An Adaptive Autopilot with Feedforward Compensation for Wave Disturbances Applied to Ship Steering

Kvist, Pär; Ruijter, Hendrik

1988

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7387)/1–24/(1988)

An Adaptive Autopilot with Feedforward Compensation for Wave Disturbances applied to Ship Steering

Pär Kvist
Hendrik Ruijter

Department of Automatic Control
Lund Institute of Technology
April 1988

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

Document name
Internal report

Date of issue
April 1988

Document Number
CODEN: LUTFD2/(TFRT-7387)/1-24/(1988)

Author(s)
Pär Kvist and Hendrik Ruijter

Supervisor
Karl Johan Åström

Sponsoring organisation

Title and subtitle
An Adaptive Autopilot with Feedforward Compensation for Wave Disturbances applied to Ship Steering

Abstract

This report describes the design of, and experiments with, an adaptive autopilot for ship steering. The autopilot is designed as an indirect adaptive regulator based on pole placement. A feedforward compensation term in the regulator is used to suppress wave disturbances.

Key words
Adaptive regulator, Autopilot, Feedforward, Ship steering

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
24

Recipient's notes

Security classification

Contents

1. Introduction

During design of autopilots for ships it has been noticed that it is not sufficient to use a fixed controller. The dynamics of a ship can vary considerable due to e.g. weather conditions, different load and velocity. In fact, under certain circumstances, not even stability can be guaranteed.

In this report an autopilot based on an adaptive controller is designed. Wave disturbances are suppressed with a feedforward compensation term in the controller. Simulations in the simulation language SIMNON are done. The report is structured as follows. In Chapter 2 the basic equation of motion for a ship is discussed. Chapter 3 is devoted to the design of the autopilot, that is, regulator design and design of the feedforward compensation. Some experiments are done in Chapter 4, where we use parameters from real ships. In Chapter 5 there is a final discussion and conclusions are drawn. Some useful references are given in Chapter 6. Finally, in Appendix the SIMNON program used is listed.

2. Ship Steering Dynamics

A very simple mathematical model of ship steering dynamics, due to Nomoto, relating rudder angle, δ, and disturbing torque, e, generated by waves, to turning rate, r, and to heading angle, ψ, can be stated as a system of linear ordinary differential equations (Källström, 1979),

$$\frac{d}{dt}\begin{pmatrix} r(t) \\ \psi(t) \end{pmatrix} = \begin{pmatrix} -\alpha & 0 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} r(t) \\ \psi(t) \end{pmatrix} + \begin{pmatrix} K & C \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \delta(t-\tau) \\ e(t) \end{pmatrix} \tag{2.1}$$

where higher order dynamics are modelled as a time delay, τ, in the system, see Figure (2.1). This model is a linearization of a more complex nonlinear model, and the parameters, α, K, and C, will depend on the working conditions. Eliminating r from (2.1) gives

$$\psi(t) = \frac{Ke^{-p\tau}}{p(p+\alpha)}\delta(t) + \frac{C}{p(p+\alpha)}e(t) \tag{2.2}$$

where $p = d/dt$ is the differential operator.

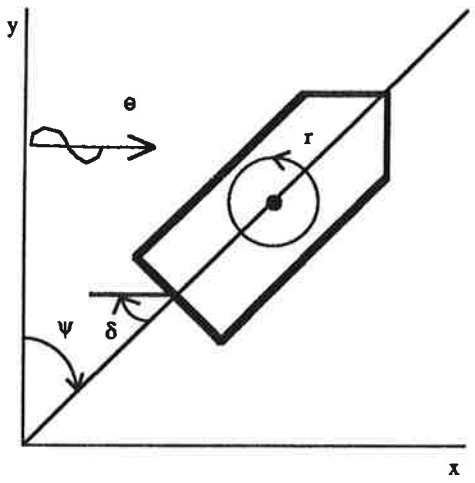

Figure 2.1 Heading angle, ψ, turning rate, r, rudder angle, δ, and disturbing torque, e, generated by waves

Sampling (2.2) with sampling rate $1/h$ under the assumption that $h > \tau$ yields

$$\psi(k) = \frac{b_1 q^2 + b_2 q + b_3}{q(q-1)(q-a)}\delta(k) + \frac{c_1 q + c_2}{(q-1)(q-a)}e(k) \tag{2.3}$$

where the sampling period is used as time unit and q is the forward shift operator. The discrete time parameters a, b_1, b_2, b_3, c_1, and c_2 can be expressed as functions of K, α, τ, C, and h, but for our purposes it is sufficient to treat the ship steering dynamics based on knowledge of the discrete time parameters only.

3. Autopilot Structure

The autopilot is designed as an indirect self tuning regulator based on pole placement. The regulator is composed of three parts, a parameter estimator, a design calculation unit, and an implementation of the control law. In the following two sections, the estimator and the regulator structure will be discussed. In Figure (3.1) a block diagram of the ship and the autopilot can be seen.

