ISSN 0280–5316
ISRN LUTFD2/TFRT--5541--SE

Cross-coupling Effects in Submarine Control

Hans Nilsson

Department of Automatic Control
Lund Institute of Technology
November 1995

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
November 1995

Document Number
ISRN LUTFD2/TFRT--5541--SE

Author(s)
Hans Nilsson

Supervisor
Rolf Johansson, LTH and Ola Dahl, Kockums AB

Sponsoring organisation

Title and subtitle
Cross-coupling Effects in Submarine Control. (Korskopplingseffekter i ubåtsreglering.)

Abstract

Depth and course control of a submarine are traditionally handled by two separate controllers. A practical problem is that coupling effects between depth and course makes the depth controller unable to work properly during course change. In order to investigate this problem a simulator has been developed. Simulations show that a depth error occurs during course change. The magnitude of the error depends on surge speed and course rudder angle.

It has been observed in practice that manual depth control during course change can be improved by allowing an initial pitch into the turn. An approach to solve the problem of cross-coupling between course and depth is therefore to introduce a pitch angle reference for the depth controller. This idea is elaborated and investigated by simulations. The result is a control scheme which reduces the depth error.

Key words
Submarine control, cross-coupling, pitch angle references

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
81

Recipient's notes

Security classification

Contents

Abstract

Depth and course control of a submarine are traditionally handled by two separate controllers. A practical problem is that coupling effects between depth and course makes the depth controller unable to work properly during course change. In order to investigate this problem a simulator has been developed. Simulations show that a depth error occurs during course change. The magnitude of the error depends on surge speed and course rudder angle.

It has been observed in practice that manual depth control during course change can be improved by allowing an initial pitch angle when going into the turn. An approach to solve the problem of cross-coupling between course and depth is therefore to introduce a pitch angle reference for the depth controller. This idea is elaborated and investigated by simulations. The result is a control scheme which reduces the depth error.

KEY WORDS: Submarine control, cross-coupling, pitch angle references

List of Variable Names

Abbreviations

b.f.c.s.	body-fixed coordinate system
CB	Center of Buoyancy
CG	Center of Gravity
c.s.	coordinate system
e.f.c.s.	earth-fixed coordinate system
rps	Revolutions Per Second. (turning speed of propeller axis)

Symbols

B	Buoyancy force
F_e	External force acting on the submarine
$J_1(\phi, \theta, \psi)$	Transformation matrix (Velocity)
$J_2(\phi, \theta, \psi)$	Transformation matrix (Angular velocity)
M_e	Momentum due to external forces
m	Submarine mass
p	Angular velocity around body-fixed x-axis
q	Angular velocity around body-fixed y-axis
r	Angular velocity around body-fixed z-axis
u	Velocity component in body-fixed x-axis direction. (SURGE)
v	Velocity component in body-fixed y-axis direction. (SWAY)
V_b	Velocity vector of CG expressed in body-fixed coordinates. $V_b = [u\ v\ w]^T$
V_e	Velocity vector of CG expressed in earth-fixed coordinates. $V_e = [\dot{x}_e\ \dot{y}_e\ \dot{z}_e]^T$
W	Submarine weight
w	Velocity component in body-fixed z-axis direction. (HEAVE)
ω_b	Angular velocity vector in body-fixed coordinates. $\omega_b = [p\ q\ r]^T$
ω_e	Euler-rate vector. $\omega_e = [\dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T$
x	X-axis in body-fixed c.s.
X_e	X-axis in earth-fixed c.s
x_e	X_e coordinate
$\dot{x}_e$	CG velocity component in X_e (earth-fixed) direction
x_G	CG's x-coordinate in the body-fixed c.s.

Symbols Continued

y	Y-axis in body-fixed c.s.
Y_e	Y-axis in earth-fixed c.s
y_e	Y_e coordinate
$\dot{y}_e$	CG velocity component in Y_e direction
y_G	CG's y-coordinate in the body-fixed c.s.
z	Z-axis in body-fixed c.s.
Z_e	Z-axis in earth-fixed c.s.
z_e	Z_e coordinate
$\dot{z}_e$	CG velocity component in Z_e direction
z_G	CG's z-coordinate in the body-fixed c.s.

Greek Alphabet

δ_b	Bowplane angle
δ_r	Rudder angle
δ_s	Sternplane angle
θ	PITCH angle
$\dot{\theta}$	PITCH angle velocity
ϕ	ROLL angle
$\dot{\phi}$	ROLL angle velocity
ψ	YAW angle
$\dot{\psi}$	YAW angle velocity

State Variables in Nonlinear Simulation Model

x[0]	u
x[1]	v
x[2]	w
x[3]	p
x[4]	q
x[5]	r
x[6]	x
x[7]	y
x[8]	z
x[9]	ϕ
x[10]	θ
x[11]	ψ

Additional Simulation Variables

U0	Initial value of x[0]

Simulation Parameters

Min Step Size	Minimum step size used by selected integration method
Max Step Size	Maximum step size used by selected integration method
Tolerance	relative local error permitted
Eq. Solver	Integration method. Euler, rk23, rk45, linsim, adams or gear.

1. Introduction

This Masters Thesis has been done at the Department of Automatic Control at Lund's Institute of Technology, in cooperation with Kockums Submarine Systems AB. It is a 16 week project tutored by Ola Dahl at Kockums and Rolf Johansson at LTH.

1.1 Problem Description

In a real submarine there is a problem keeping the desired depth during course change. This is partly due to the design of the depth regulator, which is based on a linearized model of a submarine. This design does not take into consideration the cross coupling effects that exist between depth and course.
A way to eliminate the depth error during course change, while keeping the current design of the depth controller, is investigated.

1.2 Report Outline

In Chapter 2 a general introduction to submarine dynamics is given, followed by a short explanation of the David Taylor Standard Submarine Equations. Chapter 3 is a description of the linearized submarine model used when designing the depth regulator. This regulator is a state feedback controller, computed using LQ optimization.
The nonlinear submarine model is introduced in Chapter 4. The depth controller, as described in chapter three, is tested on the nonlinear model. Also introduced is a gain scheduling scheme, to compensate for the dependence of the surge speed. The cross coupling effects due to a manual rudder angle are then presented. The chapter is ended with an analysis of how the cross-coupling effects may be viewed. Chapter 5 presents two different methods for elimination of the cross-coupling effects. One is a state feedback method and the other is a pitch reference design based on ideas from the analysis given in chapter four. The proposed methods are evaluated with regard to practical limitations and expectations of the system. Chapter 6 summarizes the results found and conclusions are drawn.

2. Submarine Dynamics

A short explanation of the Standard Submarine Equations of Motion from the Naval Surface Warfare Center (formerly David Taylor) [1]is presented [4]. These equations are the foundation on which the submarine simulator is built, though deviations occur due to recommendations from Kockums.

2.1 Coordinate Systems

Coordinate Systems and Transformations

To describe the motion of a submarine submerged in a fluid, relative to earth, two different coordinate systems are used. The first coordinate system (c.s.) is an earth-fixed system $X_e Y_e Z_e$. This c.s. is shown in Figure 2.1.

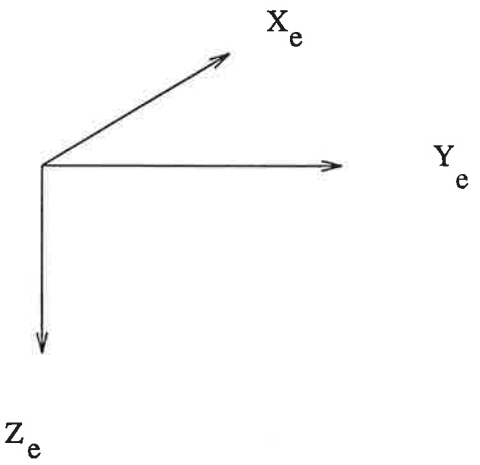

Figure 2.1 Earth-fixed coordinate system

1 WWW address: http://www50.dt.navy.mil/

One representation of the submarine is by its Center-of-Gravity (CG). This point is located at x_e, y_e, z_e relative to the earth-fixed c.s., as shown in Figure 2.2.

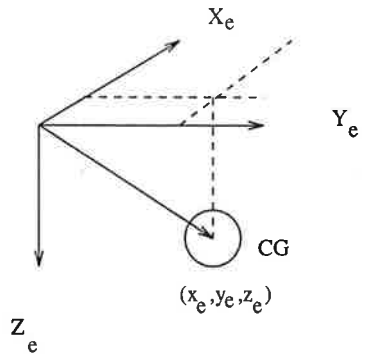

Figure 2.2 Submarine CG in $X_eY_eZ_e$

The second coordinate system is fixed in the submarine. This c.s., which is called the body-fixed c.s., is shown in Figure 2.3 and Figure 2.4.

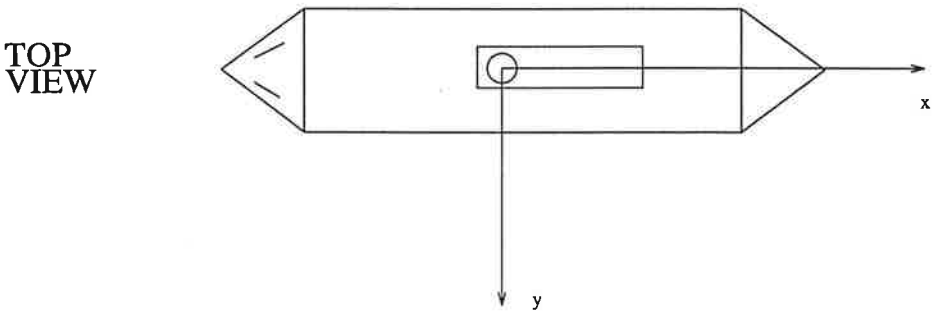

Figure 2.3 Submarine from above

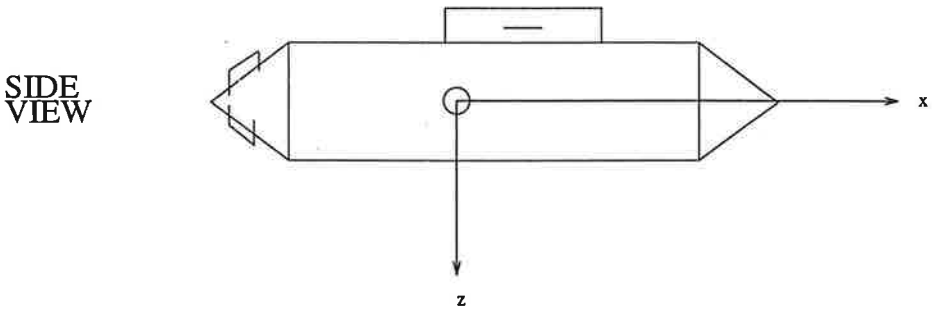

Figure 2.4 Submarine from side

The coordinates of the CG in the body-fixed c.s. are (x_G, y_G, z_G). In figures 2.3 and 2.4 the body-fixed origin coincide with CG and then $(x_G, y_G, z_G) = (0, 0, 0)$.
The submarine can turn around all three axis x,y,z of the body-fixed c.s. The different movements can be described in the following way: Imagine that you sit in the submarine facing the front with your head up along the periscope. If you turn the submarine right/left then you increase/decrease YAW angle (ψ). If, on the other hand, you decide to point

your nose upwards/downwards, you increase/decrease PITCH angle (θ). Finally, if you lean right/left, then you increase/decrease ROLL angle (ϕ).

CONCLUSION: Position and orientation of the submarine can be described with the variables x_e, y_e, z_e (position in the earth-fixed c.s), and ψ,θ,ϕ (yaw, pitch and roll angles).

Velocity

The velocity of the CG relative to the earth-fixed c.s is represented by the vector

$$V_e = (\dot{x}_e, \dot{y}_e, \dot{z}_e)^T \tag{2.1}$$

The velocity of the CG expressed in body-fixed coordinates is denoted V_b. Its components are u,v and w i.e.

$$V_b = (u, v, w)^T \tag{2.2}$$

The angular velocity is represented in the body-fixed c.s. as:

$$\omega_b = (p, q, r)^T \tag{2.3}$$

A last definition is needed and that is the Euler-rate vector w_e

$$\omega_e = (\dot{\phi}, \dot{\theta}, \dot{\psi})^T \tag{2.4}$$

Mathematical tools for transformation between the two c.s.

Transformations that, knowing V_b (body related velocity) and ω_b (body related angular velocity), give V_e (earth related velocity) and ω_e (Euler-rate vector), are sought. In other words functions such that:

$$J_1(\phi, \theta, \psi): V_b \rightarrow V_e \qquad J_2(\phi, \theta, \psi): \omega_b \rightarrow \omega_e$$

Mathematical preliminaries GOAL: Given a vector V in the c.s. shown in Figure 2.5,

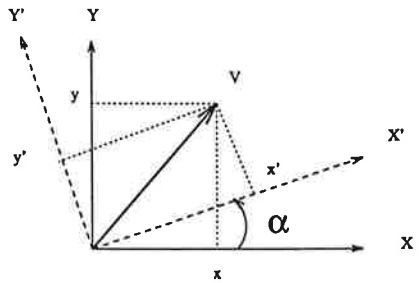

Figure 2.5 Vector V represented in old c.s. (solid) and new c.s. (dotted)

with coordinates x,y in the X-Y system (old system) and the coordinates x',y' in the X'-Y' system (new system), it is seen that the transformation from x',y' to x,y is given by

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

To find the coordinates of the vector V in the new c.s we invert the matrix above and find that:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Another way of visualizing this can be seen in Figure 2.6:

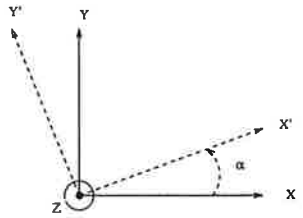

Figure 2.6 Old (solid) and new (dotted) coordinate axis when turning an angle α around the z-axis

Turn the old c.s a positive angle α around the z-axis, naming it the new c.s at the new position. The coordinate transformation then becomes

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = C_{Z,\alpha} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.5}$$

Keeping track of directions, the following c.s is found when turning around the y-axis a positive angle α:

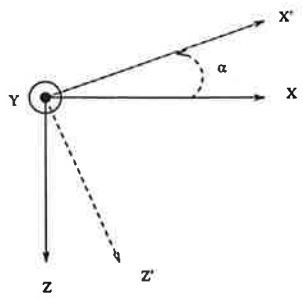

Figure 2.7 Old (solid) and new (dotted) coordinate axis when turning an angle α around the y-axis

A comparison with Figure 2.6 shows that

$$C_{Y,\alpha} = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix}$$

Finally turning around the x-axis gives the matrix:

$$C_{X,\alpha} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix}$$

Submarine Velocity Representation It is standard to use the angles ψ, θ and ϕ to represent the orientation of the submarine relative the earth-fixed c.s. This is done in the following way:

Starting from the earth-fixed c.s and the vector $V_e = [\dot{x}_e\ \dot{y}_e\ \dot{z}_e]^T$, the following steps are performed:

1. Rotate a yaw angle ψ around the Z-axis. This gives a new c.s, where the coordinates of V_e are given by: $\rightarrow V_1 = C_{Z,\psi} V_e$
2. Rotate the new c.s a pitch angle θ around the Y-axis: $\rightarrow V_2 = C_{Y,\theta} V_1 = C_{Y,\theta} C_{Z,\psi} V_e$
3. Rotate this new c.s a roll angle ϕ around the X-axis: $\rightarrow V_3 = C_{X,\phi} V_2 = C_{X,\phi} C_{Y,\theta} C_{Z,\psi} V_e$

The resulting c.s, achieved after the three operations yaw, pitch and roll, as described above, represents the submarine's orientation. The resulting c.s. thus coincides with the body-fixed c.s. Since $V_b = [u\ v\ w]^T$ is the submarine velocity, represented in body-fixed coordinates, this gives $V_b = V_3$, i.e.

$$V_b = C_{X,\phi} C_{Y,\theta} C_{Z,\psi}\, V_e$$

Most of the time interest will focus on finding earth-fixed coordinates, given body-fixed coordinates. Since the matrices $C_{X,\phi}$,$C_{Y,\theta}$ and $C_{Z,\psi}$ are orthogonal, that is $C^T_{X,\phi} C_{X,\phi} = \mathrm{I}$ so that $C^{-1}_{X,\phi} = C^T_{X,\phi}$ etc., this can be written as :

$$V_e = C^T_{Z,\psi} C^T_{Y,\theta} C^T_{X,\phi} V_b = J_1(\phi,\theta,\psi) V_b$$

The matrix $J_1(\phi,\theta,\psi)$, when written with s = sin and c = cos , looks like [2]:

$J_1(\phi,\theta,\psi) =$

$$\begin{pmatrix} c(\psi)c(\theta) & -s(\psi)c(\phi) + c(\psi)s(\theta)s(\phi) & s(\psi)s(\phi) + c(\psi)c(\phi)s(\theta) \\ s(\psi)c(\theta) & c(\psi)c(\phi) + s(\phi)s(\theta)s(\psi) & -c(\psi)s(\phi) + s(\theta)s(\psi)c(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{pmatrix} \qquad (2.6)$$

2 The coordinate transformation can also be found in [4] and[5]

Submarine Angular Velocity Representation There is also a need for a transformation between the body-fixed angular velocity $\omega_b = (p, q, r)^T$ and the Euler-rate vector $\omega_e = (\dot{\phi}, \dot{\theta}, \dot{\psi})^T$ such that

$$\omega_e = J_2(\phi, \theta, \psi)\omega_b \tag{2.7}$$

$J_2(\phi, \theta, \psi)$ is found by the following reasoning:
Assume that the three rotations *yaw*(ψ), *pitch*(θ) and *roll*(ϕ), as described by steps 1, 2 and 3 above, have been done and that the new coordinate system is oriented the same way as the body-fixed c.s.

If $\dot{\theta}$=$\dot{\psi}$=0 and $\dot{\phi} \neq 0$, it can be seen that the angular velocity ω_b is a pure rotation around the x-axis;

$$\omega_b = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix}$$

Suppose instead that $\dot{\phi}$=$\dot{\psi}$=0 and $\dot{\theta} \neq 0$. Then ω_b is a rotation around the y-axis. But this y-axis belongs to the intermediate coordinate system in step two above. Therefore make step three to find the angular velocity in body-fixed coordinates.
This gives

$$\omega_b = C_{X,\phi} \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix}$$

The same reasoning is used when $\dot{\phi}$=$\dot{\theta}$=0, $\dot{\psi} \neq 0$. This is a rotation around the z-axis belonging to the first coordinate system above (step one). To obtain the angular velocity therefore rotate the system twice (steps 2 and 3). The result is

$$\omega_b = C_{X,\phi} C_{Y,\theta} \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix}$$

Add these results to find $\omega_b = (p, q, r)^T$.

$$\omega_b = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + C_{X,\phi} \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + C_{X,\phi} C_{Y,\theta} \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} = S(\phi, \theta, \psi)\omega_e$$

where

$$S(\phi,\theta,\psi) = \begin{pmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{pmatrix}$$

which leads to [3]:

$$J_2(\phi,\theta,\psi) = S(\phi,\theta,\psi)^{-1} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{pmatrix} \tag{2.8}$$

Notice that the matrix $S(\phi,\theta,\psi)$ is singular if $\theta = 90$ deg ($\frac{\pi}{2}$ rad) for example.

2.2 Forces and Moments

The six variables $(x_e(t), y_e(t), z_e(t), \phi(t), \theta(t), \psi(t))$ describe the position and orientation of the submarine as a function of time t. They can be found with six equations, three force equations and three moment equations.

The external force acting on the submarine is F_e=$[F_x, F_y, F_z]^T$. It represents e.g. propeller thrust and hydrodynamic forces due to water turbulence and rudder angles. To find F_x,F_y and F_z, use Figure 2.8

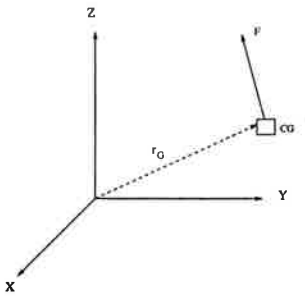

Figure 2.8 Body-fixed c.s situated in the submarine

and the relationship that gives the time derivative in earth-fixed coordinates when given body-oriented variables:

$$\frac{d}{dt_e}A = \frac{d}{dt_b}A + \omega \times A \tag{2.9}$$

A = variable e.g. position or velocity in body-coordinates.
d/dt_e= derivative in earth-fixed c.s.
d/dt_b= derivative in body-fixed c.s.
ω= angular velocity in body-fixed c.s.

---

3 Can also be found in [4] and [5]

An explanation of 2.9 can be found in [9].

GOAL:*Derive an expression for the force*:

$$F_e = ma_e$$

Knowing the vector r_G, (the distance between the origin of the body-fixed c.s. and the CG of the submarine) as shown in figure 2.8, and using equation 2.9, the velocity of the submarine can be expressed in the earth-fixed c.s, as:

$$v_e = v_b + \omega \times r_G$$

and the acceleration as:

$$a_e = \frac{d}{dt_e}v_e = \frac{d}{dt_b}v_e + \omega \times v_e = \dot{v}_b + \dot{\omega} \times r_G + \omega \times \dot{r}_G + \omega \times v_e = \dot{v}_b + \dot{\omega} \times r_G + \omega \times (v_b + \omega \times r_G)$$

The force can then be seen as:

$$\boxed{m(\dot{v}_b + \omega \times v_b + \dot{\omega} \times r_G + \omega \times (\omega \times r_G)) = F_e}$$

With $v_b = (u, v, w)^T$, $\dot{v}_b = (\dot{u}, \dot{v}, \dot{w})^T$, $\omega = (p, q, r)^T$, $\dot{\omega} = (\dot{p}, \dot{q}, \dot{r})^T$, r_G=(x_G, y_G, z_G) and m=submarine weight, the boxed expression can be expanded to find the left hand side of the three David Taylor (DT) force equations [4].The equations are also given in Appendix A.

