapté au fil des ans afin d'y inclure la rétroaction de DSA.