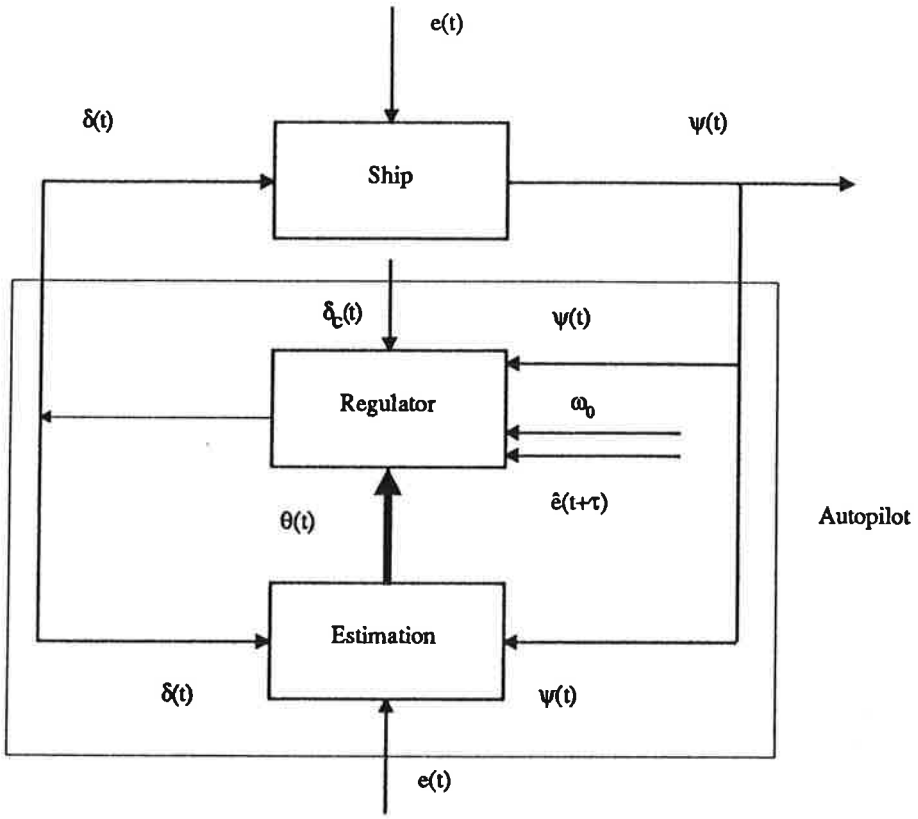

Figure 3.1 A block diagram of the system involving ship and autopilot

3.1 Parameter Estimation

When designing the estimator it is advantaguous to use all the a priori knowledge possible. From Equation (2.3) we have structural knowledge of the system. Equation (2.3) can be written as

$$q(q-1)(q-a)\psi(k) = \left(b_1 q^2 + b_2 q + b_3\right)\delta(k) + (c_1 q^2 + c_2 q)e(k) \qquad (3.1)$$

Rearranging (3.1) and introducing $\Delta\psi(k) = (1 - q^{-1})\psi(k)$ yields

$$\left(1 - aq^{-1}\right)\Delta\psi(k) = \left(b_1q^{-1} + b_2q^{-2} + b_3q^{-3}\right)\delta(k) + \left(c_1q^{-1} + c_2q^{-2}\right)e(k) \tag{3.2}$$

Equation (3.2) can be written as

$$\Delta\psi(k) = \varphi^T(k-1)\,\theta \tag{3.3}$$

where we have defined the regression vector, φ, and the parameter vector, θ, as

$$\begin{aligned} \varphi(k) &= \begin{pmatrix} \Delta\psi(k) & \delta(k) & \delta(k-1) & \delta(k-2) & e(k) & e(k-1) \end{pmatrix}^T \\ \theta &= \begin{pmatrix} a & b_1 & b_2 & b_3 & c_1 & c_2 \end{pmatrix}^T \end{aligned} \tag{3.4}$$

It is now straightforward to apply the recursive least mean square algorithm to (3.3). The recursive least mean square algorithm can be written as

$$\begin{aligned} \widehat{\theta}(k) &= \widehat{\theta}(k-1) + K(k)\left(\Delta\psi(k) - \varphi^T(k)\widehat{\theta}(k-1)\right) \\ K(k) &= P(k-1)\varphi(k)\left(\lambda + \varphi^T(k)P(k-1)\varphi(k)\right)^{-1} \\ P(k) &= \left(I - K(k)\varphi^T(k)\right)P(k-1)/\lambda \end{aligned} \tag{3.5}$$

where λ is a forgetting factor. In the basic recursive equations problems are associated with the covariance matrice when there are long periods with no excitation. To make sure that the covariances stay bounded an extended algorithm can be used for estimation of the parameters (Åström and Wittenmark, 1988)

$$\begin{aligned} \widehat{\theta}(k) &= \widehat{\theta}(k-1) + a(k)K(k)\left(\Delta\psi(k) - \varphi^T(k)\widehat{\theta}(k-1)\right) \\ K(k) &= P(k-1)\varphi(k)\left(1 + \varphi^T(k)P(k-1)\varphi(k) + \bar{c}\varphi^T(k)\varphi(k)\right)^{-1} \\ \bar{P}(k) &= \bar{P}(k-1) - a(k)\frac{P(k-1)\varphi(k)\varphi^T(k)P(k-1)}{1 + \varphi^T(k)P(k-1)\varphi(k) + \bar{c}\varphi^T(k)\varphi(k)} \\ P(k) &= c_1\frac{\bar{P}(k)}{\mathrm{tr}\bar{P}(k)} + c_2I \\ a(k) &= \begin{cases} \bar{a}, & \text{if } |\Delta\psi(k) - \varphi^T(k)\widehat{\theta}(k-1)| > 2\delta \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3.6}$$

In this algorithm the parameters should be choosen such that $c_1 > 0$, $c_2 \geq 0$, $\bar{c} \geq 0$ and $\bar{a} \in [0.1 \ldots 0.5]$. δ is an estimate of the magnitude of the noise. Compared to the standard implementation of the recursive least squares algorithm with forgetting factor (3.5), the extended method (3.6) leads to slower convergence for the parameter estimates, but the conditional updating gives a security against P matrix explosion, which may occur in the standard algorithm if excitation is poor for a long period. Hence, if we use (3.5) until we get good estimates and then switch algorithm to (3.6), we avoid both slow convergence and P matrix explosion. If we choose $a(k) = 1$, $\bar{c} = 0$, $c_1 = \mathrm{tr}\bar{P}(k)$, and $c_2 = 0$ in (3.6) we get (3.5) with $\lambda = 1$, that is, with no forgetting factor. Thus, it is simple to implement the switching technique.