Find three momentum equations The angular momentum is defined as L = $I_0\omega$ where ω is the angular velocity and I_0 the moment of inertia matrix.

Lets illustrate the following discussion with Figure 2.9.

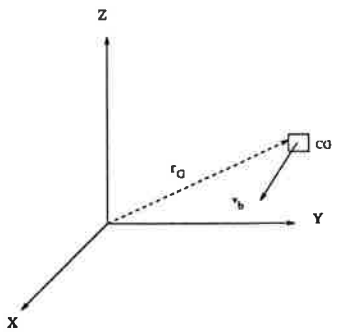

Figure 2.9 The CG with a speed V_b represented in the body-fixed c.s.

If r_G=0 then:

$$M_e = \frac{d}{dt}L = \frac{d}{dt_e}(I_0\omega) = [equation\ 2.9] = I_0\dot{\omega} + \omega \times (I_0\omega) \qquad (2.10)$$

If, on the other hand, $r_G \neq 0$ then an extra moment is needed:

$$M_e = r_G \times F = r_G \times m\frac{d}{dt_e}v_b = r_G \times m(\dot{v}_b + \omega \times v_b) \tag{2.11}$$

Put together the moment can be seen as:

$$\boxed{mr_G \times \dot{v}_b + mr_G \times (\omega \times v_b) + I_0\dot{\omega} + \omega \times (I_0\omega) = M_e}$$

With $v_b = (u, v, w)^T$, $\dot{v}_b = (\dot{u}, \dot{v}, \dot{w})^T$, $\omega = (p, q, r)^T$, $\dot{\omega} = (\dot{p}, \dot{q}, \dot{r})^T$, r_G=(x_G, y_G, z_G) and m=submarine weight the boxed expression can be expanded to find the left hand side of the three David Taylor (DT) momentum equations found in [4] and in Appendix A.

F_e and M_e are found through investigations of forces affecting the submarine, such as waves, currents,skin friction etc. A short presentation of these can be found in the next section.

2.3 The David Taylor Standard Submarine Equations of Motion

On the right hand side of the DT equations a number of forces and moments are found. Among others there are added mass forces and forces due to the specific hull construction. An extensive treatment of these is outside the scope of this report.
A few specific details worth mentioning though, are:

1. The equations are normalized. Consider e.g. the axial force equation F_x which includes the dimensionless constant X'_{qq}. Suppose that the force F_x depends on the angular velocity component q, and that a Taylor expansion of F_x is done. This gives $F_x \approx ... + X_{qq}q^2 + ...$ with $F_x[\frac{kgm}{s^2}]$ and X_{qq}[kgm] and $q^2[\frac{1}{s^2}]$. The dimensionless constant $X_{qq}{}'$ is then obtained by multiplication with $\frac{2}{\rho L^4}$-$[\frac{1}{kgm}]$.
   In other words, the factor $\frac{\rho}{2}L^4$ appears in the axial force equation to make the dimensions right.
2. F_{XP} (In equation F_x, Appendix A)and Q_p (In equation M_{roll}, Appendix A) models the propeller.
3. C_d (In equations F_y,F_z,M_{pitch} and M_{yaw}, Appendix A): Coefficient used in integrating forces and moments along the hull due to local cross-flow.
4. $\overline{C}_L$(In equations F_y,F_z,M_{roll},M_{pitch} and M_{yaw}, Appendix A): Lift-curve slope used in computing the effects of the hull-bound vortex due to lift on the bridge fairwater.
5. C (In equations F_x,F_y,F_z,M_{roll},M_{pitch} and M_{yaw}, Appendix A):Variable coefficient used in scaling model thrust and drag data to full-scale.

3. Linear Control Design

3.1 Introduction

Control of depth and course is handled by two separate linear controllers. The controller parameters are computed using a linearized model of the submarine.
The depth controller has two output signals (sternplane δ_s (aft rudder) and bowplane δ_b (front rudder)) and four input signals (depth z, speed in normal direction w, pitch angle θ and angular velocity around the y-axis q). The course regulator, which will not be treated here, is often a PID-regulator. It has one output signal (rudder δ_r (aft rudder)).

Since the problem investigated in this master thesis is depth-keeping, the design of the depth regulator will be shown in detail.
The chapter starts with the linearization of the nonlinear submarine model and then goes on to describe the depth-regulator design. Then the linear open loop system and the linear closed loop system are simulated using Simulink.

3.2 Linear Model for Depth Control

Using the David Taylor differential equations (see appendix A) a linear model for depth control analysis is sought.

A linear model can be found in two ways. One is to linearize the submarine around an equilibrium point and by doing that getting access to all the powerful tools of linear theory. This method will be used here. Another method, which utilizes the structure of the nonlinear equations, is exact linearization [8].

Linearization

To simplify the model of the submarine, a little knowledge of its behavior is needed. The six degrees of freedom (u,v,w,p,q,r) can be reduced to three if assuming that the submarine only moves forward and up/down. While moving up/down it can be observed that the pitch angle θ changes (the same phenomena can be observed in an airplane), so the model should allow for movements around the y-axis (pitch). Two equations, from Appendix A, can be used to describe this simple model: the normal force equation (F_z) and the pitching moment equation (M_{pitch}). They look as follows.

Normal Force equation F_z:

$$
\begin{aligned}
&m[\dot{w} - uq + vp + x_G(rp - \dot{q}) + y_G(rq + \dot{p}) - z_G(p^2 + q^2)] = \\
&\tfrac{\rho}{2} l^4 [Z'_{\dot{q}} \dot{q}] \\
&\tfrac{\rho}{2} l^3 [Z'_{\dot{w}} \dot{w} + Z'_q uq + Z'_{vp} vp + Z'_{vr} vr] + \\
&\tfrac{\rho}{2} l^2 [Z'_* u^2 + Z'_w uw] + Z'_{vv} vv \\
&\tfrac{\rho}{2} l^2 [Z'_{|w|} u|w| + Z'_{ww} |w(v^2 + w^2)^{1/2}| + \\
&\tfrac{\rho}{2} l^2 [Z'_{\delta_s} u^2 \delta_s + Z'_{\delta_b} u^2 \delta_b + Z'_{\delta_s \eta} u^2 \delta_s (\eta - 1/C) C] -
\end{aligned}
$$

$\frac{\rho}{2} C_d \int_l b(x)\, w(x)\{[w(x)]^2 + [v(x)]^2\}^{1/2} dx +$
$(W - B)\cos\theta\cos\phi$

Pitching moment equation M_{pitch}:

$I_y\dot{q} + (I_x - I_z)rp - (\dot{p} + qr)I_{xy} + (p^2 - r^2)I_{xz} + (qp - \dot{r})I_{yz} + m[z_G(\dot{u} - vr + wq) - x_G(\dot{w} - uq + vp)] =$
$\frac{\rho}{2} l^5 [M'_{\dot{q}}\dot{q} + M'_{rp}rp] +$
$\frac{\rho}{2} l^4 [M'_{\dot{w}}\dot{w} + M'_q uq + M'_{vr}vr] +$
$\frac{\rho}{2} l^3 [M'_* u^2 + M'_w uw + M'_{w|w|_R} w|(v^2 + w^2)|^{1/2}] +$
$\frac{\rho}{2} l^3 [M'_{\delta_s} u^2\delta_s + M'_{\delta b} u^2\delta_b + M'_{\delta_s \eta} u^2\delta_s(\eta - \frac{1}{C})C] +$
$\frac{\rho}{2} C_d \int_l x\, b(x)\, w(x)\{[w(x)]^2 + [v(x)]^2\}^{1/2} dx -$
$(x_G W - x_B B)\cos\theta\cos\phi - (z_G W - z_B B)\sin\theta$

Restricting the degrees of freedom leads to the following approximations:

1. r=0 No angular velocity around the z-axis
2. p=0 No angular velocity around the x-axis
3. v=0 No velocity component in the y direction.

For the type of submarine studied here, the following assumptions are common engineering practice:

1. u is assumed constant. The depth controller is thus designed for constant speed. Varying speed is taken care of by gain-scheduling.
2. There is no displacement in the y- and x-direction of CB (Center of Buoyancy) and CG (Center of Gravity), i.e. $x_G = x_B = y_G = y_B = 0$
3. The inertia matrix is diagonal, i.e. I_{xy}=I_{xz}=I_{yz}=0, since the body-fixed c.s. coincides with the main spin axes.
4. C is a variable coefficient used in scaling model thrust and drag to full scale and is not used in this model.
5. M_* is a lift factor due to unsymmetric hull. M_*=0.
6. Terms with absolute value are damping coefficients and they are not needed in this model.
7. The roll motion is neglected, i.e $\cos\phi \approx 1$
8. W=B, i.e. the weight of the submarine is the same as the Buoyancy force, in other words assume that the submarine is well balanced at a certain depth.
9. $\frac{\rho}{2} C_d \int_l b(x)\, w(x)\{[w(x)]^2 + [v(x)]^2\}^{1/2} dx$ and
   $\frac{\rho}{2} C_d \int_l x\, b(x)\, w(x)\{[w(x)]^2 + [v(x)]^2\}^{1/2} dx$ models cross-flow and can be neglected.
10. q small $\Rightarrow q^2 \approx 0$
11. w small $\Rightarrow wq \approx 0$
12. Small pitch angles, i.e. $\sin\theta \approx \theta$

These assumptions hold if the submarine is moving at a relatively large depth, and does not undertake any abrupt motions.

The normal force equation and the pitching moment equation are now simplified.

Normal Force equation:

$m[\dot{w} - uq] =$
$\frac{\varrho}{2} l^4 Z'_{\dot{q}} \dot{q} +$
$\frac{\varrho}{2} l^3 [Z'_{\dot{w}} \dot{w} + Z'_q uq] +$
$\frac{\varrho}{2} l^2 [Z'_w uw +$
$\frac{\varrho}{2} l^2 [Z'_{\delta_s} u^2 \delta_s + Z'_{\delta_b} u^2 \delta_b]$

Pitching moment equation:

$I_y \dot{q} =$
$\frac{\varrho}{2} l^5 [M'_{\dot{q}} \dot{q}] +$
$\frac{\varrho}{2} l^4 [M'_{\dot{w}} \dot{w} + M'_q uq] +$
$\frac{\varrho}{2} l^3 [M'_w uw] +$
$\frac{\varrho}{2} l^3 [M'_{\delta_s} u^2 \delta_s + M'_{\delta_b} u^2 \delta_b] -$
$(Z_G W - Z_B B)\theta$

This can be written as

$$\begin{bmatrix} m - \frac{\varrho}{2} l^3 Z'_{\dot{w}} & -\frac{\varrho}{2} l^4 Z'_{\dot{q}} \\ -\frac{\varrho}{2} l^4 M'_{\dot{w}} & I_y - \frac{\varrho}{2} l^5 M'_{\dot{q}} \end{bmatrix} \begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} =$$

$$\begin{bmatrix} \frac{\varrho}{2} l^2 Z'_w & m + \frac{\varrho}{2} l^3 Z'_q \\ \frac{\varrho}{2} l^3 M'_w & \frac{\varrho}{2} l^4 M'_q \end{bmatrix} \begin{bmatrix} w \\ q \end{bmatrix} u + \begin{bmatrix} \frac{\varrho}{2} l^2 Z'_{\delta_s} & \frac{\varrho}{2} l^2 Z'_{\delta_b} \\ \frac{\varrho}{2} l^3 M'_{\delta_s} & \frac{\varrho}{2} l^3 M'_{\delta_b} \end{bmatrix} \begin{bmatrix} \delta_s \\ \delta_b \end{bmatrix} u^2 + \tag{3.1}$$

$$\begin{bmatrix} 0 \\ -(Z_G W - Z_B B) \end{bmatrix} \theta$$

Introducing

$$y_l = \begin{pmatrix} w \\ q \end{pmatrix} \qquad \delta = \begin{pmatrix} \delta_s \\ \delta_b \end{pmatrix}$$

this system can be written as:

$$M\dot{y}_l = Cuy_l + Du^2\delta + G\theta \tag{3.2}$$

which can be transformed into:

$$\dot{y}_l = M^{-1}Cuy_l + M^{-1}Du^2\delta + M^{-1}G\theta \tag{3.3}$$

In order to obtain a complete state space model, differential equations for θ and z are needed. Equations 2.6 and 2.8 gives:

$$\dot{\theta} = \cos\theta q - r\sin\phi$$
$$\dot{z} = -u\sin\theta + v\cos\theta\sin\phi + w\cos\theta\cos\phi$$

Under the same assumptions as before they are written:

$$\dot{\theta} = q \qquad \dot{z} = w - u\theta \tag{3.4}$$

Introduce the state vector x_l as:

$$x_l = [\ w\ q\ \theta\ z\ ]^T$$

Combining 3.3 and 3.4 then gives:

$$\dot{x}_l = Ax_l + B\delta$$

where

$$A = \begin{pmatrix} M^{-1}Cu & & M^{-1}G & 0_{2\times 1} \\ 0 & 1 & 0 & 0 \\ 1 & 0 & -u & 0 \end{pmatrix} \qquad B = \begin{pmatrix} M^{-1}Du^2 \\ 0_{2\times 2} \end{pmatrix}$$

with the following structure :

$$A = \begin{bmatrix} a_{11}\,u & a_{12}\,u & a_{13} & 0 \\ a_{21}\,u & a_{22}\,u & a_{23} & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & -u & 0 \end{bmatrix} \qquad B = \begin{bmatrix} b_{11}\,u^2 & b_{12}\,u^2 \\ b_{21}\,u^2 & b_{22}\,u^2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Since u is assumed constant, A and B define a linear model for depth control. Expressions for the coefficients a_{ij} in A and b_{ij} in B are given in Appendix B.

3.3 Depth-regulator Design

GOAL Find a MIMO (Multiple Input, Multiple Output) controller that stabilizes the submarine and gives it good depth control characteristics. The following constraints are given:

- The pitch angle θ should not at any time be larger than 10 degrees.
- No overshoot is wanted
- Rudder angles δ_s and δ_b are limited to $\pm$ 30 degrees.

A robust design, in the sense that it works well when used with the original nonlinear model, is sought.

Solution LQ (Linear-Quadratic) methods.

Description The LQ regulator is a state feedback controller, where the poles of the closed loop system are assigned, not by the designer directly, but indirectly by minimizing a performance index. Write the system to control as

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{cases}$$

where $x(t_0)$ is given. Let matrices Q and R be nonnegative and positive definite, respectively. Define the quadratic performance index for the regulator problem as

$$V(x(t_0), u(\cdot), t_0) = \int_{t_0}^{\infty} (u^T(t)Ru(t) + x^T(t)Qx(t))dt \tag{3.5}$$

Its minimization leads to a linear state feedback law, i.e. $u(t)$ = - L $x(t)$ [2]. The optimum control can be written as

$$u(t) = -R^{-1}B^T Px(t)$$

where P is the solution to the Riccati equation $PA + A^TP - PBR^{-1}B^TP + Q = 0$ [4]

The design parameters are R, which is a weight matrix for the control signals, and Q which is a weight matrix for the states. Q is penalizing states that are not zero. If there were no limitation on the control signal the states could be brought to zero in a time T, chosen arbitrarily close to t_0= start time for control. This is not realistic and R is therefore introduced to penalize the control signals.

---

4 Matlab >> help lqr

Design (How to choose R and Q)

General information:

1. Good knowledge about the physical system is important as it will facilitate the choosing of parameters in R and Q [2].
2. For single input systems, the optimally designed regulator possesses an infinite gain margin and a phase margin > 60 degrees, i.e. good stability properties. For multiple input systems a diagonal R lets the system tolerate independent scalar gain variations between 0.5 and ∞, and phase variations less than 60 degrees in each scalar input without disturbing stability. In the nondiagonal R case there is no guarantee in any way of robustness to gain or phase variations on any single input [2].
3. For the multiple input system, having a diagonal R with entries of very different size gives poor robustness to input cross coupling [2].
4. One approach to find Q and R is to set the diagonals of Q and R to the inverse of the squared maximum allowed deviations in the corresponding variables.$(1/max^2)$
5. It is possible to make the poles of the closed system move to the stable poles of the open system by letting Q=diag(ρI), where ρ is small [2]
6. It is possible to design a LQ regulator by which the closed loop system gets prescribed eigenvalues [3]

Boat Model A nonlinear submarine model describing a typical submarine was obtained from Kockums.
Based on the supplied model, a controller was found using the ideas in the design part such that

$$Q = \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 300 & 0 \\ 0 & 300 \end{pmatrix} \tag{3.6}$$

This controller is used in all simulations.

3.4 Simulations

There are three different ways to run a simulation in the Simulink environment [6] [7]. One way is to do a graphical setup from the menu bar which gives a clear and attractive picture of how the system is organized. The second method is to work in Matlab with a model built up by Matlab-syntax. The third method is to write a mex-file in C, which describes the system and its different parts in the same way as the underlying logic which simulates the graphical model. This enables the user to define home-made SIMULINK blocks, tailored for a specific task. The two fastest methods, the graphical setup/simulation and the mex-file in C were tested. The graphical method will be used in the following simulation of the linear system.

Open Loop System

The open loop system can be drawn in the following way in Simulink (Figure 3.1).

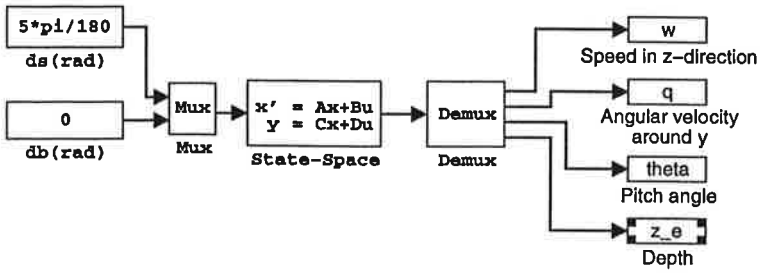

Figure 3.1 Simulink representation of the open loop system

Figure 3.2 shows the response to a positive sternplane angle (rudder deflected down).

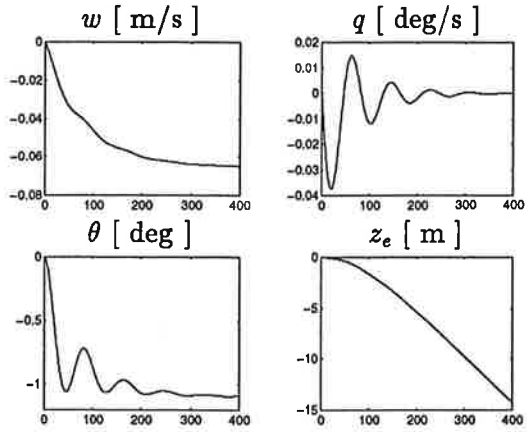

Figure 3.2 Submarine response to a 5 degree sternplane angle when u (surge speed) = 1 m/s. w = speed z-direction (body-fixed c.s.), q = angular velocity around y-axis (body-fixed c.s.), theta (θ) = pitch angle and z_e = depth

As can be seen in the lower right plot, the submarine slowly rises towards the surface. This due to the lift forces introduced by the downward deflected rudder. With a somewhat higher surge speed the same manœuver results in the following movement:

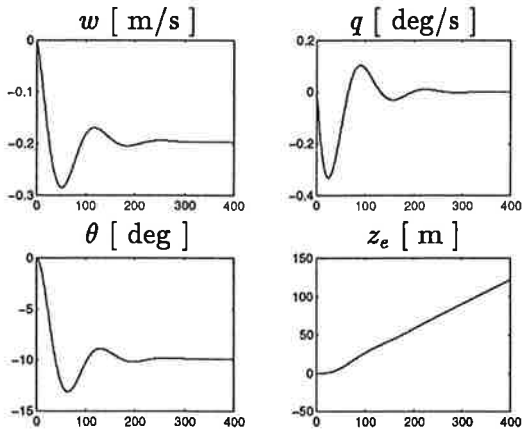

Figure 3.3 Submarine response to a 5 degree sternplane angle when u (surge speed)= 3 m/s. w = speed z-direction (body-fixed c.s.), q = angular velocity around y-axis (body-fixed c.s.), theta (θ) = pitch angle and z_e = depth

This shows that the response to a change of sternplane angle (δ_s) is different for different speeds. Above a certain speed the downward deflected sternplane will introduce a negative pitch θ large enough to make the submarine sink. The different pitch angles (θ) induced by the sternplane is clearly visible in Figures 3.3 and 3.2

Figure 3.4 shows the poles and zeros of the transfer function from sternplane (δ_s) to depth (z), for different speeds. As can be seen in the left plot the linear system is unstable for u = 12 m/s. Also note that the system is non-minimum phase.