3.2 Regulator Structure

The process to be controlled can be described by the equation

$$A(q)y(k) = B(q)\delta(k) + C(q)e(k) \tag{3.7}$$

where the polynomials $A(q)$, $B(q)$, and $C(q)$ are given by (3.1). We postulate a linear control law of the form

$$R(q)\delta(k) = T(q)\delta_c(k) - S(q)y(k) + F(q)\hat{e}(k+\tau) \tag{3.8}$$

where $\hat{e}(k+\tau)$ is a estimate of $e(k+\tau)$. Eliminating the control signal, $\delta(k)$, in (3.7) and (3.8) yields

$$y(k) = \frac{BT}{AR+BS}\delta_c(k) + \frac{1}{AR+BS}\left(BF\hat{e}(k+\tau) + RCe(k)\right) \tag{3.9}$$

where the argument in the polynomials have been suppressed. The problem is now to select the polynomials $R(q)$, $S(q)$, $T(q)$, and $F(q)$. In the first subsection below the polynomials $R(q)$, $S(q)$, $T(q)$ will be chosen according to a pole placement procedure. How to select the feedforward polynomial $F(q)$, is then discussed in the next subsection.

Regulator Design

Here we only concentrate on the first term on the right hand side of (3.9). The purpose is to select the regulator polynomials such that the closed loop system is

$$y(k) = \frac{B_m(q)}{A_m(q)}\delta_c(k) \tag{3.10}$$

where the polynomials $B_m(q)$ and $A_m(q)$ have to be chosen by the designer. Equations (3.9) and (3.10) give the Diophantine equation

$$A(q)R(q) + B(q)S(q) = A_o(q)A_m(q) \tag{3.11}$$

and

$$T(q) = \frac{A_m(1)}{B(1)}A_o(q) \tag{3.12}$$

where the observer polynomial $A_o(q)$ has been introduced, and also has to be chosen by the designer. Demanding a causal regulator and using the fact that the time delay in the closed loop system cannot be less than the time delay in the open loop system results in the polynomials (Åström and Wittenmark, 1984)

$$\begin{aligned} A_o(q) &= q^3 + a_{o1}q^2 + a_{o2}q + a_{o3} \\ A_m(q) &= q^3 + a_{m1}q^2 + a_{m2}q + a_{m3} \\ B_m(q) &= \frac{A_m(1)}{B(1)}B(q) \\ S(q) &= s_0q^3 + s_1q^2 + s_2q + s_3 \\ R(q) &= (q-1)(q^2 + r_0q + r_1) \end{aligned} \tag{3.13}$$

where we have avoided cancellation of process zeros and forced an integration action on the regulator. The regulator polynomials are then given as the

solution of the Diophantine equation (3.11) with polynomials as in (3.12) and (3.13).

It is advantaguous to make the specifications in continuous time and then transform the polynomials into discrete time. Here we use

$$A_m^c(s) = (s + \alpha_m\omega_m)(s^2 + 2\zeta_m\omega_m s + \omega_m^2)$$

and the relations

$$\begin{aligned} a_{m1} &= p_3 + p_1 \\ a_{m2} &= p_3p_1 + p_2 \\ a_{m3} &= p_3p_2 \end{aligned}$$

with

$$\begin{aligned} p_1 &= -2e^{-\zeta_m\omega_m h}\cos(\sqrt{1-\zeta_m^2}\omega_m h) \\ p_2 &= e^{-2\zeta_m\omega_m h} \\ p_3 &= -e^{-\alpha_m\omega_m h} \end{aligned}$$

The corresponding relations are also valid for the observer polynomial $A_o(q)$.

The control law (3.8) can be written as

$$R^*(q^{-1})\delta(k) = T^*(q^{-1})\delta_c(k) - S^*(q^{-1})\psi(k) + F^*(q^{-1})\hat{e}(k+\tau) \qquad (3.14)$$

where we have used the reciprocal polynomials. A practical implementation of the control law (3.14), which compensates for antireset windup, is given by (Åström and Wittenmark, 1988)

$$\begin{aligned} A_o^*v(k) &= T^*\delta_c(k) - S^*\psi(k) + (A_o^* - R^*)\,\delta(k) + F^*\hat{e}(k+\tau) \\ \delta(k) &= \mathrm{sat}(v(k)) \end{aligned}$$

Feedforward Compensation

If $\hat{e}(k+\tau)$ is a good estimate of $e(k+\tau)$ and if the major energy of the waves is centered around the frequency ω_0, the wave disturbance influence on the heading angle can be made small by selecting the polynomial $F(q)$ in a proper way. In the following we will assume that the above holds and that ω_0 and $\hat{e}(k+\tau)$ are at our disposal. Taking account of the time delay in the control signal and using (3.9), gives the condition

$$B(q)F(q) + R(q)C(q) = 0 \qquad (3.15)$$

Using the fact that the disturbance is almost periodic, with known frequency, gives that (3.15) can be simplified to

$$B(e^{i\omega_0})F(e^{i\omega_0}) + R(e^{i\omega_0})C(e^{i\omega_0}) = 0 \qquad (3.16)$$

that is, we only demand perfect fitting at the frequency ω_0. Setting real- and imaginary parts of (3.16) equal zero, with $F(q) = f_0q^3 + f_1q^2$, yields

$$\begin{aligned} &\mathrm{Re}\{B(e^{i\omega_0})\}\mathrm{Re}\{F(e^{i\omega_0})\} - \mathrm{Im}\{B(e^{i\omega_0})\}\mathrm{Im}\{F(e^{i\omega_0})\} + \\ &\mathrm{Re}\{R(e^{i\omega_0})\}\mathrm{Re}\{C(e^{i\omega_0})\} - \mathrm{Im}\{R(e^{i\omega_0})\}\mathrm{Im}\{C(e^{i\omega_0})\} = 0 \\ &\mathrm{Im}\{B(e^{i\omega_0})\}\mathrm{Re}\{F(e^{i\omega_0})\} + \mathrm{Re}\{B(e^{i\omega_0})\}\mathrm{Im}\{F(e^{i\omega_0})\} + \\ &\mathrm{Im}\{R(e^{i\omega_0})\}\mathrm{Re}\{C(e^{i\omega_0})\} + \mathrm{Re}\{R(e^{i\omega_0})\}\mathrm{Im}\{C(e^{i\omega_0})\} = 0 \end{aligned} \qquad (3.17)$$

Solving (3.17) for f_0 and f_1 gives the solution. The solution is a rather complicated expression, and it is programmed in Appendix.

Solution of the Diophantine Equation

The design method used is pole placement. To solve the Diophantine equation

$$A(q)R(q) + B(q)S(q) = A_o(q)A_m(q)$$

is the same as solving a set of linear equations. If there are common factors in the A and B polynomials it is necessary to eliminate them to get a solution. The design of the regulator will be exactly the same but the degree for the A_o, A_m, S, and R polynomials will be reduced with 1 for every common factor. This means tedious calculations for the special cases when $b_3 = 0$, $b_2 = 0$ and $b_3 = 0$, etc.

4. Experiments

In this chapter the autopilot, designed in Chapter 3, is used on different ships, in order to establish the robustness of the design method. In the following experiments are done with a minesweeper and a cargo ship. Both regulation with and without feedforward is treated. The ship velocity is 10 m/s. The values we have used in the experiments are

- Minesweeper $\ell = 55$, $a = -0.14$, and $b = -1.4$.
- Cargo $\ell = 161$, $a = 0.19$, and $b = -1.63$.

The corresponding relationship between these parameters and the parameters used in Chapter 2 are

$$\alpha = \frac{ua}{\ell}$$

$$K = \frac{u^2 b}{\ell^2}$$

where u is the ship velocity. This corresponds to a scaling of the parameters. In these experiments, a high amplitude square wave is exciting the system in open loop in the first 20 time units. This is done to avoid large transients in the heading angle. In a real situation reasonable parameters might be known, and could be used as start values in the estimation. Another possibility is to let the ship be manually steered, with the estimation turned on, until good parameters are available, and then switch on the autopilot.

The results of simulations of a minesweeper are shown in Figures (4.1) and (4.2). Figure (4.1) shows a simulation without feedforward from waves, and Figure (4.2) shows a simulation, with the same wave disturbance acting on the ship, with feedforward compensation. As can be seen, the feedforward gives much better performance on the heading deviation. As can be expected, from the nature of feedforward, the control signal contains a high frequency component, but has a lower amplitude in general.

Better performance is obtained with feedforward than without, doing experiment with a cargo ship. The results are not as good as with the minesweeper, though, see Figure (4.3) and (4.4). Even here a high frequency component is involved in the control signal.