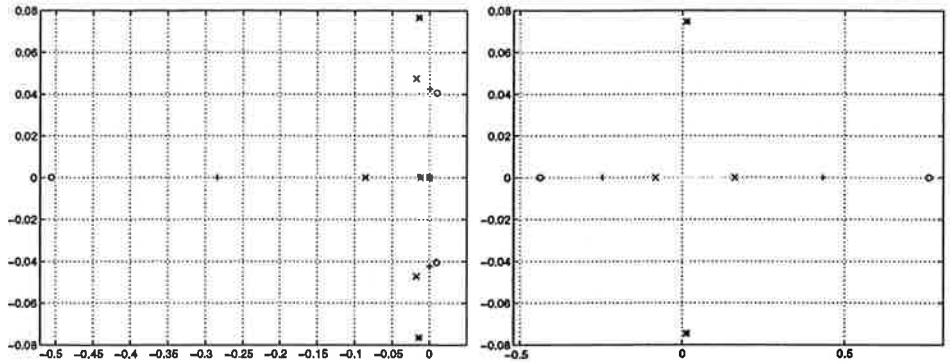

Figure 3.4 Poles (left) and zeros (right). *(1 m/s), x(3 m/s), +(7 m/s) and o(12 m/s)

Figures 3.5 and 3.6 show the response to a change of the bowplane angle. Deflecting the bowplane a positive angle δ_b is the same as deflecting it down an angle δ_b and that results in lifting forces moving the submarine upwards.

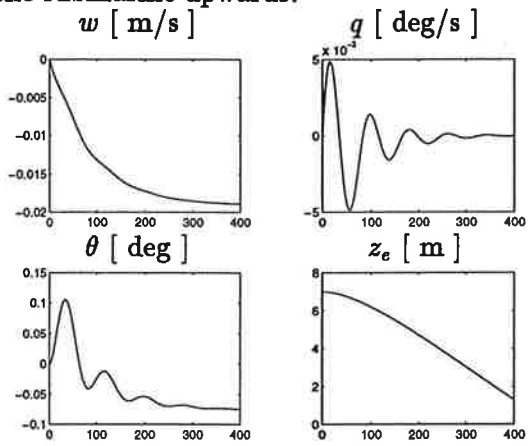

Figure 3.5 Submarine response to a 5 degree bowplane angle, u (surge speed) = 1 m/s

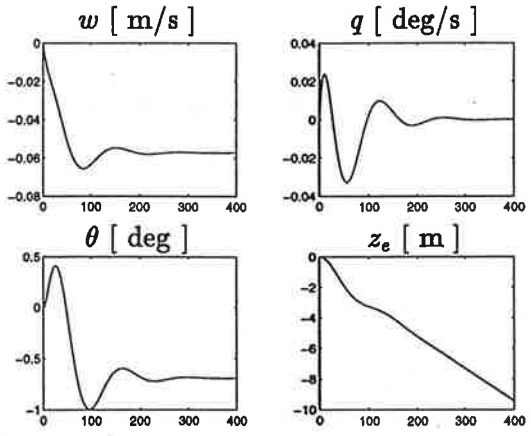

Figure 3.6 Submarine response to a 5 degree bowplane angle, u (surge speed) = 3 m/s

The poles and zeros for the transfer funktion from δ_b to depth are shown in Figure 3.7

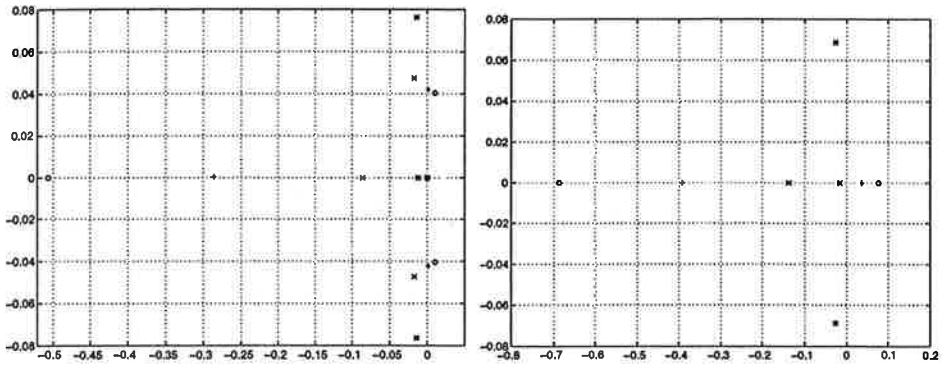

Figure 3.7 Poles (left) and zeros (right). *(1 m/s), x(3 m/s), +(7 m/s) and o(12 m/s)

The following differences in response to sternplane and bowplane changes can be seen;

- The direction of the depth response to a positive sternplane angle is speed dependent. This is not the case for the bowplane.
- The bowplane does not affect depth as much as the sternplane.
- The pitch θ is much smaller when working with the bowplane.

Closed Loop System

The closed loop system can be visualized as follows in Simulink:

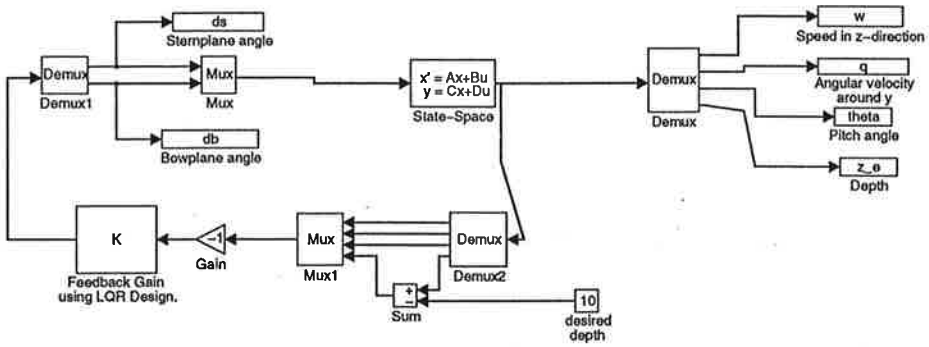

Figure 3.8 Closed loop system with linear submarine model and Lq designed depth regulator.

Using $u = 3$ m/s and the weight matrices Q and R found in the design section, a simulation where the submarine starts at zero depth and strives to go down to 10 meters can be seen in Figure 3.9:

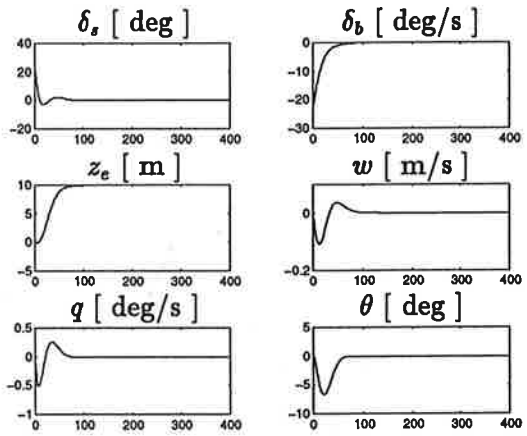

Figure 3.9 Step response when zr = depth reference is changed from 0 to 10 meters. $u = 3$ m/s

As can be seen in Figure 3.9 the system performs well.

4. Nonlinear Submarine Model and Cross Coupling Effects

4.1 Introduction

A linear model was used for computation of a depth controller in chapter three. This model does however not include any cross coupling effects between course and depth. This chapter presents simulations using the full nonlinear model, as presented in Appendix A. First the open nonlinear system is simulated, to see if any deviations compared to the linearized system are visible. Then the control loop is closed and the depth controller is investigated. In connection with this, the cross coupling effects between course and depth are shown. The chapter ends with an analysis of how the cross-coupling may be explained.

Simulation Facts

In the following chapter, the submarine is represented by a mex-file. The C-code is given in Appendix C.
In the linear model the surge speed u is constant. This is not the case in the nonlinear model, where u is a state variable. There is however an approximately linear relation between the stationary value of u and the propeller revolution rate per second (rps). For the submarine model used here 1 rps results in a stationary surge velocity of approximately 3 m/s. In order to minimize the time to stationary u, the initial value of u is therefore chosen as $3 \cdot$ rps.

Throughout the following chapters the following simulation parameters have been used:

- Min Step Size: 0.0001 sec
- Max Step Size: 1 sec
- Tolerance: 1e-3
- Equation.Solver: Runge-Kutta 5

4.2 Open Loop System

Figure 4.1 shows the response of the nonlinear system (solid line) and the linear system (dashed line) to a sternplane angle $\delta_s = 5$:

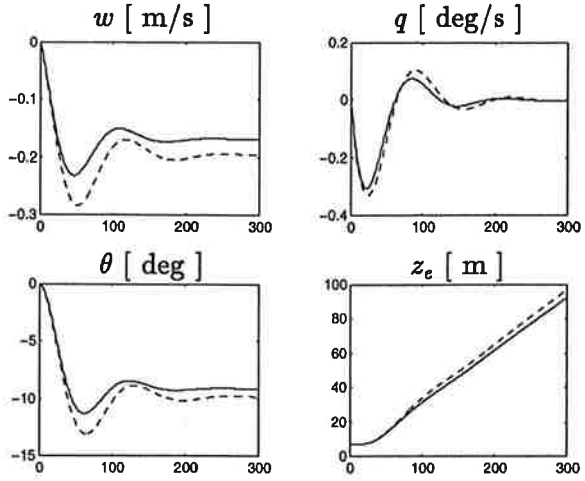

Figure 4.1 Open system given a sternplane angle (δ_s) of 5 degrees. rps=1.03 (6 knots). Nonlinear system (solid), linear system (dashed)

The surge speed in the linear model is 3 m/s. The propeller revolution rate rps was selected to obtain a stationary u as close as possible to 3 m/s. This resulted in rps = 1.03. As can be seen in Figure 4.1 the response of the linear model resembles the response of the nonlinear model.
Even closer resemblance between the linear and nonlinear systems is apparent when using the bowplane (front rudder), Figure 4.2.

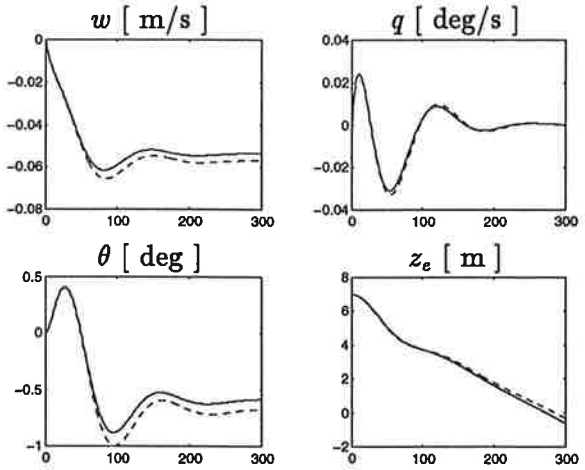

Figure 4.2 Open system given a bowplane angle (δb) of 5 degrees. rps=1.03 (6 knots) right. Nonlinear system (solid), linear system (dashed)

Cross Coupling

The result of applying a 5 degree rudder angle δ_r to the nonlinear model is shown in Figures 4.3 and 4.4. The propeller revolution rate rps = 1, which corresponds to the stationary surge speed 3 m/s. The states associated with the course change are shown in Figure 4.3. As can be seen in this figure the submarine turns with a stationary turning rate r $\approx$ -1.2 deg/s, which is an expected result of the change in course rudder.

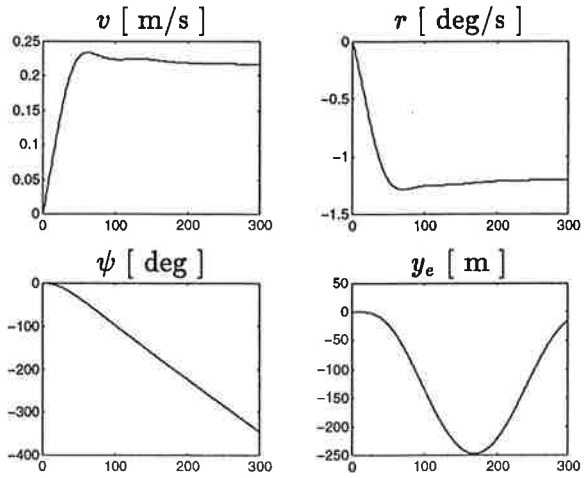

Figure 4.3 Submarine response to a 5 degree change in rudder angle (δ_r). v = speed y-direction (body-fixed c.s.), r = angular velocity around z-axis (body-fixed c.s.), ψ = yaw angle and y_e = side displacement. rps = 1

The states associated with the depth model are shown in Figure 4.4. As can be seen, the course change results in a pitch angle θ of approximately 8 degrees and a decreasing depth.

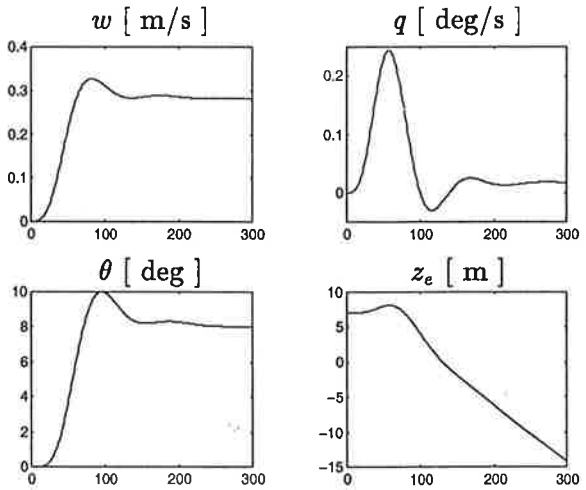

Figure 4.4 Course rudder effect on states found in the linear model. rps = 1

Conclusion The linear model used to design the depth regulator seems to have captured all essential dynamics of the nonlinear model (regarding depth change). It was then shown that the rudder (δ_r) introduces coupling effects in the open system, Figure 4.4, such that both the pitch θ and the depth are affected.

4.3 Closed System

The Simulink representation of the closed loop system with a nonlinear submarine model and linear depth controller is shown in Figure 4.5.

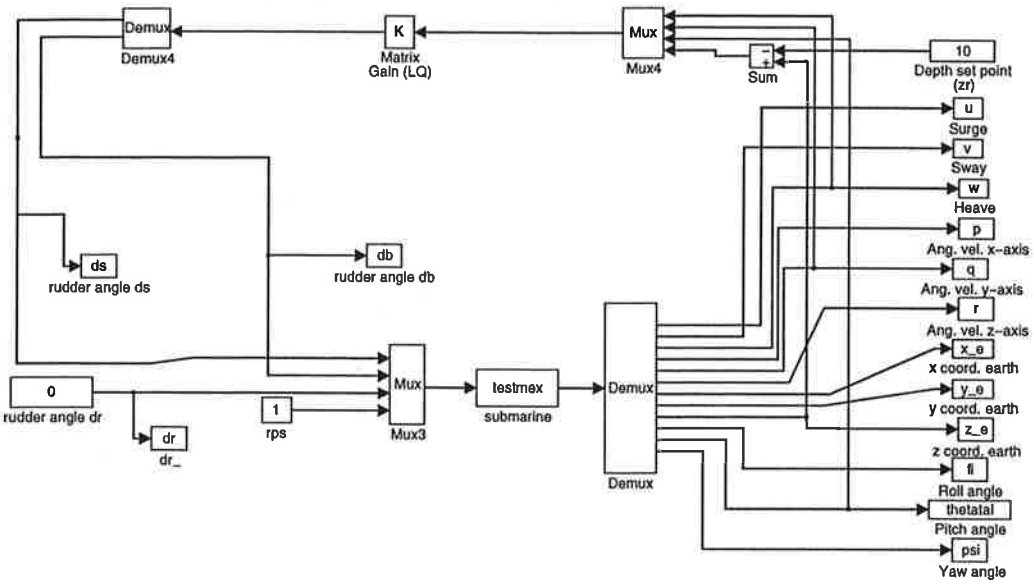

Figure 4.5 LQ-control of nonlinear submarine system

Figure 4.6 shows the response to a change in the depth set point. The initial depth is 7 meters, the initial surge speed is 3 m/s = 6 knots, the set point is $z_r = 10$ meters and the depth regulator was calculated with the weight matrices in 3.6.

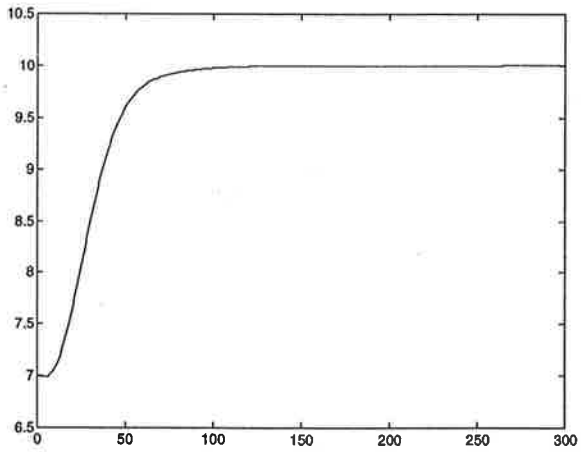

Figure 4.6 Depth change from 7 meters to 10 meters, rps = 1

As can be seen the LQ-regulator does a good job.

Gain Scheduling

Since the equations that are used to find the LQ-regulator are changing with the surge speed u, the optimal LQ-regulator has to follow the changes in u. This can be done by gain scheduling, which means that LQ parameters are calculated for a set of u-values, e.g. u=1,2..,15 and then the parameters for the current surge speed $u(t)$ are found through linear interpolation. The Simulink representation of the closed loop system with gain scheduling is shown in Figure 4.7.

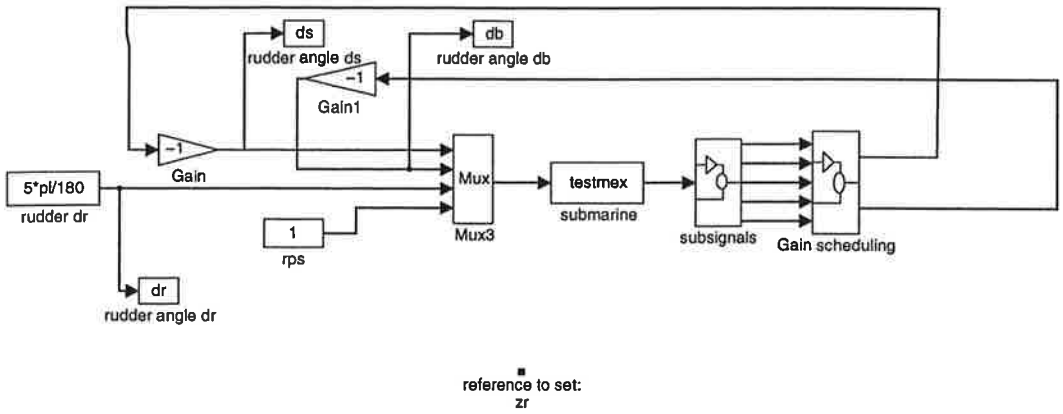

Figure 4.7 Submarine with gain scheduling. Reference to be set is zr = depth reference

The step response of the closed loop system in Figure 4.7 at the speeds rps=0.3 (1 m/s) and rps=1 (3 m/s) can be seen in Figure 4.8). The gain scheduling parameters were calculated with the Q and R in Equation 3.6.

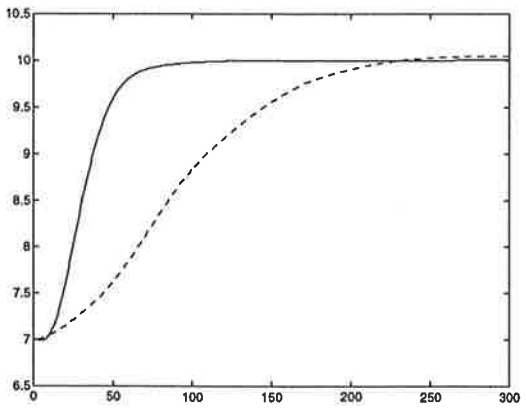

Figure 4.8 Descend to 10m, rps = 1 (solid) and rps = 0.3 (dashed)

Cross Coupling

In the open loop system a cross coupling between depth and course can be observed. In the closed loop system of Figure 4.7, with rps = 1 and a manual rudder angle δ_r of 5 degrees, the following simulation result, Figure 4.9, can be observed. See Figure 4.4. The depth set point is 7 meters.

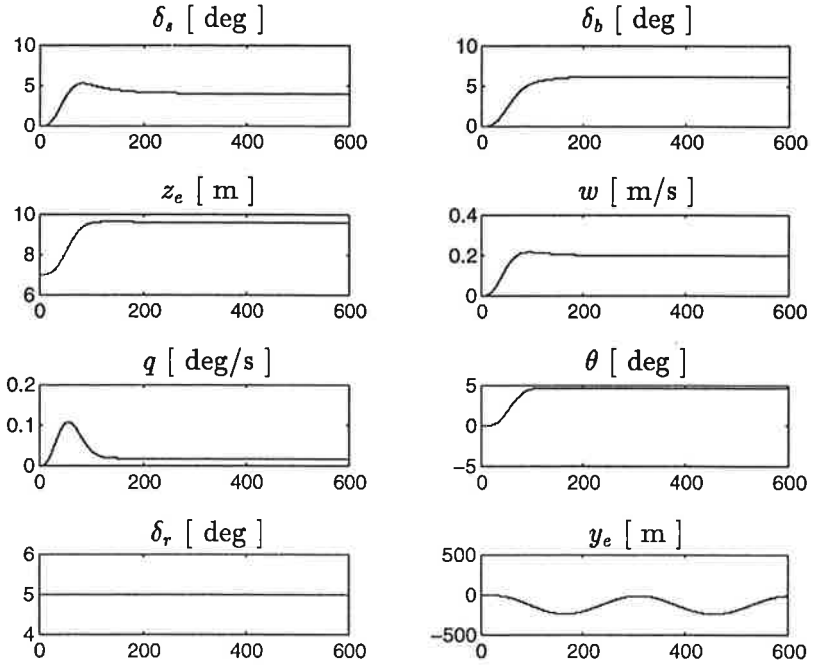

Figure 4.9 Trying to keep 7m with δ_r=5 degrees, rps=1

As can clearly be seen in Figure 4.9 its not possible to retain the depth of 7 meters. The cross coupling seen before in the open system, Figure 4.4, can also be seen here in the closed system. Notice that the effects of the cross coupling is different. In the open system the pitch angle is approximately 8 degrees and the submarine rises towards the surface. In the closed loop system the pitch angle is somewhat less than 5 degrees and the submarine sinks almost 3 meters.