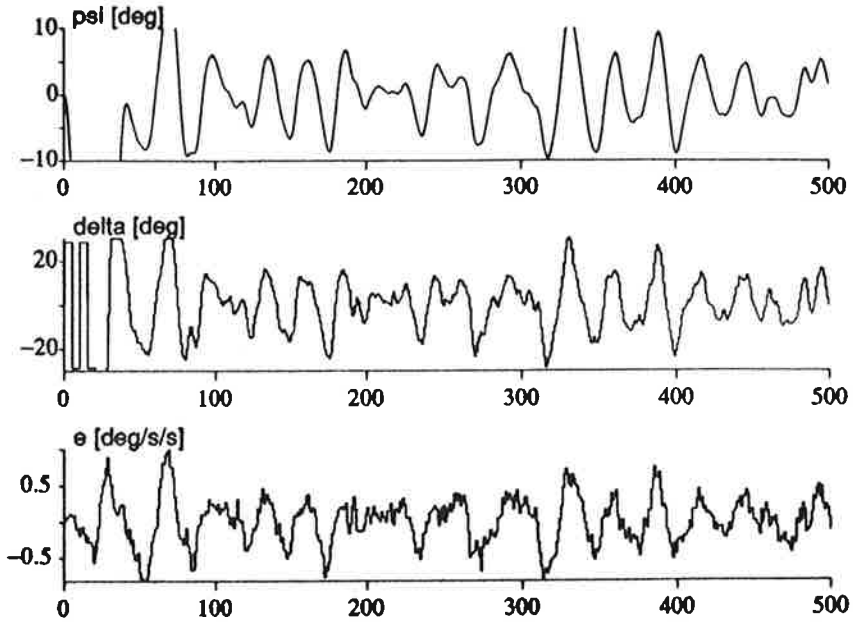

Figure 4.1 Minesweeper, without feedforward

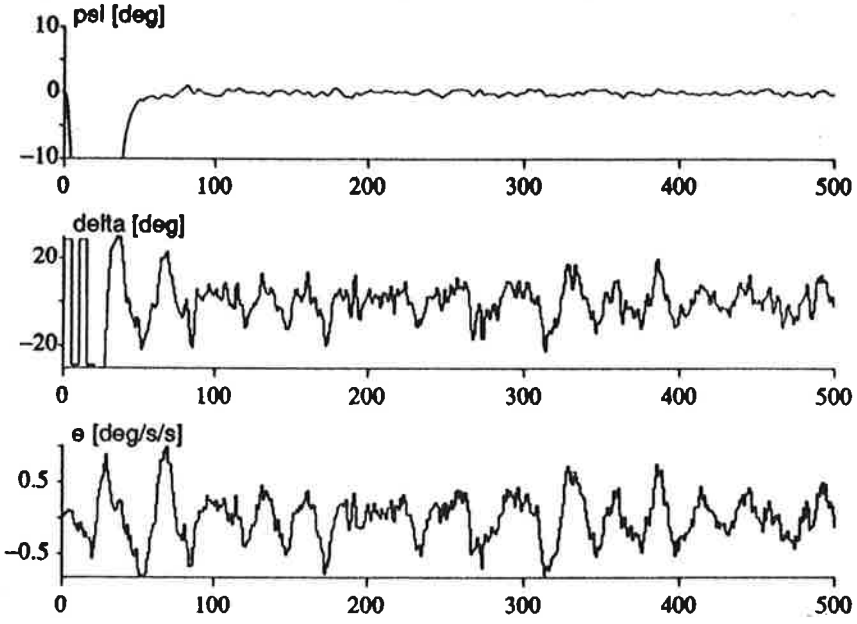

Figure 4.2 Minesweeper, with feedforward

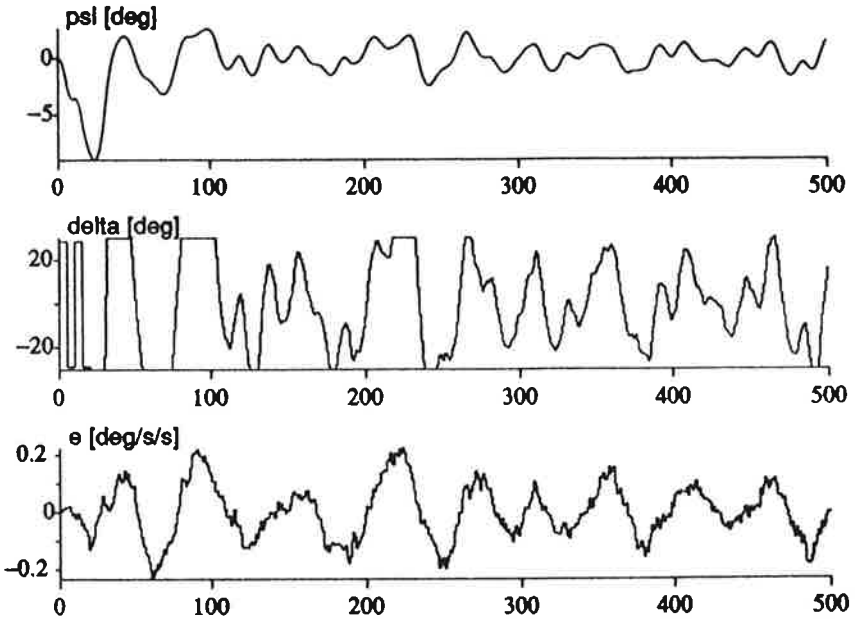

Figure 4.3 Cargo ship, without feedforward

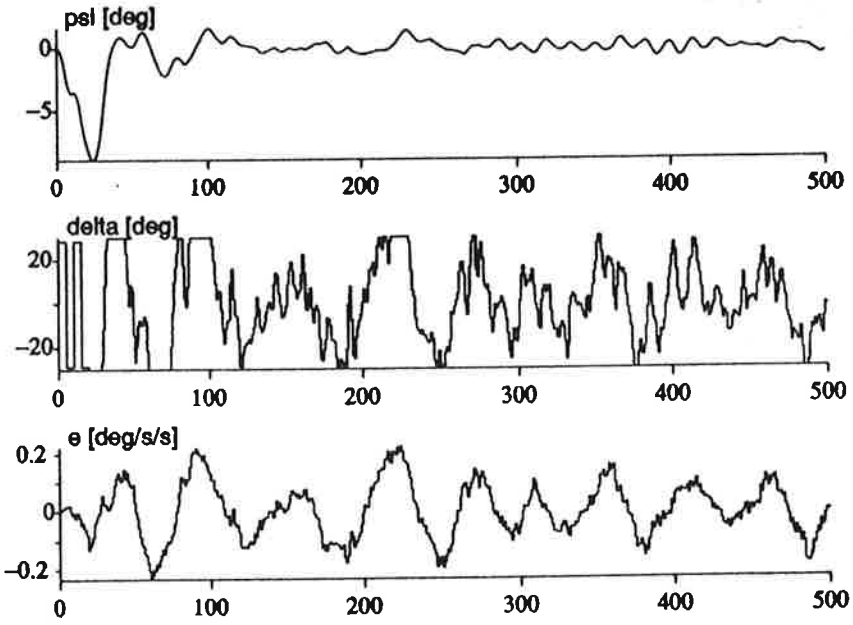

Figure 4.4 Cargo ship, with feedforward

5. Conclusions

Design of an adaptive autopilot for ship steering is discussed in this report. The autopilot is designed as an indirect adaptive regulator based on pole placement with feedforward compensation for wave disturbances. Our approach is different from the usual way of treating this kind of problem. It is customary to use a LQ-controller, since a lossfunction can be derived from physical properties. Severe simulations show that the basic self-tuning regulator is very robust and performs well for a wide range of operating conditions. The regulator's ability to compensate for wave disturbances is good. Performance is increased with feedforward, particularly for the minesweeper.

The experience with the simulation language SIMNON has been both positive and negative. The good properties are that differential- and difference equations simply can be written in this language, the ability to connect systems has also been valuable for the structure of our program. We are used to work with high level languages, so we find it remarkable that it is not possible to write a statement on several lines. We also miss a high level programming utility such as the simple if-then-else construction.

6. References

KÄLLSTRÖM, CLAES G. (1979): "Identification and Adaptive Control Applied to Ship Steering," PhD Thesis CODEN LUTFD2/(TFRT-1018), Departement of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ÅSTRÖM, K.J, AND B. WITTENMARK (1984): *Computer Controlled Systems*, Prentice Hall.

ÅSTRÖM, K.J, AND B. WITTENMARK (1988): *Adaptive Control*, Addison-Wesley.

Appendix SIMNON program

```
MACRO MacShip

let n.noise1 = 3
let nodd.noise1 = 13191

let n1.delay = 0
let n2.delay = 2
let space.delay = 7000

SYST delay noise1 Wave Ship STR Con

STORE degpsi[Ship] degdel[Ship] degm[STR]

SIMU 0 500 0.1
SPLIT 3 1
ASHOW degpsi
TEXT 'psi [deg]'
ASHOW degdel
TEXT 'delta [deg]'
ASHOW degm
TEXT 'e [deg/s/s]'

par dt: 1
par stdev1[noise1] : 1 "Wave disturbance
par stdev2[noise1] : 1 "Measurement noise of wave
par stdev3[noise1] : 1 "Measurement noise
END
```

```
CONNECTING SYSTEM Con
TIME t

td1[delay]  = t-tau       " Delay control signal
u1[delay]   = delta[STR]  " delta(t-tau)
delta[Ship] = y1[delay]   "

e[Wave]     = e1[noise1]

mpred[STR] = m[Wave]/wscale+e2[noise1]/nscale

td2[delay]  = t-tau       " Delay wave
u2[delay]   = mpred[STR]  " disturbance
m[STR]      = y2[delay]   "

m[Ship]     = m[STR]

deltac[STR] = 0  " Rudder command signal

psi[STR]    = y[Ship]+e3[noise1]/1000

omega[STR]  = omega " Frequency of
omega[Wave] = omega " wave disturbance

tau   : 0.5
h     : 1
omega : 0.2

wscale : 3000
nscale : 6000
END
```

```
CONTINUOUS SYSTEM Ship
"Model of ship steering dynamics
"
"
"
" d [ r(t) ] [-alfa  0][ r(t) ] [K  C][delta(t-tau)]
"---[      ]=[        ][      ]+[    ][            ]
"dt [psi(t)] [  1    0][psi(t)] [0  0][   m(t)     ]
"
" psi          = heading angle
" r            = turning rate
" delta        = rudder angle
" m            = disturbance torque from waves
" K, alpha, C = parameters specific for different ships
" tau          = time delay
"
INPUT delta m
OUTPUT y
STATE r psi
DER dr dpsi

dr   = -alpha*r+K*delta+ C*m
dpsi = r

y= psi
degpsi = 180*psi/pi
degdel = 180*delta/pi

alpha = u*a/l
K = u*u*b/(l*l)

pi = 3.1415926536

l : 55       "Ship length [m]
u : 10       "Ship velocity [m/s]

a : -0.14
b : -1.4
c : 1

END
```

```
CONTINUOUS SYSTEM Wave
" Waves modelled as white noise filtered through
"
"
"                      s
" G(s) =  ----------------------------
"           2                         2
"          s + 2*zeta*omega*s + omega
"
"
"
INPUT e omega
OUTPUT m
STATE m1 m2
DER dm1 dm2

dm1 = -2*zeta*omega*m1+m2+e
dm2 = -omega*omega*m1
m=m1

zeta : 0.1

END
```

```
DISCRETE SYSTEM STR
"An indirect self tuning regulator based on pole placement
"with feedforward compensation for wave disturbances.
"A time switch makes it possible to switch between two
"different estimation algorithms, the standard recursive
"least squares method and a method with conditional
"updating.
"
"Input and output parameters
INPUT psi deltac m mpred omega

OUTPUT delta

"State and New Declarations
"Symmetrical P-matrix, covariances for parameters
STATE p11 p12 p13 p14 p15 p16 p22 p23 p24 p25 p26
STATE p33 p34 p35 p36 p44 p45 p46 p55 p56 p66
NEW n11 n12 n13 n14 n15 n16 n22 n23 n24 n25 n26