The influence of changes in surge speed u and rudder angle δ_r, for the system in Figure 4.7, is shown in Figures 4.10 and 4.11. In Figure 4.10 the result of different speeds, given the same rudder angle δ_r, is shown.

As can be seen the depth error is different for different speeds. It also seems that the bigger the rudder angle δ_r the bigger the depth error. Looking at the Q matrix in Equation 3.6 it's clear that the controller not only punishes depth error, it also punishes pitch angle (θ) error. As can be seen in Figure 4.4, the turn at rps=1 introduces a pitch, and this introduces lift forces, making the submarine rise. The regulator works at lowering the pitch and the depth error. As can be seen in Figure 4.9 the pitch (θ) angle has been lowered, compared to the open system, but not in such a way that the set point depth can be maintained. Instead the submarine now sinks instead of rises. Thus the lifting forces introduced by the pitch are not enough to help the submarine maintain its depth.

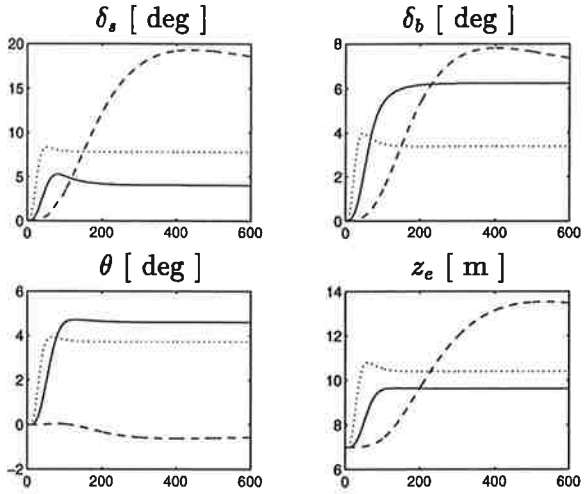

Figure 4.10 Simulations with different speeds: rps = 0.3 (dashed), rps = 1 (solid), rps = 2 (dotted). δ_r = 5 degrees

Figure 4.11 shows the result of varying δ_r angles, for a fixed initial speed $u(0) = 3$ m/s (rps = 1).

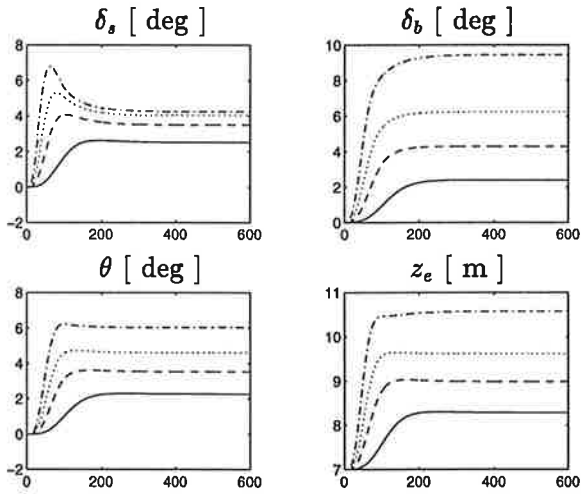

Figure 4.11 Simulations with different δ_r: δr=1 degrees(solid), δr=3 degrees(dashed), δr=5 degrees(dotted), δr=8 degrees(dash-dot). rps=1.

Analysis Looking at the simulations done on the open system, Figure 4.4, it can be seen that a course change introduces changes of depth and pitch angle (θ). Looking into the nonlinear equations that were used when deriving the linear submarine model (Normal Force- and Pitching Moment equation, Appendix A), it can be seen that three terms are affected due to the speed in the y-direction (v) and the rotation around the z-axis (r). These are $Z'_{vv}vv$, $Z'_{vr}vr$ and $M'_{vr}vr$. If assuming that the nonlinear system is decoupled, it is possible to view v and r as inputs to the depth system, i.e. disturbances, see Figure 4.12

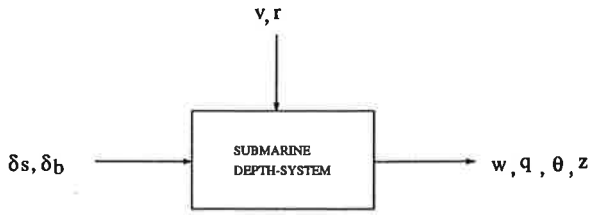

Figure 4.12 Nonlinear decoupled depth system, with disturbance signals v and r

These disturbances introduce something that can be viewed as a a weight disturbance, and this weight disturbance makes the submarine move vertically, and as the introduced weight is not situated in the CG of the submarine, but rather somewhere along the x-axis of the submarine, a pitch angle θ is also introduced. This can be shown in the following way:

The weight introduced in the submarine can be seen as a force working in the z-direction and if this force is a weight, then the force = mg is equal to components represented by Z_{vr} and Z_{vv} in the force equation F_z in Appendix A, thus:

$$mg = \frac{\rho}{2}(L^3 Z'_{vr} vr + L^2 Z'_{vv} vv) \tag{4.1}$$

Introducing the moment = mgl and equating with the moment represented by M_{vr} in the moment equation. M_{pitch} in Appendix A gives:

$$mgl = \frac{\rho}{2}(L^4 M'_{vr} vr) \tag{4.2}$$

Thus it is possible to find m and l, given v and r. An example will be shown to clarify the reasoning:
To find v and r, the open loop nonlinear system is simulated with a rudder angle $\delta_r = 5$ degrees. This simulation shows

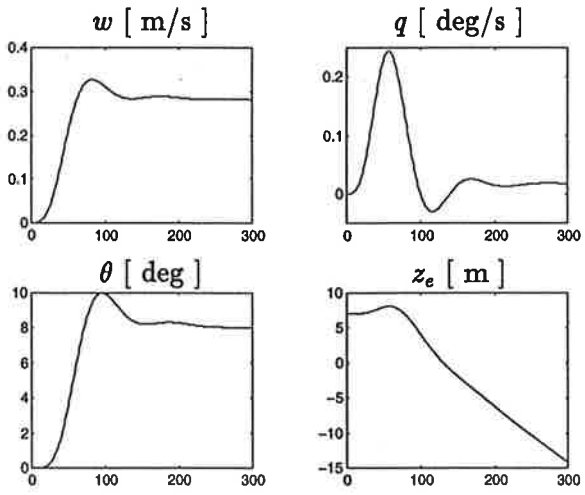

Figure 4.13 Open nonlinear system with $\delta r = 5$, rps=1

The stationary v and r are then substituted into the equations in 4.1 and 4.2 which gives m and l. The open system, with mg added to the F_z equation and mgl added to the M_{pitch} equation and no rudder turned ($\delta_r = 0$) is then simulated. The result is shown in Figure 4.14

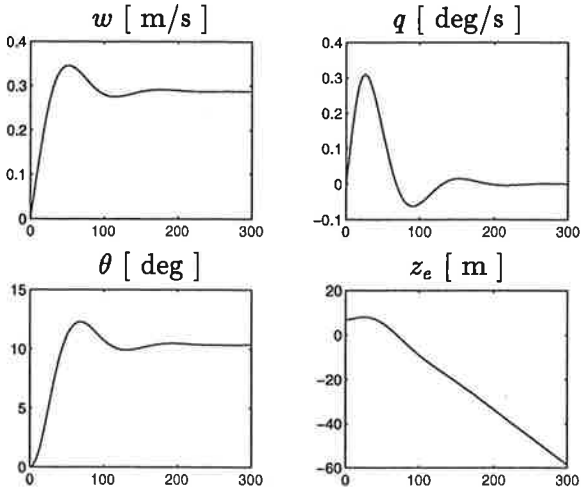

Figure 4.14 Weight added, no rudders and rps=1

As can be seen in Figure 4.14, the introduced pitch angle is too big to be equivalent with the one introduced by the turn,and the ascent of the submarine is too fast. But the test shows that turning the submarine affects the depth and pitch of the submarine in the same way as adding a weight to the submarine. This weight disturbance can be countered by changing the pitch angle, using lift or down forces on the submarine to retain its depth. (Tests made on a real submarine, support this view.)
If investigating the closed loop nonlinear system the same way, i.e. finding stationary values by turning the closed system and then adding the weight disturbance calculated from these values and setting $\delta_r = 0$, the result will be as in Figure 4.15

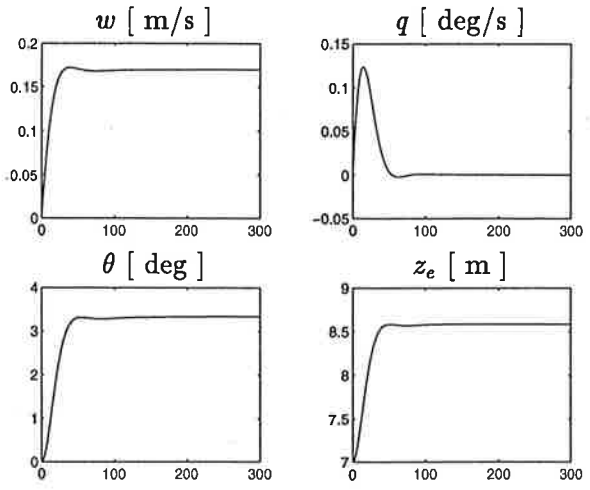

Figure 4.15 Closed loop system with weight added,$\delta r = 0$ and rps=1

These results can be compared with the ones found in Figure 4.9. It can be observed that

the general behavior of the two systems are very much alike, although the weight introduced does not make the submarine dive as deep or pitch as much as the turn does.

SUMMARY: The depth controller, computed from the linearized depth model, does what is expected when depth control is desired. But a unwished, stable, depth offset occurs if a rudder angle, δr, is applied. The offset is speed and rudder angle dependent.
Finally it was proposed and shown that turning the submarine can be seen as introducing a weight into the submarine, making the depth regulator unable to maintain the required depth.

5. Nonlinear Control Design

5.1 Introduction

This chapter presents control strategies aiming at elimination of the depth error during course change. As has been shown in chapter four, coupling effects from course to depth act as a load disturbance in the submarine. One way to counter the force introduced by this weight is to control the pitch angle.
Two ideas for pitch angle control are presented and investigated. The first is to combine reference generation for the states associated with the LQ-controller (among them the pitch angle) with nominal rudder references. The second idea is to find a pitch angle that balances the lift force introduced by the turn, by letting the depth error control a pitch angle reference.
The chapter is divided into four sections. In the first one a few facts affecting the simulations are presented. Following that is a short section where the control strategies are discussed. It should be seen as an introduction to the two last sections, Turn Acquired Reference Control and Dynamic Pitch Control. Here the two suggested pitch reference methods mentioned above are presented and tested. Each section ends with an evaluation.

5.2 Simulation Facts

- **Rudders** In the David Taylor equations there are three rudders (δ_s,δ_r,δ_b). Two in the back (sternplane (δ_s) and rudder (δ_r)) and one in the front (bowplane (δ_b)). The two rudder signals (δ_s,δ_r) for the back rudders are split up into four signals in the real submarine. This is because there are four aft rudders, configured as in Figure 5.1

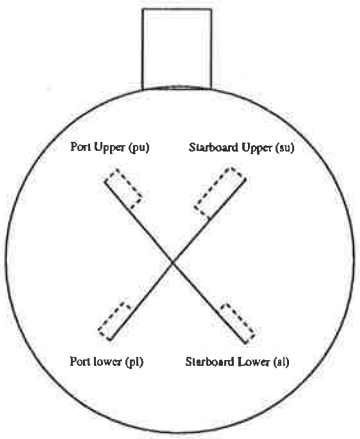

Figure 5.1 Submarine viewed from behind

This construction is added to the simulation model in order to better understand the problem of saturating rudders. Saturation may occur since the rudders have a limited

range of action, set to = ± 30 degrees.

- **Surge Speed (u)** Sternplane reversal occurs if the speed of the submarine sinks below a certain level. Therefore only rudder angles δ_r around 5 degrees (in order to achieve a small speed reduction during the turn) are investigated initially, with rps = 1.
- **LQ-controller** To optimize the performance of the LQ controller, it is possible to introduce Q and R matrices where the elements are dependent on the speed of the submarine. The effect of this is so small however that the controller used in this chapter is the same as in the previous chapters, i.e. the weight matrices are:

$$Q = \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 300 & 0 \\ 0 & 300 \end{pmatrix} \tag{5.1}$$

- **Settling time** The elimination of the depth error should be finished before or at the same time as the turn stops. The time for a turn can be approximated as proportional to the number of degrees that is turned.

5.3 Control Strategy

In chapter four, Figure 4.9, it could be observed that when a rudder angle δ_r was applied not only did the submarine turn but it increased its pitch angle and changed its depth. The depth regulator, on the form $\delta = l_r z_r$ - Lx, used in the earlier chapters view these two deviations as errors, and is incapable of correcting them.
To counter this, three possible solution may be found:

- Control the submarine in such a way that no pitch is introduced by the turn.
- View the pitch introduced by the turn as something natural, something that should be maintained. To help the LQ controller to achieve this, reference values for states and rudders are introduced.
- Use the concept of turn-introduced weight disturbance (analysis chapter 4, page 35). This weight disturbance can be balanced by forcing the submarine to have a certain pitch angle.

Of these three only the last two have been investigated. The first method is called **Turn Acquired Reference Control** and the other method is called **Dynamic Pitch Control**. The two methods are described and evaluated in sections 5.4 and 5.5.

5.4 Turn Acquired Reference Control

The section consists of three parts. The first one (A) treats constant state and rudder references. The second one (B) treats time varying references. An evaluation ends the section.

Pitch reference: θ_r

Reference trajectories for θ_r were found using

1. The closed nonlinear model, Figure 4.7. This system is called the uncompensated system, since it represents the original system with the nonlinear submarine model and the linear depth controller.
2. Filters.

Heave reference: w_r

Introducing the pitch reference θ_r means that a w_r is needed. The reason for this is that $\dot{z}$=w - $u\theta$ (equation 3.4) and no depth change is wanted, i.e. $\dot{z} = 0$. This means that the reference for w should be chosen as $w_r = u\,\theta_r$.

Methods

Five ways of using state reference values θ_r, w_r, q_r and rudder reference values δ_s^r, δ_b^r are investigated:

1. *Independent feedback compensation* Using only constant state references θ_r, w_r and q_r.
2. *Linear feedforward compensation* Using constant state references θ_r, w_r, q_r and constant nominal rudder references δ_s^r, δ_b^r. The rudder references are found by using the linear model in chapter 3.
3. *Nonlinear feedforward compensation* Expanding point 2, by introducing cross coupling effects in the rudder reference calculations. This is done by introducing $Z'_{vr}vr$, $Z'_{vv}vv$ and $M'_{vr}vr$ in the linear model used to calculate the rudder references.
4. *Independent time varying feedback compensation* Using time variant state references, i.e. $\theta_r(t)$, $w_r(t)$ and $q_r(t)$.
5. *Nonlinear timevarying feedforward compensation* Using time variant state references, i.e. $\theta_r(t)$, $w_r(t)$, $q_r(t)$ and timevarying rudder references calculated as in method 3, $\delta_s^r(t)$, $\delta_b^r(t)$.

A. Constant References

First only state references are used. Then rudder references are added, without and with additions of dynamics from the nonlinear submarine model.

Independent feedback compensation The constant reference values w_r, q_r and θ_r were found by simulating the uncompensated system, i.e. the nonlinear submarine model with the linear depth controller. The simulation time was chosen long enough for the uncompensated system to stabilize and then its stable values w(equilibrium), q(equilibrium) and θ(equilibrium) were used as references.
The system thus consists of the uncompensated system with added references, which gives the following way of calculating the rudder angles $\delta_s(t)$ and $\delta_b(t)$:

$$\delta(t) = [\delta_s(t)\,\delta_b(t)]^T = -(l_w(w(t) - w_r) + l_q(q(t) - q_r) + l_\theta(\theta(t) - \theta_r) + l_z(z(t) - z_r))$$

where $L_{2\times 4} = [l_w\ l_q\ l_\theta\ l_z]$ are the feedback gains, calculated from LQ optimization and gain scheduling with respect to the surge speed u.
An initial example where rps = 1, δ_r= 5 degrees and the initial depth, which is also the depth reference z_r, is 7 meters, is simulated and the result can be seen in Figure 5.2.

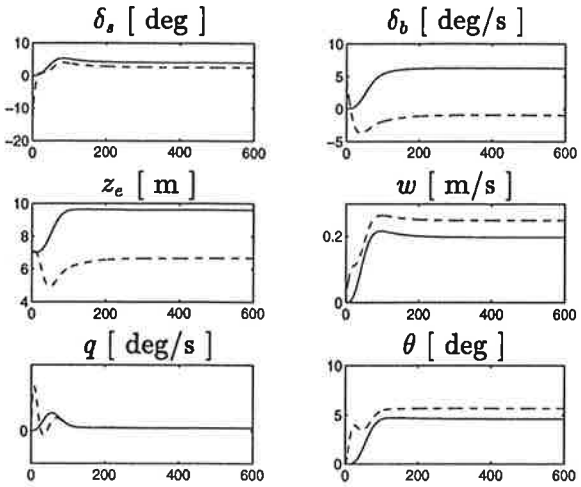

Figure 5.2 Submarine performance when applying a rudder angle (δ_r) of 5 degrees. Uncompensated (solid) and with constant references (dashed). rps = 1

As Figure 5.2 shows, the performance of the submarine system is improved by adding the constant state references. Not only is the depth error smaller for this system than for the uncompensated system, the rudder angles needed to control the submarine are also smaller. The result can be summarized in a table as:

	Uncompensated submarine	*Constant references* w_r, q_r, θ_r	$q_r = 0$
Maximum depth deviation (m)	2.64	2.1	1.98
Static depth error after 500 sek (m)	2.61	0.35	0.20

The last column q_r=0 is added due to factors involved in comparing different time varying references generations. See section Time Varying References, page 47.

Linear feedforward compensation To improve the performance, constant reference angles for the sternplane (δ_s) and the bowplane (δ_b) are calculated. These are found by examining the linearized Equation(3.1) and introducing:

$$x_1 = \begin{bmatrix} w \\ q \end{bmatrix} \qquad x_2 = \begin{bmatrix} \theta \\ z \end{bmatrix} \qquad \Delta = \begin{bmatrix} \delta_s \\ \delta_b \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 0 & 0 \\ -(Z_G W - Z_B B) & 0 \end{bmatrix}$$

thus permitting (3.2) to be written as

$$M\dot{x}_1 = Cx_1 u + D\Delta u^2 + G_2 x_2 \tag{5.2}$$

Matrices M, C and D can be found in (3.1). This gives the following way to find reference rudder angles.

$$\Delta = [\delta_s^r \; \delta_b^r]^T = \frac{1}{Du^2}(M\dot{x_1} - Cx_1u - G_2x_2) \tag{5.3}$$

Since the aim is to compute constant rudder references, $\dot{x_1} = 0$ and the last equation becomes:

$$\Delta = [\delta_s^r \; \delta_b^r]^T = \frac{1}{Du^2}(-Cx_1u - G_2x_2) \tag{5.4}$$

The approach is to find steady state values for $x_1 = [u \; w]^T$ and $x_2 = [\mathrm{q} \; \theta]^T$ from the uncompensated submarine, and calculate the sternplane reference $= \delta_s^r$ and the bowplane reference $= \delta_b^r$ using (5.4). Thus the new rudder angles are given by:

$$\delta(t) = [\delta_s(t) \, \delta_b(t)]^T =$$

$$-(l_w(w(t) - w_r) + l_q(q(t) - q_r) + l_\theta(\theta(t) - \theta_r) + l_z(z(t) - z_r)) + [\delta_s^r \; \delta_b^r]^T \tag{5.5}$$

With $\delta_r = 5$ degrees, rps = 1 and zr = 7 m, this $\delta(t)$ gives the system the following performance (Figure 5.3):

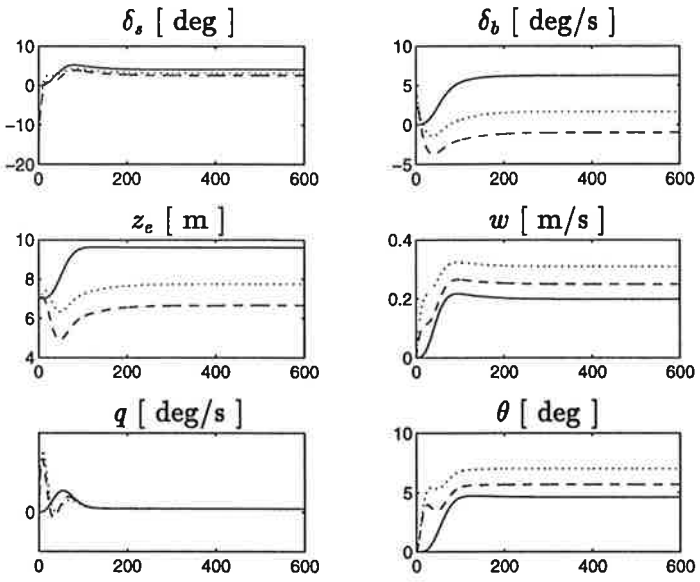

Figure 5.3 Submarine performance for three different cases: Uncompensated (solid), state references (dashed), state and rudder references (dotted)

	Uncompensated submarine	*Constant references* w_r, q_r, θ_r, δ_s^r, δ_b^r	$q_r = 0$
Maximum depth deviation (m)	2.64	0.74	0.74
Static depth error after 500 sek (m)	2.61	0.74	0.74

Compared to the simulation using only state references, this reference strategy improves the maximum depth error. It does however not improve the static depth error. Nor does it lessen the work done by the rudders δ_s and δ_b.

Nonlinear feedforward compensation The linearized submarine model (equation 3.1) was used to calculate the constant rudder reference values used in the previous simulation. If adding dynamic due to the turn, see analysis page 35, the following can be added:

$$DR = \begin{pmatrix} F_z \\ M_{pitch} \end{pmatrix} = \begin{pmatrix} \frac{\rho}{2}(L^3 Z'_{vr} v r + L^2 Z'_{vv} v^2) \\ \frac{\rho}{2}(L^4 M'_{vr} v r) \end{pmatrix}$$

To make the calculations of $\Delta = [\ \delta_s^r\ \delta_b^r\ ]$, according to (5.3) more accurate, i.e. taking heed to sway and angular velocity around the z-axis, DR is added to equation 5.2 as:

$$M\dot{x}_1 = Cx_1u + D\Delta u^2 + G_2x_2 + DR \tag{5.6}$$

which can then be written:

$$\Delta = [\delta_s^r\ \delta_b^r]^T = \frac{1}{Du^2}(-Cx_1u - G_2x_2 - DR) \tag{5.7}$$

With this addition, the rudder angles entering the submarine are still given by (5.5) and the result of a given rudder angle $\delta_r = 5$ degrees is shown in Figure 5.4.

	Uncompensated submarine	*Constant references* w_r, q_r, θ_r, δ_s^r, δ_b^r	$q_r = 0$
Maximum depth deviation (m)	2.64	2.77	2.67
Static depth error after 500 sek (m)	2.61	0.03	0.19

The introduction of DR thus makes the static depth error better. The transient behavior has deteriorated though and is now even worse than for the uncompensated submarine. Gained by the introduction of reference rudder angles are better rudder angles δ_s and δ_b, i.e they are smaller.

Conclusions By using static reference values, the behavior of the submarine can be improved, i.e. an asymptotically much better depth keeping can be found if using θ_r, w_r

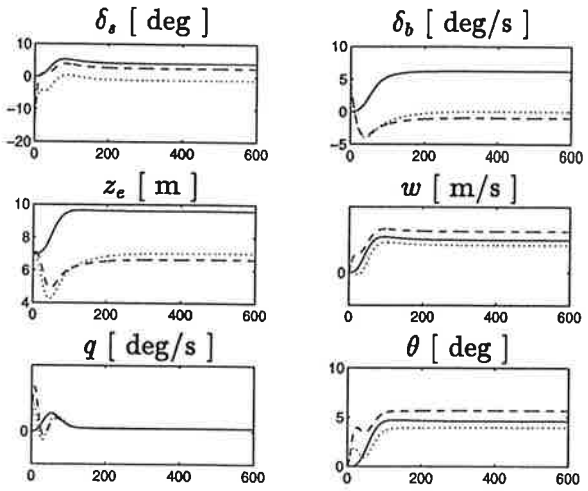

Figure 5.4 Submarine performance when adding a rudder angle (δ_r) of 5 degrees. Uncompensated (solid), state references (dashed) and with state and non-linear rudder references (dotted). rps = 1

and q_r. Adding rudder references (δ_s^r and δ_b^r) improves the static depth error control even more.

B. Time Varying Reference Generation

By addressing the problem dynamically, i.e. calculating time varying reference values it is possible to improve the transient behavior compared to the static approach. In effect this approach produces a trajectory reference for the submarine to follow, letting the linear depth controller adjust state deviations around this trajectory.
It is possible to generate references by use of the uncompensated system. Two possibilities exist, depending on whether state references suffice or rudder references have to be added. If it is enough to use q_r(t),w_r(t) and θ_r(t) (state references only) the uncompensated system can be replaced by a filter that produces θ_r(t) and $\dot{\theta}_r$(t) since that will give:

- θ_r(t)
- q_r(t)=$\dot{\theta}_r$(t)
- w_r(t)=u(t)θ_r(t) [5]

This will make it possible to have a filter bank, not having to simulate the whole model in real time. This method is called *Independent timevarying feedback compensation*
If rudder references are needed, both the filter and the uncompensated system have to be used. The reason for this is that the rudder equation (from Equation 5.7)

$$\Delta(t) = [\delta_s^r(t)\ \delta_b^r(t)]^T = \frac{1}{D\,u^2(t)}(M\,\dot{x}_1(t) - C\,x_1(t)\,u(t) - G_2 x_2(t) - DR(t)) \qquad (5.8)$$

needs the following additional values to be solved:

- $\dot{q}_r$(t)=$\ddot{\theta}$(t)

[5] From equation 3.4, setting $\dot{z}$=0

- $\dot{w}_r(t) \approx u(t)\dot{\theta}_r(t)$
- $v_r(t)$
- $r_r(t)$

where the first two will come from the filter and the two others from the uncompensated system. This method is called *Nonlinear timevarying feedforward compensation.*
This way of producing reference values may be visualized the following way[1]:

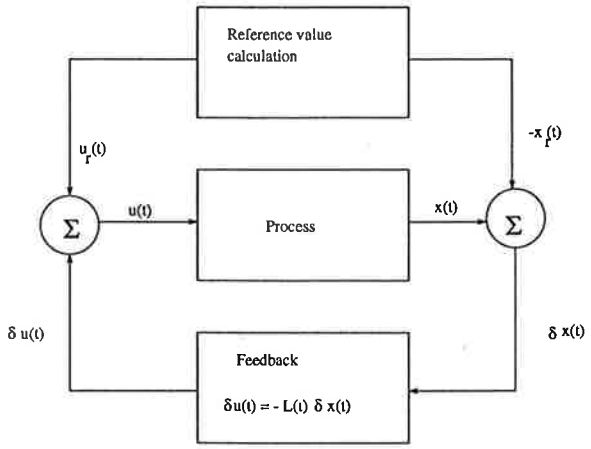

Figure 5.5 Generation of state and control references

Independent Timevarying Feedback Compensation The system at hand looks as follows in Simulink:

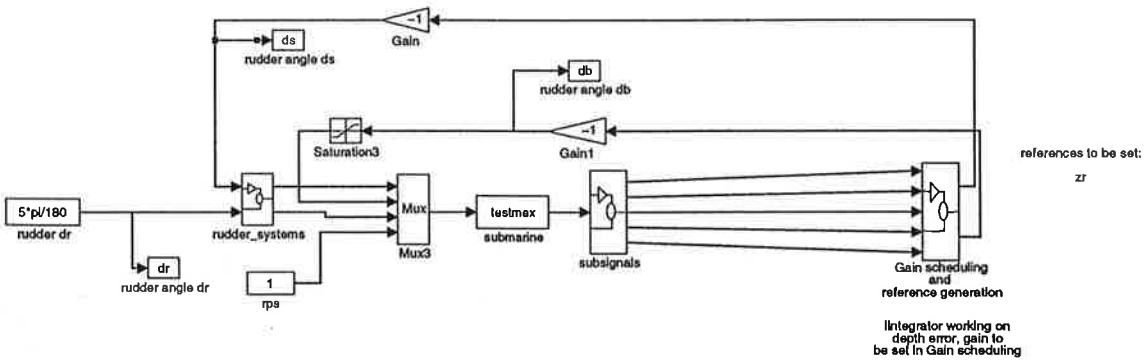

Figure 5.6 Simulink representation with dynamic references generated by second order filter

A second order filter with a step input is incorporated in the Gain Scheduling block, shown in Figure 5.6. The filter is shown in Figure 5.7. The output of the filter is matched to the pitch angle response of the uncompensated system. The filter thus produces $\theta_r(t)$, $q_r(t)$ and $w_r(t)$.

The rudder signals, $\delta_s(t)$ and $\delta_b(t)$, are found through:

$$\delta(t) = [\delta_s(t)\,\delta_b(t)]^T = -(l_w(w(t) - w_r(t)) + l_q(q(t) - q_r(t)) + l_\theta(\theta(t) - \theta_r(t)) + l_z(z(t) - z_r(t)))$$

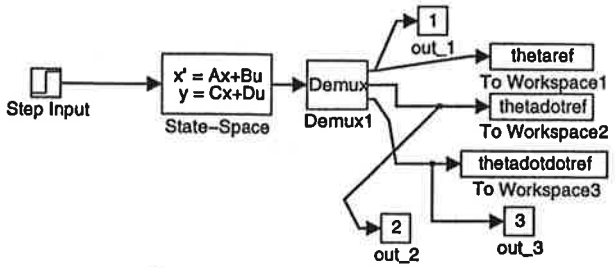

Figure 5.7 reference generation

Simulating this system with a rudder angle $\delta_r = 5$ degrees, rps $= 1$, gives the following result, (Figure 5.8).

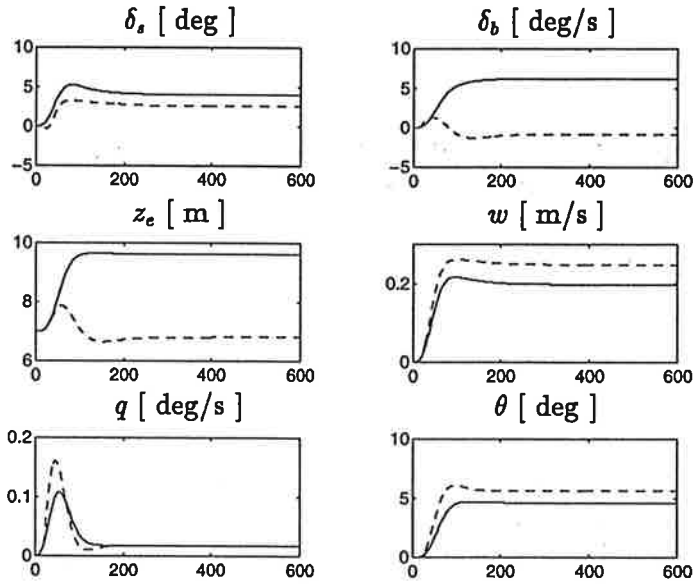

Figure 5.8 Uncompensated (solid) and with filter references (dashed)

	Uncompensated submarine	*Filter references* $w_r(t)$, $q_r(t)$, $\theta_r(t)$	$q_r(t) = 0$
Maximum depth deviation (m)	2.64	0.87	0.71
Static depth error after 500 sek (m)	2.61	0.19	0.11

The overall result is better now than when using only constant references, since a low maximum depth error is combined with a low static depth error.
Added in the list above is a test where q_r is set to zero. This is due to the fact that if the uncompensated system is used to produce references, its $q_r(t)$ will not diminish to zero once a stable $\theta(t)$ is reached and that will make the filter and uncompensated system references deviate. Thus, to be able to compare simulations with references from either a filter or an uncompensated system, the column $q_r(t) = 0$ is added.

Nonlinear Timevarying Feedforward Compensation Adding rudder references to the system called Independent timevarying feedback compensation, the rudder signals $\delta_s(t)$ and $\delta_b(t)$ are given by:

$$\delta(t) = [\delta_s(t)\,\delta_b(t)]^T =$$

$$-(l_w(w(t)-w_r(t))+l_q(q(t)-q_r(t))+l_\theta(\theta(t)-\theta_r(t))+l_z(z(t)-z_r(t)))+[\delta_s^r(t)\,\delta_b^r(t)]^T \quad (5.9)$$

with

$$\Delta(t) = [\delta_s^r(t)\ \delta_b^r(t)]^T = \frac{1}{D\,u^2(t)}(M\,\dot{x}_1(t) - C\,x_1(t)\,u(t) - G_2x_2(t) - DR(t)) \qquad (5.10)$$

The signal $r(t)$ and $v(t)$ are produced by the uncompensated model. Simulating this gives the result shown in Figure 5.9

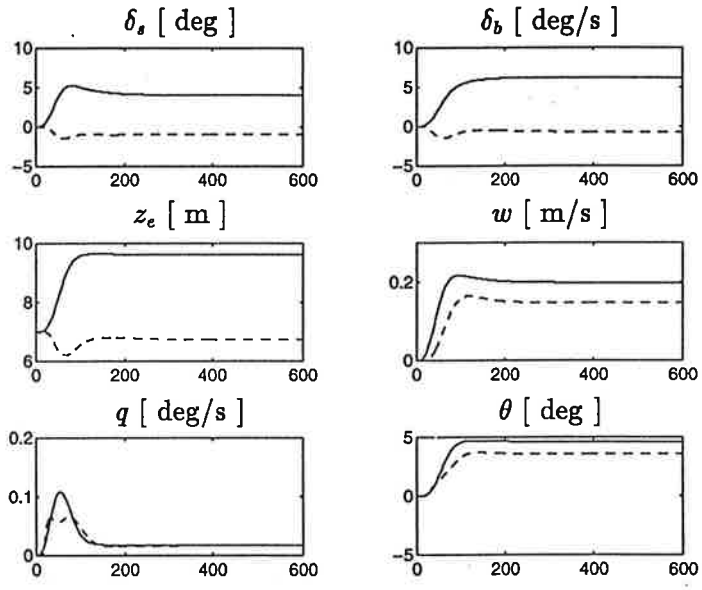

Figure 5.9 Uncompensated (solid) and compensated (dashed)

	Uncompensated submarine	*Filter ref:* $w_r(t)$, $q_r(t)$, $\theta_r(t)$ *Uncomp.sub. ref:* $v_r(t)$, $r_r(t)$	$q_r(t) = 0$
Maximum depth deviation (m)	2.64	0.79	0.42
Static depth error after 500 sek (m)	2.61	0.27	0.27

As can be seen in the table above, the transient performance is improved, having added rudder references. Comparing Figures 5.8 and 5.9 it can also be found that the rudder movements are less in the system with rudder references and that is good since it gives the controller flexibility to perform other tasks if needed, and it also means that the pitch angle introduced is close to being 'natural' thus not needing big forceful rudder actions to be executed.

Introduced in this calculation are $\dot{w}(t)$ and $\dot{q}(t)$, through matrix M in equation 5.10. If they are excluded in the calculations the following is the result:

	Uncompensated submarine	*Filter references* $w_r(t)$, $q_r(t)$, $\theta_r(t)$	$q_r = 0$
Maximum depth deviation (m)	2.64	0.73	0.29
Static depth error after 500 sek (m)	2.61	0.27	0.27

This seems to improve the performance of the submarine. The conclusion is that the derivatives are unnecessary. If one does not want to make a filter bank, this means that it is possible to use the uncompensated submarine model as a reference generator, since $w_r(t)$, $q_r(t)$ and $\theta_r(t)$ are the only references needed.
This is not true however, as a simulation using only the uncompensated system as reference generator, generating $w_r(t)$, $q_r(t)$ and $\theta_r(t)$ show in Figure 5.10

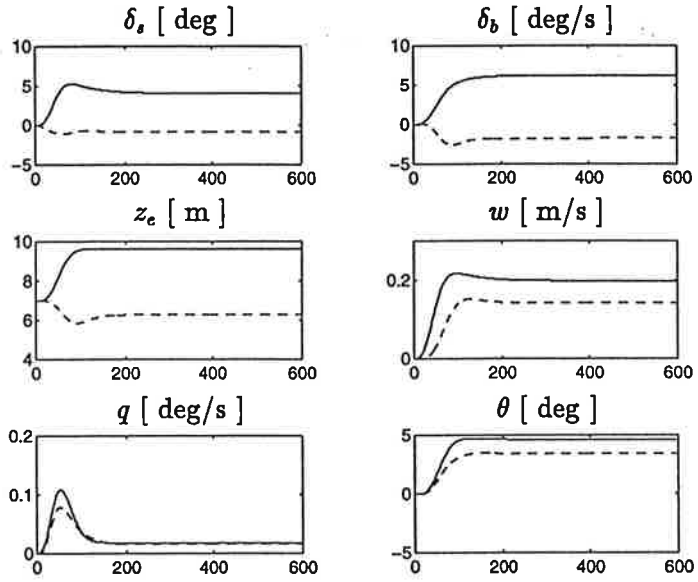

Figure 5.10 Uncompensated (solid) and with references (dashed)

	Uncompensated submarine	*Uncomp.sub.ref* $w_r(t)$, $q_r(t)$, $\theta_r(t)$	$q_r = 0$
Maximum depth deviation (m)	2.64	1.13	0.73
Static depth error after 500 sek (m)	2.61	0.72	0.27

This performance is not as good as when both a filter and an uncompensated submarine model are used to produce reference values.

Evaluation

The tests done in this first part, i.e. Turn Acquired Reference Control can be summarized in the following way. The page reference indicates where a more thorough presentation of the simulations can be found. The values in brackets are obtained when simulating with $q_r = 0$;

SYSTEM DESCRIPTION	Page reference	Maximum depth deviation (m)	Static depth error (m)
1. Uncompensated submarine	34	2.64	2.61
2. Independent feedback compensation	41	2.1 (1.98)	0.35 (0.20)
3. Linear feedforward compensation	42	0.74 (0.74)	0.74 (0.74)
4. Nonlinear feedforward compensation	44	2.77 (2.67)	0.03 (0.19)
5. Independent timevarying feedback compensation	46	0.87 (0.71)	0.19 0.11)
6. Nonlinear timevarying feedforward compensation	48	0.73 (0.29)	0.27 (0.27)

As can be seen two systems can be selected as being best in their respective group. In the constant reference group (nr 1 - 4), the *Independent feedback compensation with* $q_r = 0$, nr 2, is the best. In the time varying group (nr 5 - 6), the *Nonlinear timevarying feedforward compensation*, nr. 6, is the best. These two methods, nr 2 and 6, are presented below, where the depth of the submarine is investigated for different manual course angles and at different speeds.

Independent feedback compensation First presented on page 41. Note that $q_r = 0$ in the following simulations.

rps = 0.3

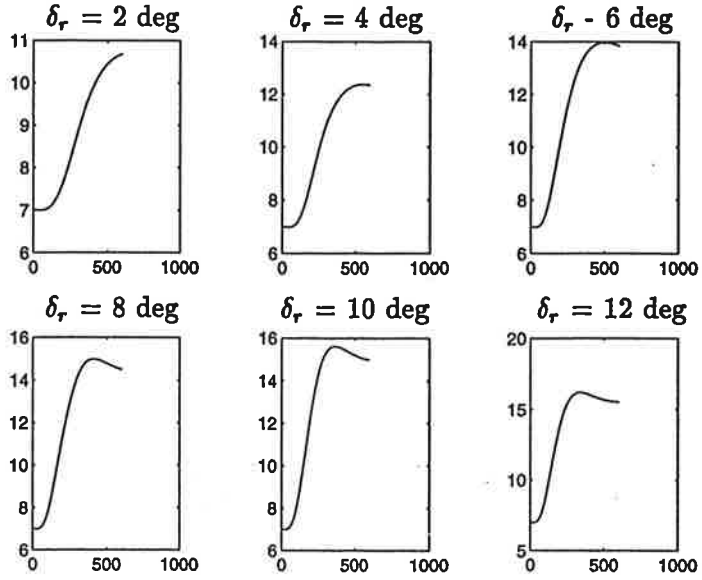

Figure 5.11 Independent feedback compensation system given different δ_r at rps = 0.3. The figure shows submarine depth as a function of time.

rps = 1

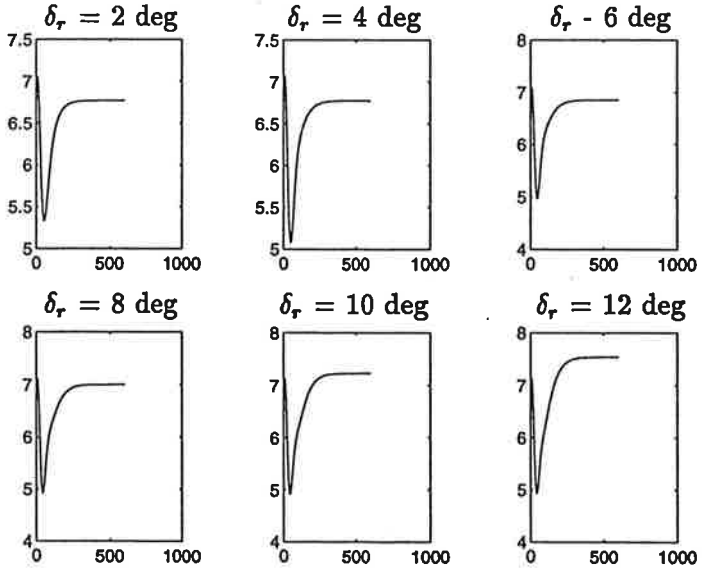

Figure 5.12 Independent feedback compensation system given different δ_r at rps = 1

rps = 2

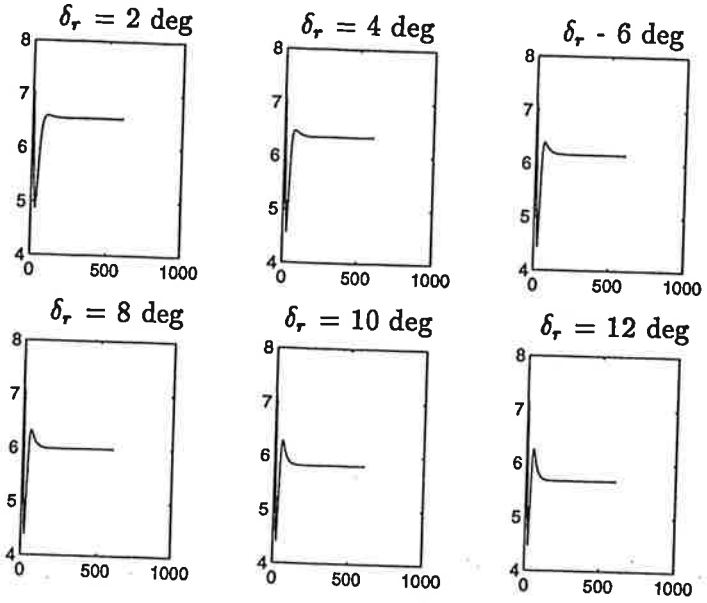

Figure 5.13 Independent feedback compensation system given different δ_r at rps = 2

As can be seen in Figures 5.11, 5.12 and 5.13, this is not a method which can be utilized in the depth control of the submarine. The static depth error for speeds around rps=0.3 is unacceptable and for speeds above rps = 1 the transient response is unsatisfactory.