NEW n33 n34 n35 n36 n44 n45 n46 n55 n56 n66

"Temporary P-matrix
STATE tempp11 tempp12 tempp13 tempp14 tempp15 tempp16
STATE tempp22 tempp23 tempp24 tempp25 tempp26 tempp33
STATE tempp34 tempp35 tempp36 tempp44 tempp45 tempp46
STATE tempp55 tempp56 tempp66
NEW tempn11 tempn12 tempn13 tempn14 tempn15 tempn16
NEW tempn22 tempn23 tempn24 tempn25 tempn26 tempn33
NEW tempn34 tempn35 tempn36 tempn44 tempn45 tempn46
NEW tempn55 tempn56 tempn66

"Phi vector, regression vector
STATE f1 f2 f3 f4 f5 f6
NEW nf1 nf2 nf3 nf4 nf5 nf6

"Theta vector, parameter vector
STATE th1 th2 th3 th4 th5 th6
NEW nth1 nth2 nth3 nth4 nth5 nth6

"Filter variables
STATE psi1 psi2 psi3 delta1 delta2 delta3 m1 m2 m3
NEW npsi1 npsi2 npsi3 ndelta1 ndelta2 ndelta3 nm1 nm2 nm3
STATE v1 v2 v3 deltac1 deltac2 deltac3
NEW nv1 nv2 nv3 ndeltac1 ndeltac2 ndeltac3

"Time and Sample Declarations
TIME t
TSAMP ts

"Start of estimation
```

```
"Residual epsilon
e= psi-psi1-th1*f1-th2*f2-th3*f3-th4*f4-th5*f5-th6*f6

"K (Column Vector)= P*Phi
k1= p11*f1+p12*f2+p13*f3+p14*f4+p15*f5+p16*f6
k2= p12*f1+p22*f2+p23*f3+p24*f4+p25*f5+p26*f6
k3= p13*f1+p23*f2+p33*f3+p34*f4+p35*f5+p36*f6
k4= p14*f1+p24*f2+p34*f3+p44*f4+p45*f5+p46*f6
k5= p15*f1+p25*f2+p35*f3+p45*f4+p55*f5+p56*f6
k6= p16*f1+p26*f2+p36*f3+p46*f4+p56*f5+p66*f6

"Denominator (Scalar)= 1+Phi*P*Phi+cbar*Phi*Phi, cbar >= 0
tempff= f1*f1+f2*f2+f3*f3+f4*f4+f5*f5+f6*f6
d= 1+f1*k1+f2*k2+f3*k3+f4*k4+f5*k5+f6*k6+cbar*tempff

"Update Theta values (Row vector)
nth1= th1+xa*k1*e/d
nth2= th2+xa*k2*e/d
nth3= th3+xa*k3*e/d
nth4= th4+xa*k4*e/d
nth5= th5+xa*k5*e/d
nth6= th6+xa*k6*e/d

"Update temporary P-matrix, xa= 0 when residual < ediff
tempn11= (tempp11-xa*k1*k1/d)
tempn12= (tempp12-xa*k1*k2/d)
tempn13= (tempp13-xa*k1*k3/d)
tempn14= (tempp14-xa*k1*k4/d)
tempn15= (tempp15-xa*k1*k5/d)
tempn16= (tempp16-xa*k1*k6/d)
tempn22= (tempp22-xa*k2*k2/d)
tempn23= (tempp23-xa*k2*k3/d)
tempn24= (tempp24-xa*k2*k4/d)
tempn25= (tempp25-xa*k2*k5/d)
tempn26= (tempp26-xa*k2*k6/d)
tempn33= (tempp33-xa*k3*k3/d)
tempn34= (tempp34-xa*k3*k4/d)
tempn35= (tempp35-xa*k3*k5/d)
tempn36= (tempp36-xa*k3*k6/d)
tempn44= (tempp44-xa*k4*k4/d)
tempn45= (tempp45-xa*k4*k5/d)
tempn46= (tempp46-xa*k4*k6/d)
tempn55= (tempp55-xa*k5*k5/d)
tempn56= (tempp56-xa*k5*k6/d)
tempn66= (tempp66-xa*k6*k6/d)

"Update P-matrix, xc1 > 0
trP= tempn11+tempn22+tempn33+tempn44+tempn55+tempn66
n11= if t>chtime then xc1*tempn11/trP+xc2 else tempn11
n12= if t>chtime then xc1*tempn12/trP else tempn12
n13= if t>chtime then xc1*tempn13/trP else tempn13
```

```
n14= if t>chtime then xc1*tempn14/trP else tempn14
n15= if t>chtime then xc1*tempn15/trP else tempn15
n16= if t>chtime then xc1*tempn16/trP else tempn16
n22= if t>chtime then xc1*tempn22/trP+xc2 else tempn22
n23= if t>chtime then xc1*tempn23/trP else tempn23
n24= if t>chtime then xc1*tempn24/trP else tempn24
n25= if t>chtime then xc1*tempn25/trP else tempn25
n26= if t>chtime then xc1*tempn26/trP else tempn26
n33= if t>chtime then xc1*tempn33/trP+xc2 else tempn33
n34= if t>chtime then xc1*tempn34/trP else tempn34
n35= if t>chtime then xc1*tempn35/trP else tempn35
n36= if t>chtime then xc1*tempn36/trP else  tempn36
n44= if t>chtime then xc1*tempn44/trP+xc2 else tempn44
n45= if t>chtime then xc1*tempn45/trP else tempn45
n46= if t>chtime then xc1*tempn46/trP else tempn46
n55= if t>chtime then xc1*tempn55/trP+xc2 else tempn55
n56= if t>chtime then xc1*tempn56/trP else tempn56
n66= if t>chtime then xc1*tempn66/trP+xc2 else tempn66

"Update filter variables
nv1 = v
nv2 = v1
nv3 = v2

npsi1= psi
npsi2= psi1
npsi3= psi2

ndelta1= delta
ndelta2= delta1
ndelta3= delta2

ndeltac1 = deltac
ndeltac2 = deltac1
ndeltac3 = deltac2

nm1= m
nm2= m1
nm3= m2