Nonlinear timevarying feedforward compensation

First presented on page 49.
The second order filter is described by the transfer function

$$\frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2}$$

and transformed to the state space description

$$A = \begin{pmatrix} 0 & 1 \\ -\omega^2 & -2\zeta\omega \end{pmatrix} \qquad B = \begin{pmatrix} 0 \\ \omega^2 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -\omega^2 & -2\zeta\omega \end{pmatrix} \qquad D = \begin{pmatrix} 0 \\ 0 \\ \omega^2 \end{pmatrix}$$

which is then used in the Simulink simulation.
In this test series filters for δ_r = 2, 5 and 10 degrees were found for the speeds given by rps = 0.3, 1 and 2. Then interpolation gave the values for ω and ζ at the other speeds.

rps = 1

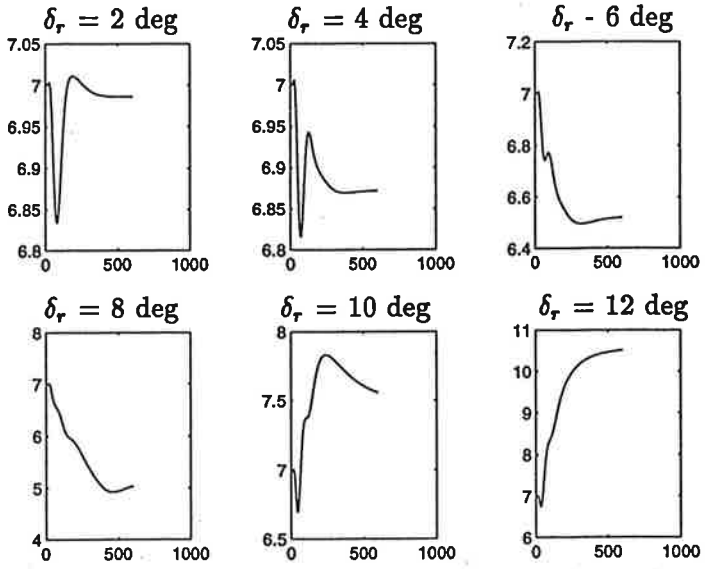

Figure 5.14 Nonlinear timevarying feedforward compensation system at rps = 1. The figure shows submarine depth as a function of time.

As can be seen the interpolation is only a rough instrument, the simulation for $\delta_r = 12$ degrees would probably show a better depth error curve if a specific filter was found for this turn.

rps = 2

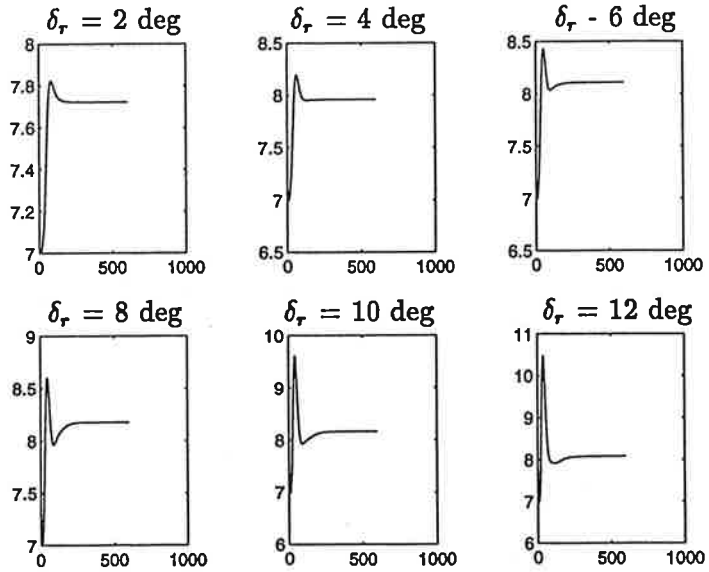

Figure 5.15 Nonlinear timevarying feedforward compensation system at rps = 2

The promising results from the initial testing, page 49, are not to be seen. If viewing the period of interest as the one between 1 - 100 seconds (since the turning rate is approximately 1 degree per second and most turns are less than 100 degrees), it is clear

that there is no control of the depth in this period. That a acceptable depth is reached after 500 seconds does not promote this method as being useful.

5.5 Dynamic Pitch Control

Introduction

In the analysis at the end of chapter four, the idea of weight change was introduced. This effect can in fact be controlled by changing the pitch angle. By changing the pitch, the lift forces affecting the submarine can be changed and thus depth control is introduced.
If the submarine sinks, the pitch angle needs to become larger, thus making the submarine ascend. One way to express this is

$$\frac{d\theta_r}{dt} = k_e(z - z_r) \tag{5.11}$$

where $k_e > 0$.
In Simulink this can be shown the following way

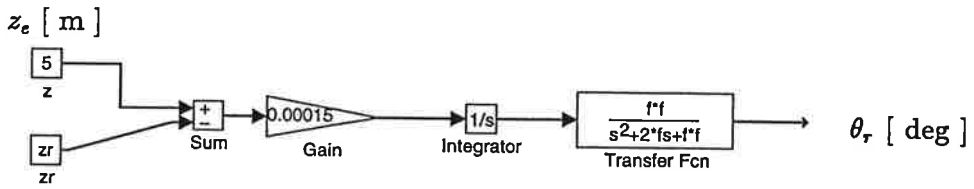

Figure 5.16 Generation of θ_r by $z_e = z$ - z_r

Adding this construction to the uncompensated submarine system, and using the fact that $w_r = u\,\theta_r$, does give a system that is capable of adjusting the pitch due to the resulting depth error that is produced when a manual course angle δ_r is given. Without presenting a method for finding the filter coefficients and the gain, a test produced the following response to a 5 degree δ_r angle with rps = 1:

	Uncompensated submarine	*Dynamic pitch control of* $\theta_r(t)$
Maximum depth deviation (m)	2.64	0.52
Static depth error after 500 sek (m)	2.61	0.05

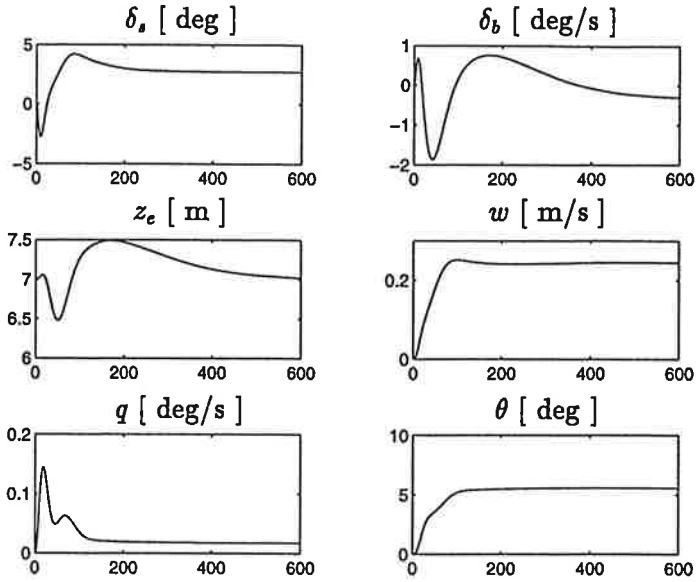

Figure 5.17 Dynamic pitch control system for a 5 degree turn, rps = 1

As can be seen in Figure 5.17 the possibility to control the depth during a turn is improved. Lacking is a method for finding coefficients for the regulator structure in Figure 5.16. One method is suggested and presented next.

PID Control

To simplify the use of a dynamic pitch reference a PID controller is used instead of the P-controller , presented in(5.11) and Figure 5.16 .
The PID parameters can be found by using Ziegler-Nichols method, i.e. setting the regulator parameters such that it is only a P-regulator and then increasing the gain until the system reaches constant oscillation. The gain that produces this oscillation is called K_c and the period of the oscillation is called T_0.
The control output u(t) can be written as:

$$u(t) = K(e(t) + \frac{1}{T_i}\int^t e(t)dt + T_D\frac{de}{dt})$$

noting that $u(t) = \frac{d\theta_r}{dt}$ and $e(t) = z(t)$ - $z_r(t)$.
The PID parameters for Simulink can be calculated as:
P = K = Kc 0.6
I = $K/T_I = K/(T_0 0.5)$
D = $KT_D = K(T_0/8)$
Applying this approach to the submarine system, i.e the uncompensated submarine with a PID added such that it generates pitch references when rps = 1 and δ_r = 5 degrees, gives Kc = 0.07 and TD = 76. This gives the PID parameters; P = 0.042, I = 0.0011 and D = 0.399 and the following result:

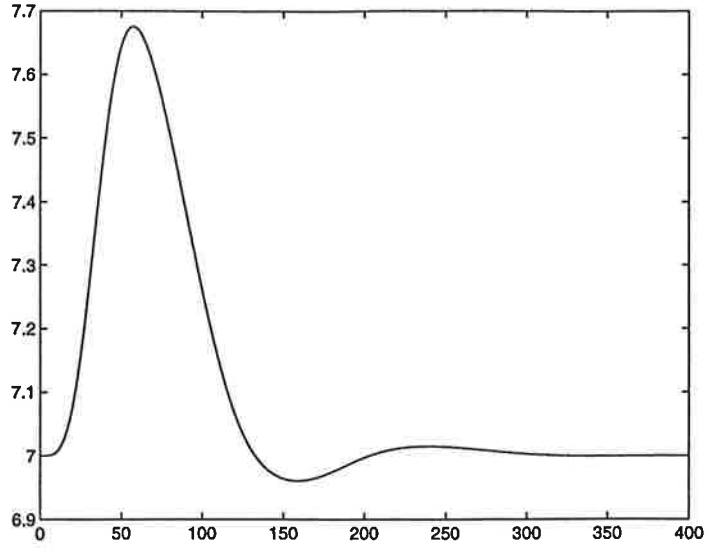

Figure 5.18 Depth of PID system at rps = 1 and $\delta_r = 5$ degrees

	Uncompensated submarine	*Pid control of* $\theta_r(t)$
Maximum depth deviation (m)	2.64	0.68
Static depth error after 500 sek (m)	2.61	0

As can be seen in Figure 5.18, the system is a little shaky, but it does handle depth control rather well. The PID parameters are not perfect since the Ziegler-Nichols method only gives a rough estimate of how to tune the regulator. To improve the damping, the following adjustment is done:
P = K = Kc 0.6
I = $K/T_I = K/(T_0/1.3)$
D = $KT_D = K(T_0/8)$
and this gives the following result:

	Uncompensated submarine	*Pid control of* $\theta_r(t)$
Maximum depth deviation (m)	2.64	0.73
Static depth error after 500 sek (m)	2.61	0

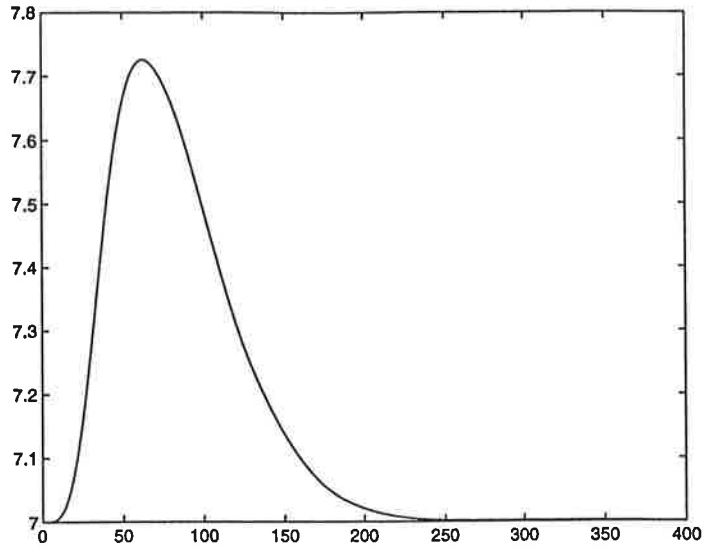

Figure 5.19 Depth of PID system at rps $= 1$ and $\delta_r = 5$ degrees

Evaluation

To investigate this method further, the following three tests show how the submarine behaves for different speeds and different rudder angles.

For rps $= 0.3$

The system does not work for this speed. This since the speed is so slow that depth control is not handled by changing the pitch angle.(This behavior was first mentioned in section 3.4, in connection with the linear model.) Thus the rudders saturate after a while, trying to control the pitch and with saturated rudders the submarine just dives.

For rps = 1 (Kc=0.07 and TD=76) the following result can be seen:

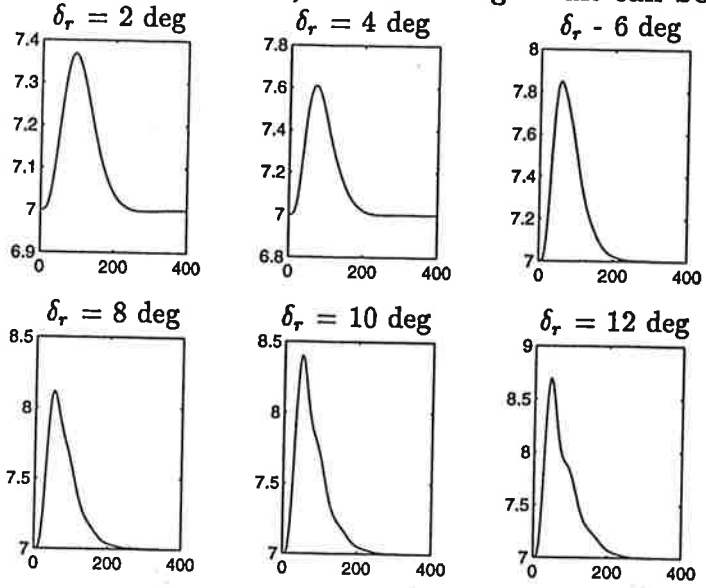

Figure 5.20 Depth of PID system turning at rps = 1

For rps = 2 (Kc=0.04 and TD=38) the following result can be seen:

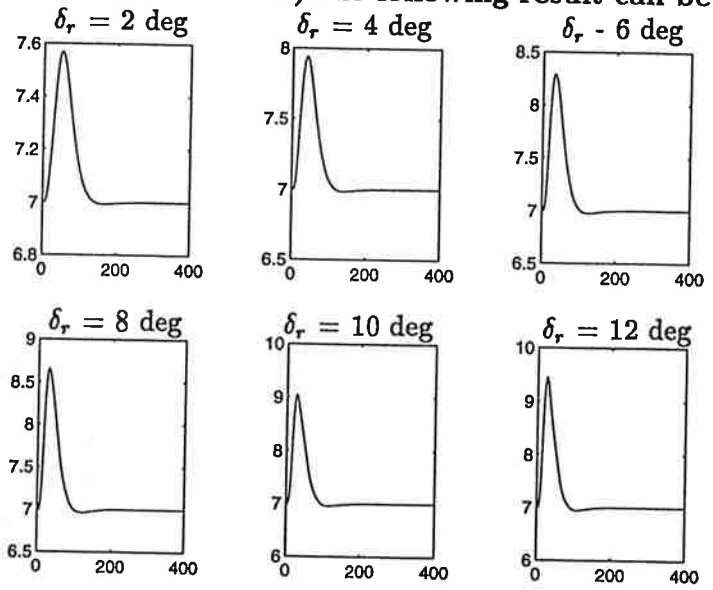

Figure 5.21 Depth of PID system at rps = 2

In the test above (rps=2) the I variable of the PID controller was reduced to $\frac{Kc}{TD}$, to avoid shaky behavior.

The simulations done in Figures 5.20 and 5.21 supports the view that this is the method among the ones tested that should be used and further developed.

A comparison with the uncompensate submarine for the speed given by rps = 1 shows that the depth error

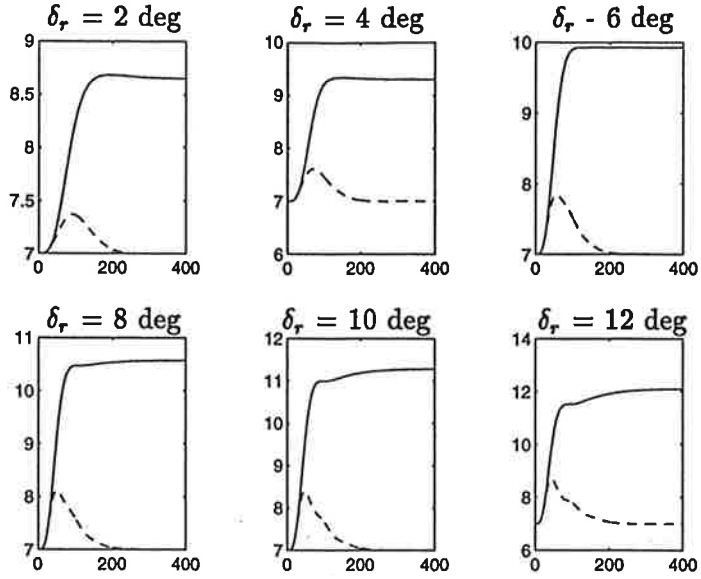

Figure 5.22 Depth at rps = 1. Uncompensated system (solid) and PID system (dashed).

is considerably lowered.
A comparison at rps = 2 shows

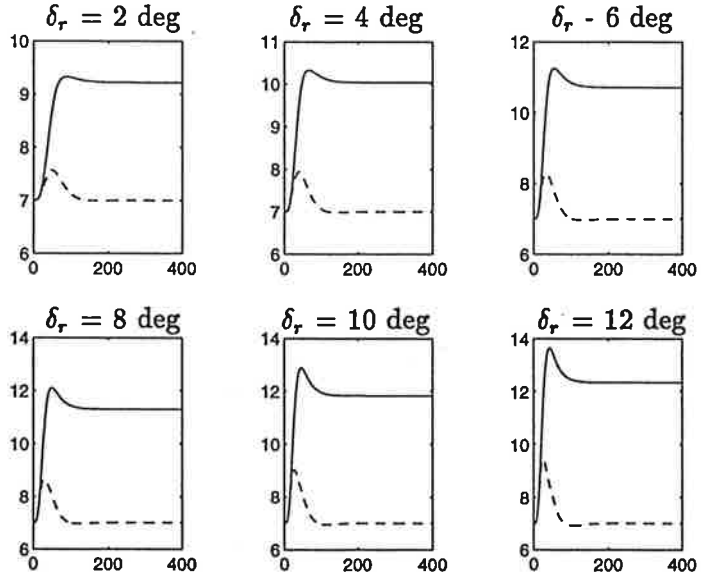

Figure 5.23 Depth at rps = 2. Uncompensated system (solid) and PID system (dashed).

that the first 20 seconds are almost the same, both in the case when rps = 1 and when rps = 2, i.e. no improvement in the depth keeping can be seen. After that the improvement is significant and after 100-200 seconds the error is in most cases eradicated by introducing the PID controller.

An interesting question that arises is if anything has been gained in terms of less rudder movements. Figures 5.24 and 5.25 show the sternplane angle δ_s for different rudder angles δ_r at the speeds given by rps = 1 and 2:

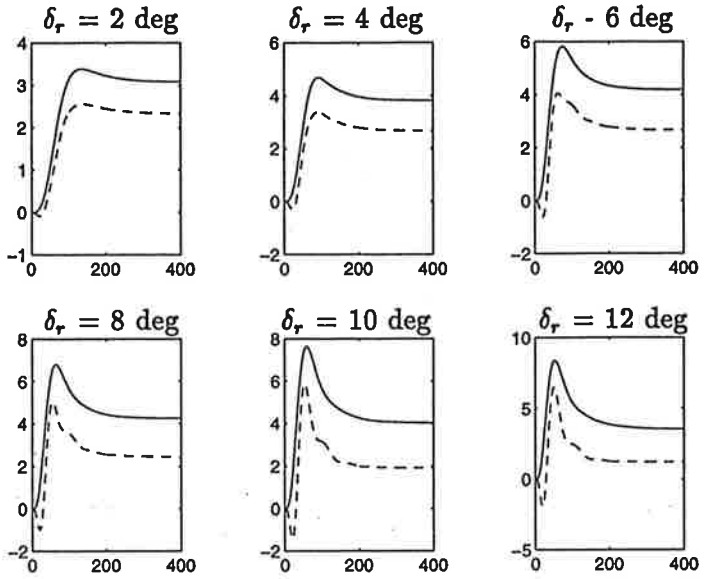

Figure 5.24 PID system at rps = 1. Rudder angles δ_s for the uncompensated system (solid) and the PID system (dashed).

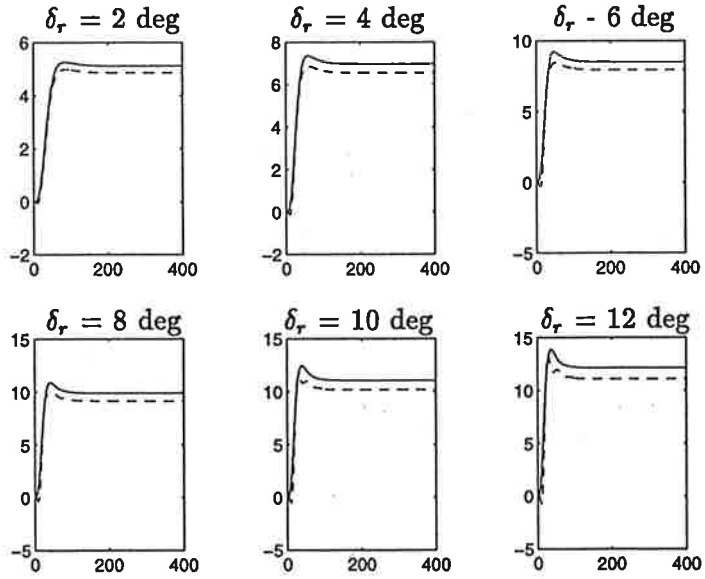

Figure 5.25 PID system at rps = 2. Rudder angles δ_s for the uncompensated system (solid) and the PID system (dashed).

It does seem as if there is only a small gain involved in using the PID scheme when looking at the aft rudder. What of the front rudder? Figures 5.26 and 5.27 shows the bowplane angle δ_b for the uncompensated system and the PID system.

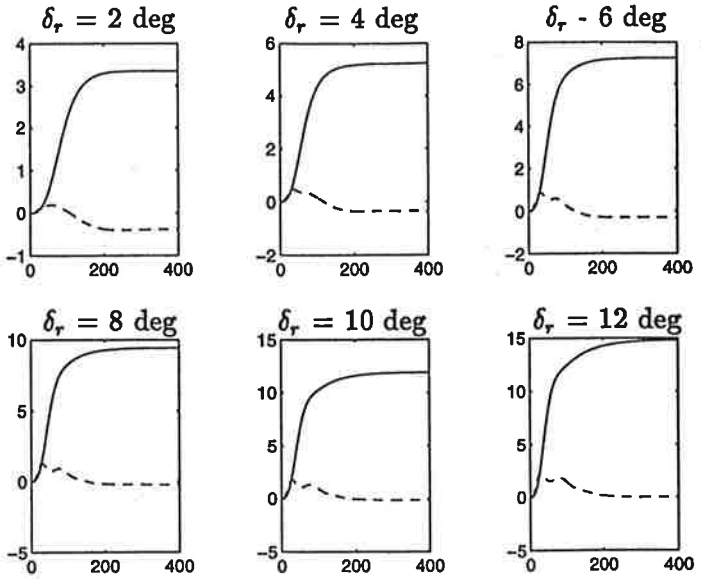

Figure 5.26 PID system at rps = 1. Rudder angles δ_b for the uncompensated system (solid) and the PID system (dashed).

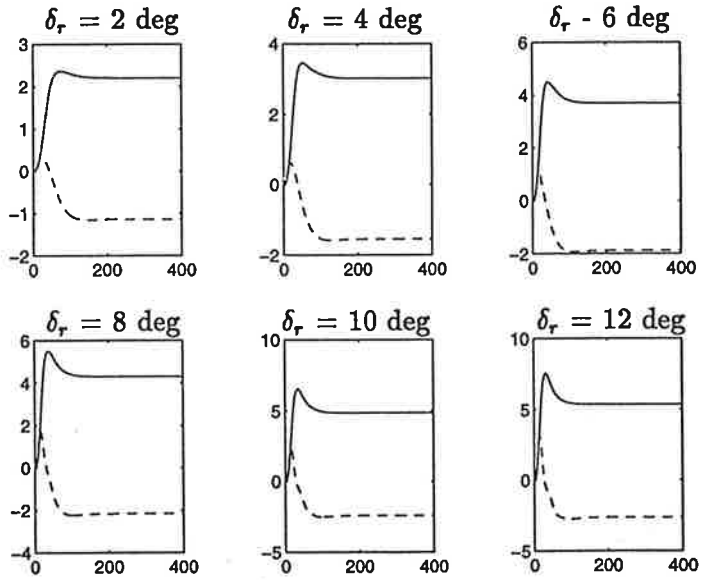

Figure 5.27 PID system at rps = 2. Rudder angles δ_b for the uncompensated system (solid) and the PID system (dashed).

Here the PID controller lessen = improve the magnitude of the control signal in a more noticeable way, which gives more room for other actions that might be necessary during the turn, as any emergency actions.
It is also a fact that the less rudder movements the better, since rudder movements means energy consumption.

Conclusions Using a PID controller to produce pitch references, together with the Ziegler-Nichols method (somewhat adjusted to increase the damping) for finding the PID parameters, made it possible to considerably lower the depth error that is the result of a

turn. The method is however not applicable to slow moving submarines, since at low speeds, it is not possible to control depth by varying the pitch angle.

6. Summary and Reflections

6.1 Overview of Master Thesis

The project was done for Kockums Submarine Systems AB in Malmö and the Department of Automatic Control at LTH.

SUBMARINE A submarine can be pictured the following way:

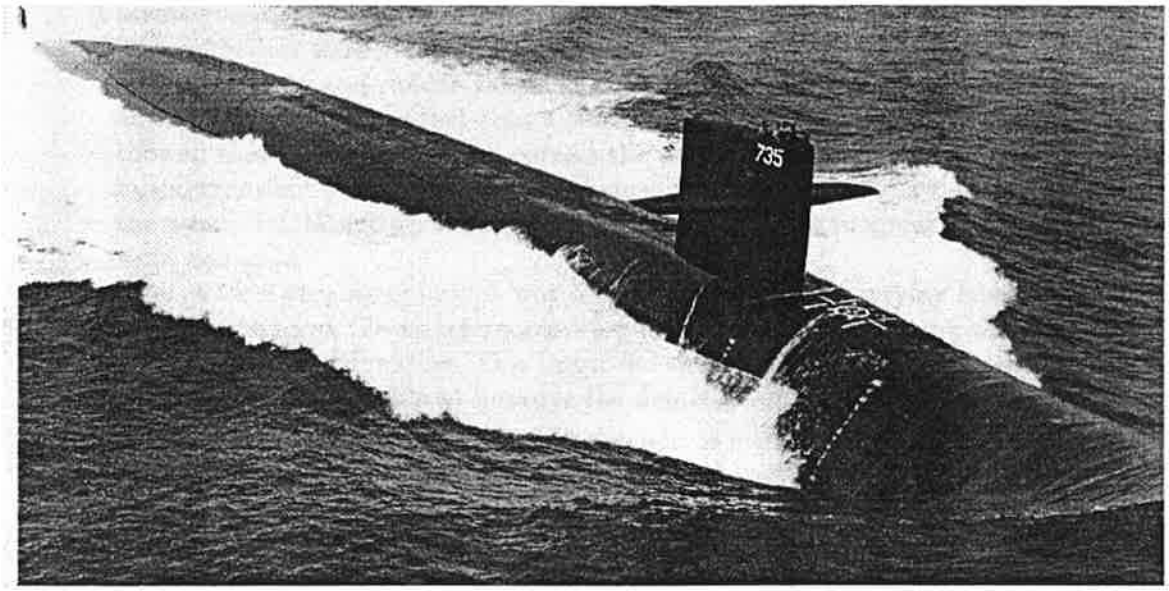

Figure 6.1 Submarine from side

with propellers and aft rudders in the upper left corner and the bowplane (δ_b) visible on the tower.
Input signals in the model of the submarine are the aft rudders, δ_s, δ_r and the bowplane δ_b and the propeller revolution rate rps.
To control the depth, an LQ-regulator is designed from a linearized model of the submarine. The LQ-regulator works with the rudders δ_s and δ_b. For control of sideway motion a PID controller is used, thus controlling the rudder δ_r. This PID is however not included in this thesis work.

PROBLEM Given a manual rudder angle δ_r, the submarine is unable to maintain its depth, although the LQ-designed depth controller performs well when there is no course change.

ASSIGNMENT Make a simulator in Matlab's Simulink such that the cross-coupling between course and depth witnessed in a real submarine can be seen and investigated, i.e. such that when a manual rudder angle δ_r is applied the submarine is unable to maintain its current depth. Then find a way to reduce the depth error. This should be done without changing the depth control system, i.e. the LQ-regulator structure.

INVESTIGATED SOLUTIONS:

1. Combine reference generation for the states associated with the LQ-controller with nominal rudder references.
2. Find a method, producing a pitch angle reference, such that the pitch angle balances the forces introduced by the turn. Do this by letting the depth error control the pitch reference.

RESULTS

1.
   - First a static solution to the cross coupling between depth and course was investigated. This was done by introducing constant state and rudder references. The references were generated by turning an uncompensated submarine model and using state and rudder values at equilibrium. This was combined with adjustments due to the fact that a zero depth change was expected. The results showed that it was possible to correct the depth error given enough time. An initial transient persisted though and since the time scale for a turn is given by the approximation: turn time (seconds) $\sim$ turned angle (degrees), the methods were unusable.
   - Then a time varying approach was tested, i.e. using time varying state and rudder references. These references were also produced by turning the uncompensated submarine. This improved the initial behavior, but none of the methods tested was able to improve the depth keeping in a satisfactory way.
2. Viewing the depth keeping problem as if a weight is introduced in the submarine when a turn is made, led to a dynamic pitch control and investigations showed that this method was the best. The method consists of a PID controlled pitch reference produced by the depth error.
   The method, further presented in section 5.6, showed that, depending on the surge speed and the course rudder angle, the transient error could be lowered to 20 - 25 % of the uncompensated system's error and that the depth error could be made to diminish altogether, in a time less than 200 seconds.

6.2 Concluding remark

The implementation of the submarine simulator in Simulink led to a working simulator, which is now used at Kockums. The results of the investigations into the cross-coupling between course and depth has led to a control scheme that reduces the depth error. Improvements can be done by refining the calculations of the PID parameters and introducing greater flexibility in the choice of the R and Q matrices in the LQ-regulator (making them functions of the surge speed).
To improve things further, it might be possible to start changing the pitch angle even before the turn is started. Investigations into the equivalent mass and a refinment of the calculations as to its whereabout in the submarine may also lead to a better scheme for compensating the turn.
Regarding the LQ-controller, the only way found to introduce cross-coupling dynamics would be to extend the linear model used to find the controller, i.e. to incorporate more

states. This was tried in an initial stage of this master thesis, but did not amount to anything. One problem being that many linearizations would have to be made to represent all equilibrium points, thus resulting in many different controllers. The point of the control strategy recommended in this thesis is that the controllers now implemented in the submarines and thus thoroughly tested, can be kept in place. The controller recommended here can be used by only adding a little code to implement the PID controller.

In the year 1995
Hans Nilsson

[1] K. J. Åström. *Reglerteori.* Almqvist & Wiksell, 1985.

[2] J. Moore B. Andersson. *Optimal Control, Linear Quadratic Methods.* Prentice-hall International Editions, 1989.

[3] J. G. Balchen D. D. Ruscio. A schur method for designing lq-optimal systems with prescribed eigenvalues. *Modeling, Identification and Control*, 2(1):55 – 72, 1990.

[4] J. Feldman. Dtnsrdc revised standard submarine equations of motion. Technical report, David W. Taylor Naval Ship Research and Development Center, 1979.

[5] T. I. Fossen. *Guidance of Ocean Vehicles.* John Wiley & Sons, 1994.

[6] The MathWorks Inc. *User's Guide*, March 1992.

[7] The MathWorks Inc. *External Interface Guide*, February 1993.

[8] H. K. Khalil. *Nonlinear Systems.* Macmillan Publishing, New York, 1992.

[9] M. Vidyasagar M. W. Spong. *Robot Dynamics and Control.* John Wiley & Sons, 1989.

7. Appendices

A: David Taylor Equations

Equations used to model the submarine.
Note that the integral $\frac{\rho}{2} l \overline{C}_L \int_{x_2}^{x_1} v(x)\, \overline{v}_{FM}(t - \tau[x])\, dx$, which appears in F_y,F_z,M_{roll},M_{pitch} and M_{yaw} in the original David Taylor equations is excluded, and replaced with Z_{vr}, Z_{vv} and M_{vr} in F_z and M_{pitch}.

Axial Force Equation (F_x)

$$
\begin{aligned}
& m[\dot{u} - vr + wq - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(pr - \dot{q})] \\
& = \\
& \frac{\rho}{2}[\\
& \quad L^4(X'_{qq}q^2 + X'_{rr}r^2 + X'_{rp}rp) \\
& \quad + L^3(X'_{\dot{u}}\dot{u} + X'_{vr}vr + X'_{wq}wq) \\
& \quad + L^2(X'_{vv}v^2 + X'_{ww}w^2) \\
& \quad + L^2(X'_{\delta r \delta r}u^2\delta_r^2 + X'_{\delta s \delta s}u^2\delta_s^2 + X'_{\delta b \delta b}u^2\delta_b^2) \\
& \quad] \\
& - (W - B)\sin\theta + F_{xp}
\end{aligned}
$$

Lateral Force Equation (F_y)

$$
\begin{aligned}
& m[\dot{v} - wp + ur + x_G(qp + \dot{r}) - y_G(r^2 + p^2) + z_G(qr - \dot{p})] \\
& = \\
& \frac{\rho}{2}[\\
& \quad L^4(Y'_{\dot{r}}\dot{r} + Y'_{\dot{p}}\dot{p} + Y'_{p|p|}p|p| + Y'_{pq}pq) \\
& \quad + L^3(Y'_{r}ur + Y'_{p}up + Y'_{\dot{v}}\dot{v} + Y'_{wp}wp) \\
& \quad + L^2(Y'_{*}u^2 + Y'_{v}uv + Y'_{v|v|R}v|(v^2 + w^2)^{1/2}) \\
& \quad + L^2(Y'_{\delta r}u^2\delta_r + Y'_{\delta r \eta}u^2\delta_r(\eta - \tfrac{1}{C})C) \\
& \quad C_d \int_l h(x)v(x)[(w(x))^2 + (v(x))^2]^{1/2}dx \\
& \quad] \\
& + (\mathrm{W} - \mathrm{B})\cos\theta\sin\phi
\end{aligned}
$$

Normal Force Equation (F_z)

$$
\begin{aligned}
& m[\dot{w} - uq + vp + x_G(rp - \dot{q}) + y_G(rq + \dot{p}) - z_G(p^2 + q^2)] \\
& =
\end{aligned}
$$

$$
\begin{aligned}
&\tfrac{\rho}{2}[\\
&\quad L^4(Z'_{\dot q}\dot q)\\
&\quad +L^3(Z'_{\dot w}\dot w + Z'_q uq + Z'_{vp}vp + Z'_{vr}vr)\\
&\quad +L^2(Z'_* u^2 + Z'_w uw + Z'_{vv}vv)\\
&\quad +L^2(Z'_{|w|}u|w| + Z'_{ww}|w(v^2+w^2)^{1/2}|\\
&\quad +L^2(Z'_{\delta s}u^2\delta_s + Z'_{\delta b}u^2\delta_b + Z'_{\delta s\eta}u^2\delta_s(\eta - \tfrac{1}{C})C)\\
&\quad -C_d\textstyle\int_l b(x)w(x)[(w(x))^2 + (v(x))^2]^{1/2}dx\\
&\quad]\\
&+ \text{(W - B)}\cos\theta\cos\phi
\end{aligned}
$$

Rolling Moment Equation (M_{roll})

$$
\begin{aligned}
&I_x\dot p + (I_z - I_y)qr - (\dot r + pq)I_{zz} + (r^2 - q^2)I_{yz} + (pr - \dot q)I_{xy} + m[y_G(\dot w - uq + vp) - z_G(\dot v - wp + ur)]\\
&=\\
&\tfrac{\rho}{2}[\\
&\quad L^5[K'_{\dot p}\dot p + K'_{\dot r}\dot r + K'_{qr}qr + K'_{p|p|}p|p|]\\
&\quad +L^4[K'_p up + K_r ur + K'_{\dot v}\dot v + K_{wp}wp]\\
&\quad +L^3[K'_* u^2 + K'_{vR}uv + K'_i uv_{FW}(t - \tau_T)]\\
&\quad +L^3[K'_{\delta r}u^2\delta_r + K'_{\delta r\eta}u^2\delta_r(\eta - \tfrac{1}{C})C]\\
&\quad +L^3(u^2 + v_S^2 + w_S^2)\beta_S^2[K'_{4S}\sin 4\phi_S + K'_{8S}\sin 8\phi_S]\\
&\quad +(y_G W - y_B B)\cos\theta\cos\phi - (z_G W - z_B B)\cos\theta\sin\phi\\
&-Q_p
\end{aligned}
$$

Pitching Moment equation (M_{pitch})

$$
\begin{aligned}
&I_y\dot q + (I_x - I_z)rp - (\dot p + qr)I_{xy} + (p^2 - r^2)I_{xz} + (qp - \dot r)I_{yz} + m[z_G(\dot u - vr + wq) - x_G(\dot w - uq + vp)]\\
&=\\
&\tfrac{\rho}{2}[\\
&\quad L^5[M'_{\dot q}\dot q + M'_{rp}rp]\\
&\quad +L^4[M'_{\dot w}\dot w + M'_q uq + M'_{vr}vr]\\
&\quad +L^3[M'_* u^2 + M'_w uw + M'_{w|w|R}w|(v^2+w^2)|^{1/2}]\\
&\quad +L^3\,[M'_{\delta s}u^2\delta_s + M'_{\delta b}u^2\delta_b + M'_{\delta s\eta}u^2\delta_s(\eta - \tfrac{1}{C})C]\\
&\quad +C_d\textstyle\int_l x\,b(x)\,w(x)\{[w(x)]^2 + [v(x)]^2\}^{1/2}dx\\
&\quad]\\
&-(x_G W - x_B B)\cos\theta\cos\phi - (z_G W - z_B B)\sin\theta
\end{aligned}
$$

Yawing Moment Equation (M_{yaw})

$$
\begin{aligned}
&I_z\dot r + (I_y - I_x)pq - (\dot q + rp)I_{yz} + (q^2 - p^2)I_{xy} + (rq - \dot p)I_{xz} + m[x_G(\dot v - wp + ur) - y_G(\dot u - vr + wq)]\\
&=\\
&\quad +L^5[N'_{\dot r}\dot r + N'_{\dot p}\dot p + N'_{pq}pq]\\
&\quad +L^4[N'_p up + N'_r ur + N'_{\dot v}\dot v]
\end{aligned}
$$

$$+L^3[N'_{\dot{u}}u^2 + N'_v uv + N'_{v|v|R}v|(v^2+w^2)^{\frac{1}{2}}|]$$
$$+L^3[N'_{\delta_r}u^2\delta_r + N'_{\delta_r\eta}u^2\delta_r(\eta - \tfrac{1}{C})C]$$
$$+C_d \int_l x\, h(x)\, v(x)\{[w(x)]^2 + [v(x)]^2\}^{1/2}dx$$
$$+(x_G W - x_B)\cos\theta\sin\phi + (y_G - y_B B)\sin\theta$$