"Update Phi vector
nf1= psi-psi1
nf2= delta
nf3= f2
nf4= f3
nf5= m
nf6= f5

"Discrete time model polynomial
slaskm = sqrt(1-zetam*zetam)*omegam*h
```

```
slaskm1 = -exp(-alpham*omegam*h)
slaskm2 = -2*exp(-zetam*omegam*h)*cos(slaskm)
slaskm3 = exp(-2*zetam*omegam*h)
am1 = slaskm1+slaskm2
am2 = slaskm1*slaskm2+slaskm3
am3 = slaskm1*slaskm3

"Discrete time observer polynomial
slasko = sqrt(1-zetao*zetao)*omegao*h
slasko1 = -exp(-alphao*omegao*h)
slasko2 = -2*exp(-zetao*omegao*h)*cos(slasko)
slasko3 = exp(-2*zetao*omegao*h)
ao1 = slasko1+slasko2
ao2 = slasko1*slasko2+slasko3
ao3 = slasko1*slasko3

"Process parameters
a  = th1
b1 = th2
b2 = th3
b3 = th4
c1 = th5
c2 = th6

"Solve Diophantine equation
AoEvala = a*a*a+ao1*a*a+ao2*a+ao3
AoEval1 = 1+ao1+ao2+ao3
AmEvala = a*a*a+am1*a*a+am2*a+am3
AmEval1 = 1+am1+am2+am3
BEvala = b1*a*a+b2*a+b3
BEval1 = b1+b2+b3
slask0 = a*b2+a*(2+a)*b1
slask1 = ao2*am3+ao3*am2+a*(2+a)*(2+a)
slask2 = a*(2+a)*(ao1+am1)+(am2+ao1*am1+ao2)*a
slask3 = slask1+slask2-a*(1+2*a)
s3 = ao3*am3/b3
slask4 = slask3-b2*s3-slask0*AmEval1*AoEval1/BEval1
slask5 = slask4+slask0*s3
slask6 = a*a*a*AmEval1*AoEval1/BEval1
slask7 = slask6-a*a*a*s3+s3
slask8 = slask7-AmEvala*AoEvala/BEvala
slask9 = slask5/(a*b1-slask0)-slask8/(a*a*(a-1))
slask10 = (b3-slask0)/(a*b1-slask0)-(a+1)/a
s2 = slask9/slask10
s1 = slask5/(a*b1-slask0)-(b3-slask0)/(a*b1-slask0)*s2
s0 = AmEval1*AoEval1/BEval1-s1-s2-s3
r0 = 2+a+ao1+am1-b1*s0
r1 = (b2*s3+b3*s2-ao2*am3-ao3*am2)/a
slask11 = AmEval1/BEval1
t0 = slask11
t1 = slask11*ao1
```

```
t2 = slask11*ao2
t3 = slask11*ao3

"Calculate Feedforward compensation term
ReC = c1*cos(2*omega)+c2*cos(omega)
ImC = c1*sin(2*omega)+c2*sin(omega)
ReB = b1*cos(2*omega)+b2*cos(omega)+b3
ImB = b1*sin(2*omega)+b2*sin(omega)
ReR = cos(3*omega)+(r0-1)*cos(2*omega)+(r1-r0)*cos(omega)-r1
ImR = sin(3*omega)+(r0-1)*sin(2*omega)+(r1-r0)*sin(omega)
ReHL = -ReR*ReC+ImR*ImC
ImHL = -ReR*ImC-ReC*ImR
FFslask1 = ReB*cos(3*omega)-ImB*sin(3*omega)
FFslask2 = ReB*cos(2*omega)-ImB*sin(2*omega)
FFslask3 = ImB*cos(3*omega)+ReB*sin(3*omega)
FFslask4 = ImB*cos(2*omega)+ReB*sin(2*omega)
FFslask5 = ReHL*FFslask3-FFslask1*ImHL
FFslask6 = FFslask2*FFslask3-FFslask1*FFslask4
ff1 = FFslask5/FFslask6
ff0 = (ReHL-FFslask2*ff1)/FFslask1

"Compute control signal with anti-windup
aor1 = ao1-r0+1
aor2 = ao2-r1+r0
aor3 = ao3+r1
AoRdelta = aor1*delta1+aor2*delta2+aor3*delta3
Tdeltac  = t0*deltac+t1*deltac1+t2*deltac2+t3*deltac3
FFmpred  = ff0*mpred+ff1*(m1+tau*(m-m1)/h)
Spsi     = s0*psi+s1*psi1+s2*psi2+s3*psi3
v = -ao1*v1-ao2*v2-ao3*v3+Tdeltac+FFmpred+AoRdelta-Spsi

d1 = if mod(t,10)<5 then 0.5 else -0.5  " Generate square
y  = if t<20 then d1 else v             " wave

delta=if y<dlow then dlow else if y<dhigh then y else dhigh


"Update sampling time
ts= t+h

"Initial Values and Constants

h: 1       "Sampling time
tau : 0.5  "Time delay in control signal

"Parameters for dead zone
"estimation algorithm
ediff: 0.001
slaskxa = if abs(e) > ediff then abar else 0
xa= if t<chtime then 1 else slaskxa
slaskxc1 = if tempff<0.000001 then 1 else 100/tempff
```

```
xc1= if t<chtime then trP else slaskxc1
xc2=0.001
cbar: 0
abar: 0.1
chtime : 250

degdc = 180*deltac/3.1415926536 "Command and wave in degrees
degm  = 180*m/3.1415926536      "

dlow : -0.5236   " Corresponds to a maximum/minimum
dhigh : 0.5236   " rudder angle of +-30 degrees

zetam  : 1     " Continuous time model
omegam : 0.5   " specifications
alpham : 0.5

zetao  : 1     " Continouos time observer
omegao : 1     " specifications
alphao : 1

th1: 0.5
th2: 0.5
th3: 0.5
th4: 0.5
th5: 0.5
th6: 0.5

tempp11: 100
tempp22: 100
tempp33: 100
tempp44: 100
tempp55: 100
tempp66: 100

END
```