B: A and B Matrix calculation (Maple)

```
> with(linalg):
```

```
> M:=evalm(matrix([[m-rho/2*l^3*ZDW,-rho/2*l^4*ZDQ],[-rho/2*l^4*MDW,Iy
> - rho/2*l^5*MDQ]]))
> ;
```

$$M := \begin{bmatrix} m - \frac{1}{2}\rho l^3 ZDW & -\frac{1}{2}\rho l^4 ZDQ \\ -\frac{1}{2}\rho l^4 MDW & Iy - \frac{1}{2}\rho l^5 MDQ \end{bmatrix}$$

```
> Minv:=evalm(inverse(M));
```

$$Minv := \begin{bmatrix} -2\frac{-2Iy + \rho l^5 MDQ}{\%1} & 2\frac{\rho l^4 ZDQ}{\%1} \\ 2\frac{\rho l^4 MDW}{\%1} & -2\frac{-2m + \rho l^3 ZDW}{\%1} \end{bmatrix}$$

$$\%1 := 4mIy - 2m\rho l^5 MDQ - 2\rho l^3 ZDW Iy + \rho^2 l^8 ZDW MDQ - \rho^2 l^8 ZDQ MDW$$

```
> C:=evalm(matrix([[rho/2*l^2*ZW,m+rho/2*l^3*ZQ],[rho/2*l^3*MW,rho/2*l^4*MQ]])*u);
```

$$C := \begin{bmatrix} \frac{1}{2}u\rho l^2 ZW & u\left(m + \frac{1}{2}\rho l^3 ZQ\right) \\ \frac{1}{2}u\rho l^3 MW & \frac{1}{2}u\rho l^4 MQ \end{bmatrix}$$

```
> MinvC:=evalm(Minv &* C);
```

$$MinvC := \left[\frac{u\rho l^2(2ZW Iy - ZW\rho l^5 MDQ + \rho l^5 ZDQ MW)}{\%1}, u\left(4mIy + 2Iy\rho l^3 ZQ - 2m\rho l^5 MDQ - \rho^2 l^8 MDQ ZQ + \rho^2 l^8 ZDQ MQ\right)/(\%1)\right]$$

$$\left[-\frac{u\rho l^3(-\rho l^3 MDW ZW - 2MW m + MW\rho l^3 ZDW)}{\%1}, -u\rho l^4 (-2MDW m - MDW\rho l^3 ZQ - 2MQ m + MQ\rho l^3 ZDW)/(\%1)\right]$$

$$\%1 := 4mIy - 2m\rho l^5 MDQ - 2\rho l^3 ZDW Iy + \rho^2 l^8 ZDW MDQ - \rho^2 l^8 ZDQ MDW$$

```
> G:=evalm(matrix([[0],[-(ZG*W-ZB*B)]]));
```

$$G := \begin{bmatrix} 0 \\ -ZGW + ZBB \end{bmatrix}$$

```
> MinvG:=evalm(Minv &* G);
```

$$MinvG :=$$
$$\left[-2\rho l^4 ZDQ\,(ZGW - ZBB)\Big/\left(4mIy - 2m\rho l^5 MDQ\right.\right.$$
$$\left.\left. - 2\rho l^3 ZDW\,Iy + \rho^2 l^8 ZDW\,MDQ - \rho^2 l^8 ZDQ\,MDW\right)\right]$$
$$\left[2(-2m + \rho l^3 ZDW)(ZGW - ZBB)\Big/\left(4mIy - 2m\rho l^5 MDQ\right.\right.$$
$$\left.\left. - 2\rho l^3 ZDW\,Iy + \rho^2 l^8 ZDW\,MDQ - \rho^2 l^8 ZDQ\,MDW\right)\right]$$

```
> A:=evalm(matrix([[MinvC[1,1],MinvC[1,2],MinvG[1,1],0],[MinvC[2,1],
> MinvC[2,2],MinvG[2,1],0],[0,1,0,0],[1,0,u,0]]));
```

$$A :=$$
$$\left[\frac{u\rho l^2(2\,ZW\,Iy - ZW\rho l^5 MDQ + \rho l^5 ZDQ\,MW)}{\%1}, u\left(4mIy\right.\right.$$
$$\left.\left. + 2\,Iy\rho l^3 ZQ - 2m\rho l^5 MDQ - \rho^2 l^8 MDQ\,ZQ + \rho^2 l^8 ZDQ\,MQ\right)\right.$$
$$\left.\Big/(\%1), -2\frac{\rho l^4 ZDQ\,(ZGW - ZBB)}{\%1}, 0\right]$$
$$\left[-\frac{u\rho l^3(-\rho l^3 MDW\,ZW - 2\,MW\,m + MW\rho l^3 ZDW)}{\%1}, -u\rho l^4\right.$$
$$(-2\,MDW\,m - MDW\rho l^3 ZQ - 2\,MQ\,m + MQ\rho l^3 ZDW)\Big/(\%1),$$
$$\left.2\frac{(-2m + \rho l^3 ZDW)(ZGW - ZBB)}{\%1}, 0\right]$$
$$[0, 1, 0, 0]$$
$$[1, 0, u, 0]$$
$$\%1 := 4mIy - 2m\rho l^5 MDQ - 2\rho l^3 ZDW\,Iy + \rho^2 l^8 ZDW\,MDQ$$
$$- \rho^2 l^8 ZDQ\,MDW$$

```
> DD:=evalm(matrix([[rho/2*l^2*ZDS,rho/2*l^2*ZDB],[rho/2*l^3*MDS,
> rho/2*l^3*MDB]])*u^2);
```

$$DD := \begin{bmatrix} \frac{1}{2}u^2\rho l^2 ZDS & \frac{1}{2}u^2\rho l^2 ZDB \\ \frac{1}{2}u^2\rho l^3 MDS & \frac{1}{2}u^2\rho l^3 MDB \end{bmatrix}$$

```
> MinvDD:=evalm(Minv &* DD);
```

$$MinvDD :=$$

$$\left[\frac{u^2 \rho l^2 (2 ZDS Iy - ZDS \rho l^5 MDQ + \rho l^5 ZDQ MDS)}{\%1},\right.$$
$$\left.\frac{u^2 \rho l^2 (2 ZDB Iy - ZDB \rho l^5 MDQ + \rho l^5 ZDQ MDB)}{\%1}\right]$$
$$\left[-\frac{\rho l^3 u^2 (-\rho l^3 MDW ZDS - 2 MDS m + MDS \rho l^3 ZDW)}{\%1},\right.$$
$$\left.-\frac{\rho l^3 u^2 (-\rho l^3 MDW ZDB - 2 MDB m + MDB \rho l^3 ZDW)}{\%1}\right]$$
$$\%1 := 4 m Iy - 2 m \rho l^5 MDQ - 2 \rho l^3 ZDW Iy + \rho^2 l^8 ZDW MDQ$$
$$- \rho^2 l^8 ZDQ MDW$$

```
> B:=evalm(matrix([[MinvDD[1,1],MinvDD[1,2]],[MinvDD[2,1],MinvDD[2,2]],[0,0],[0,0]]));
```

$$B :=$$
$$\left[\frac{u^2 \rho l^2 (2 ZDS Iy - ZDS \rho l^5 MDQ + \rho l^5 ZDQ MDS)}{\%1},\right.$$
$$\left.\frac{u^2 \rho l^2 (2 ZDB Iy - ZDB \rho l^5 MDQ + \rho l^5 ZDQ MDB)}{\%1}\right]$$
$$\left[-\frac{\rho l^3 u^2 (-\rho l^3 MDW ZDS - 2 MDS m + MDS \rho l^3 ZDW)}{\%1},\right.$$
$$\left.-\frac{\rho l^3 u^2 (-\rho l^3 MDW ZDB - 2 MDB m + MDB \rho l^3 ZDW)}{\%1}\right]$$
$$[0, 0]$$
$$[0, 0]$$
$$\%1 := 4 m Iy - 2 m \rho l^5 MDQ - 2 \rho l^3 ZDW Iy + \rho^2 l^8 ZDW MDQ$$
$$- \rho^2 l^8 ZDQ MDW$$

C: Testmex mex code

```
/*
 *
 *         Syntax  [sys, x0] = simomex(t,x,u,flag)
 *
 */

/*
 * The following #define is used to specify the name of this S-Function.
 */

#define S_FUNCTION_NAME testmex

/*
 * need to include simstruc.h for the definition of the SimStruct and
 * its associated macro definitions.
 */

#include "simstruc.h"
#include "mex.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "fpar.c"
#include "flow.c"

/*
 * mdlInitializeSizes - initialize the sizes array
 *
 * The sizes array is used by SIMULINK to determine the S-function block's
 * characteristics (number of inputs, outputs, states, etc.).
 */

static void mdlInitializeSizes(S)
    SimStruct *S;
{
    ssSetNumContStates(    S, 12); /* number of continuous states */
    ssSetNumDiscStates(    S, 0);  /* number of discrete states */
    ssSetNumInputs(        S, 4);  /* number of inputs */
    ssSetNumOutputs(       S, 12); /* number of outputs */
    ssSetDirectFeedThrough(S, 0);  /* direct feedthrough flag */
    ssSetNumSampleTimes(   S, 1);  /* number of sample times */
    ssSetNumInputArgs(     S, 0);  /* number of input arguments */
    ssSetNumRWork(         S, 0);  /* number of real work vector elements*/
    ssSetNumIWork(         S, 0); /* number of integer work vector elements*/
    ssSetNumPWork(         S, 0); /* number of pointer work vector elements*/
```

```
}

/*
 * mdlInitializeSampleTimes - initialize the sample times array
 *
 * This function is used to specify the sample time(s) for your S-function.
 * If your S-function is continuous, you must specify a sample time of 0.0.
 * Sample times must be registered in ascending order.
 */

static void mdlInitializeSampleTimes(S)
    SimStruct *S;
{
    ssSetSampleTimeEvent(S, 0, 0.0);
    ssSetOffsetTimeEvent(S, 0, 0.0);
}

/*
 * mdlInitializeConditions - initialize the states
 *
 * In this function, you should initialize the continuous and discrete
 * states for your S-function block.  The initial states are placed
 * in the x0 variable.  You can also perform any other initialization
 * activities that your S-function may require.
 */

static Matrix *Minv;

static void mdlInitializeConditions(x0, S)
    double *x0;
    SimStruct *S;
{
    Matrix *MM, *Mout[2], *slask[1];
    double *Mpr;
    double L2, L3, L4, L5;

    fpar_init();

    MM = mxCreateFull(6,6,0);
    Mpr = mxGetPr(MM);

    L2 = L*L;
    L3 = L*L*L;
    L4 = L*L*L*L;
    L5 = L*L*L*L*L;

/* u-equation */
```

```
/* u */    Mpr[0] = M-(RHO/2)*L3*XUD;
/* v */    Mpr[6] = 0;
/* w */    Mpr[12]= 0;
/* p */    Mpr[18]= 0;
/* q */    Mpr[24]= M*ZG;
/* r */    Mpr[30]= -M*YG;

/* v-equation */

/* u */    Mpr[1] = 0;
/* v */    Mpr[7]= M-(RHO/2)*L3*YVD;
/* w */    Mpr[13]= 0;
/* p */    Mpr[19]= -M*ZG-(RHO/2)*L4*YPD;
/* q */    Mpr[25]= 0;
/* r */    Mpr[31]= M*XG-(RHO/2)*L4*YRD;

/* w-equation */

/* u */    Mpr[2] = 0;
/* v */    Mpr[8]= 0;
/* w */    Mpr[14]= M-(RHO/2)*L3*ZWD;
/* p */    Mpr[20]= M*YG;
/* q */    Mpr[26]= -M*XG-(RHO/2)*L4*ZQD;
/* r */    Mpr[32]= 0;

/* p-equation */

/* u */    Mpr[3] = 0;
/* v */    Mpr[9]= -M*ZG-RHO/2*L4*KVD;
/* w */    Mpr[15]= M*YG;
/* p */    Mpr[21]= IX-(RHO/2)*L5*KPD;
/* q */    Mpr[27]= -IXY;
/* r */    Mpr[33]= -IZX-(RHO/2)*L5*KRD;

/* q-equation */

/* u */    Mpr[4] = M*ZG;
/* v */    Mpr[10]= 0;
/* w */    Mpr[16]= -M*XG-(RHO/2)*L4*MWD;
/* p */    Mpr[22]= -IXY;
/* q */    Mpr[28]= IY-(RHO/2)*L5*MQD;
/* r */    Mpr[34]= -IYZ;

/* r-equation */

/* u */    Mpr[5] = -M*YG;
/* v */    Mpr[11]= M*XG-(RHO/2)*L4*NVD;
/* w */    Mpr[17]= 0;
```

```
/* p */     Mpr[23]= -IZX-(RHO/2)*L5*NPD;
/* q */     Mpr[29]= -IYZ;
/* r */     Mpr[35]= IZ-(RHO/2)*L5*NRD;


    mexCallMATLAB(1,&Minv,1,&MM,"inv");
    mxFreeMatrix(MM);

    flow_init();

    x0[0] = U0;
    x0[1] = 0;
    x0[2] = 0;
    x0[3] = 0;
    x0[4] = 0;
    x0[5] = 0;
    x0[6] = 0;
    x0[7] = 0;
    x0[8] = 7;
    x0[9] = 0;
    x0[10]= 0;
    x0[11]= 0;

}

/*
 * mdlOutputs - compute the outputs
 *
 * In this function, you compute the outputs of your S-function
 * block.  The outputs are placed in the y variable.
 */

static void mdlOutputs(y, x, u, S, tid)
    double *y, *x, *u;
    SimStruct *S;
    int tid;
{
    y[0] = x[0];
    y[1] = x[1];
    y[2] = x[2];
    y[3] = x[3];
    y[4] = x[4];
    y[5] = x[5];
    y[6] = x[6];
    y[7] = x[7];
    y[8] = x[8];
    y[9] = x[9];
    y[10] = x[10];
```

```
    y[11] = x[11];
}

/*
 * mdlUpdate - perform action at major integration time step
 *
 * This function is called once for every major integration time step.
 * Discrete states are typically updated here, but this function is useful
 * for performing any tasks that should only take place once per
 * integration step.
 */

static void mdlUpdate(x, u, S, tid)
    double *x, *u;
    SimStruct *S;
    int tid;
{
}

/*
 * mdlDerivatives - compute the derivatives
 *
 * In this function, you compute the S-function block's derivatives.
 * The derivatives are placed in the dx variable.
 */

static void mdlDerivatives(dxvec, xvec, uvec, S, tid)
    double *dxvec, *xvec, *uvec;
    SimStruct *S;
    int tid;
{
    double *Minvpr;
    Matrix *MN,*Mout[1];
    double *Mpn;

    double ds = uvec[0];
    double db = uvec[1];
    double dr = uvec[2];
    double rps= uvec[3];

    double u = xvec[0];
    double v = xvec[1];
    double w = xvec[2];
    double p = xvec[3];
    double q = xvec[4];
    double r = xvec[5];
    double x = xvec[6];
    double y = xvec[7];
```

```
double z = xvec[8];
double fi = xvec[9];
double th = xvec[10];
double psi= xvec[11];

double dx, dy, dz, dfi, dth, dpsi;

int nauxvec = 6;
double auxvec[6];

int i, j;
double accum;
double J,DIAM4,X1,X2,Tp,Qp;

double L2, L3, L4, L5;
double u2_eta;

DIAM4=DIAM*DIAM*DIAM*DIAM;

J=u*(1-WAKE)/(rps*DIAM);
X1=RHO*DIAM4*(KT0+KTJ*J+KTJJ*J*J);
X2=RHO*DIAM4*DIAM*(KQ0+KQJ*J+KQJJ*J*J);
Tp=X1*rps*fabs(rps)*(1-TD);
Qp=X2*rps*fabs(rps);

L2 = L*L;
L3 = L*L*L;
L4 = L*L*L*L;
L5 = L*L*L*L*L;

u2_eta = VC*fabs(rps)/rps*u;

/* u-equation */

auxvec[0] = -M*(-v*r + w*q - XG*(q*q + r*r) + YG*(p*q) + ZG*(p*r)) +
      RHO/2*(
              L4*(XQQ*q*q + XQAQ*q*fabs(q) + XRR*r*r + XRP*r*p)+
      L3*(XVR*v*r + XWQ*w*q + XWAQ*w*fabs(q))+
      L2*(XUU*u*u + XVV*v*v + XWW*w*w + XWAW*w*fabs(w) +
        4*XDRDR*u*u*dr*dr +
                4*XDS*u*u*ds +
                4*XDSDS*u*u*ds*ds +
        XDBDB*u*u*db*db)
              )
              -(W-B)*sin(th)
      +Tp;

/* v-equation */
```

```
auxvec[1] = -M*(-w*p + u*r - YG*(r*r + p*p) + ZG*(q*r) + XG*(p*q)) +
      RHO/2*(
              L4*(YRAR*r*fabs(r) + YPQ*p*q + 4*YRDR*r*r*dr) +
              L3*(YR*u*r + YP*u*p + YWP*w*p) +
      L2*(YSTAR*u*u + YV*u*v + YVAV*v*sqrt(v*v + w*w) +
        4*YVDR*v*v*dr + 4*YDR*u*u*dr + 4*YDAD*u*u*dr*fabs(dr)+
                4*YDRE*dr*(u2_eta - u*u/C)*C)
              )
              + (W-B)*cos(th)*sin(fi) -
      RHO/2*CDY*y_int(v,w,q,r);

auxvec[2] = -M*(-u*q + v*p - ZG*(p*p + q*q) + XG*(r*p) + YG*(r*q)) +
            RHO/2*(
            L4*(ZQAQ*q*fabs(q) + ZRR*r*r)+
            L3*(ZQ*u*q + ZVP*v*p + ZVR*v*r + 4*ZAQDS*u*fabs(q)*ds)+
            L2*(ZSTAR*u*u + ZW*u*w + ZVV*v*v + 4*ZWDS*w*w*ds +
              ZAW*u*fabs(w) +
              ZWW*fabs(w*sqrt(v*v + w*w)) + ZWB*u*w*db +
              ZAWB*u*fabs(w)*db + 4*ZDS*u*u*ds +4*ZDAD*u*u*ds*fabs(ds)+
              ZDB*u*u*db + ZBAB*u*u*db*fabs(db) +
              4*ZDSE*ds*(u2_eta - u*u/C)*C)
            )
            + (W-B)*cos(th)*cos(fi) -
            RHO/2*CDZ*z_int(v,w,q,r);

auxvec[3] = -((IZ - IY)*q*r - (p*q)*IZX + (r*r - q*q)*IYZ + (p*r)*IXY)
            -M*(YG*(-u*q + v*p) - ZG*(-w*p + u*r)) +
    RHO/2*(
            L5*(KQR*q*r + KRAR*r*fabs(r)) +
            L4*(KP*u*p + KR*u*r + KWP*w*p)+
    L3*(KSTAR*u*u + KV*u*v + KVAV*v*fabs(v) + 4*KDR*u*u*dr)
            )
            - (YG*W-YB*B)*cos(th)*cos(fi)
            - (ZG*W-ZB*B)*cos(th)*sin(fi)
    - Qp;

auxvec[4] = -((IX - IZ)*r*p - (q*r)*IXY + (p*p - r*r)*IZX + (p*q)*IYZ)
           -M*(ZG*(-v*r + w*q) - XG*(-u*q + v*p)) +
           RHO/2*(
           L5*(MRP*r*p + MQAQ*q*fabs(q) + MRR*r*r) +
           L4*(MQ*u*q + MVR*v*r + 4*MAQDS*u*fabs(q)*ds)+
   L3*(MSTAR*u*u + MW*u*w + MWAW*w*sqrt(v*v + w*w) +
             4*MWDS*w*w*ds + MAW*u*fabs(w) +
             MWW*fabs(w*sqrt(v*v + w*w)) +
             MVV*v*v + MWB*u*w*db +
             MAWB*u*fabs(w)*db + 4*MDS*u*u*ds + 4*MDAD*u*u*ds*fabs(ds)+
             MDB*u*u*db + MBAB*u*u*db*fabs(db) +
```

```
            4*MDSE*ds*(u2_eta - u*u/C)*C )
          )
   -(XG*W-XB*B)*cos(th)*cos(fi)
   -(ZG*W-ZB*B)*sin(th) +
          RHO/2*CDM*m_int(v,w,q,r);

auxvec[5] = -((IY - IX)*p*q - (r*p)*IYZ + (q*q - p*p)*IXY + (r*q)*IZX)
            -M*(XG*(-w*p + u*r) - YG*(-v*r + w*q)) +
            RHO/2*(
            L5*(NPQ*p*q + NRAR*r*fabs(r) + 4*NRDR*r*r*dr) +
            L4*(NP*u*p + NR*u*r) +
    L3*(NSTAR*u*u + NV*u*v + NVAV*v*sqrt(v*v+w*w) +
              4*NVDR*v*v*dr + 4*NDR*u*u*dr + 4*NDAD*u*u*dr*fabs(dr) +
              4*NDRE*dr*(u2_eta - u*u/C)*C)
            ) +
            (XG*W - XB*B)*cos(th)*sin(fi) + (YG*W - YB*B)*sin(th)
            - RHO/2*CDN*n_int(v,w,q,r);

Minvpr = mxGetPr(Minv);

for (i = 0; i < 6; i++) {
   accum = 0.0;
   for (j = 0; j < 6; j++)
      accum = accum+(Minvpr[6*j + i] * auxvec[j]);
   dxvec[i] = accum;
 }

dx =  cos(psi)*cos(th)*u+(-sin(psi)*cos(fi)+cos(psi)*
         sin(th)*sin(fi))*v+(sin(psi)*sin(fi)+cos(psi)*
         cos(fi)*sin(th))*w;
dy =  sin(psi)*cos(th)*u+(cos(psi)*cos(fi)+sin(fi)*
         sin(th)*sin(psi))*v+(-cos(psi)*sin(fi)+sin(th)*
         sin(psi)*cos(fi))*w;
dz =  -sin(th)*u+cos(th)*sin(fi)*v+cos(th)*cos(fi)*w;
dfi = p+sin(fi)*tan(th)*q+cos(fi)*tan(th)*r;
dth=  cos(fi)*q-sin(fi)*r;
dpsi=  (q*sin(fi))/cos(th)+(r*cos(fi))/cos(th);

dxvec[6] = dx;
dxvec[7] = dy;
dxvec[8] = dz;
dxvec[9] = dfi;
dxvec[10] = dth;
dxvec[11] = dpsi;
```

```
}

/*
 * mdlTerminate - called when the simulation is terminated.
 *
 * In this function, you should perform any actions that are necessary
 * at the termination of a simulation.  For example, if memory was
 * allocated in mdlInitializeConditions, this is the place to free it.
 */

static void mdlTerminate(S)
    SimStruct *S;
{
    mxFreeMatrix(Minv);
}
#ifdef MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```