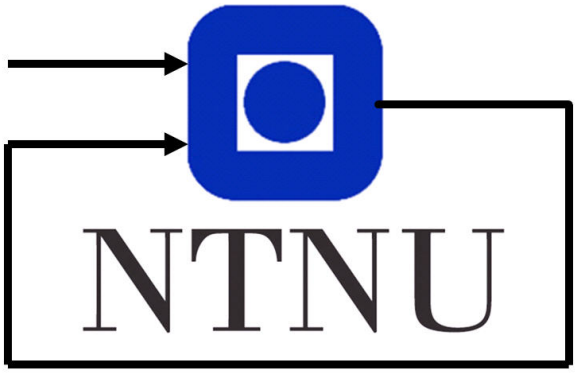
Assignment 3 in TTK4190
Guidance and Control of Vehicles

Group 17
Alexandra S. Vedeler
Ella-Lovise H. Rørvik
Marie B. Henriksen
Kjetil S. Gjerden

Fall 2017



Department of Engineering Cybernetics

Abstract

This is our answers and results for Assignment 4. This is a results and discussion text, and not as official as a report would be in presentation, standards and conventions. This text is more about the fact that you as a reader will be able to understand what we have done, what our results were and what we have been able to conclude from them.

Contents

Heading Autopilot

1.1: Model of Choice

There are several alternatives to consider when choosing a heading controller for a ship. Initially, one could recognise that a ship of this type is generally nonlinear. One might then be tempted to directly go for a nonlinear model, e.g. a nonlinear extension of the Nomoto model as described in [3]. Instead, the ship dynamics were analysed and compared to linear heading models, as linear models are not as computationally complex as nonlinear ones.

One linear alternative is the 1st order Nomoto model, defined in the Laplace domain as

$$\begin{aligned} \frac{r}{\delta}(s) &= \frac{K}{1+Ts} \\ \dot{\psi} &= r\,, \end{aligned} \tag{1}$$

where K and T is the gain and time constant, respectively, and δ is the rudder angle. The 1st order Nomoto model is valid under the assumption that the ship operates with small rudder angles, at low velocity and only in the XY-plane. This model is widely used for its simplicity and relative accuracy. The comparison between the ship response and the 1st order Nomoto model is shown in fig. 1. From this we see that the 1st order model fails to follow the ship dynamics accurately in the short time right after the rudder is changed.

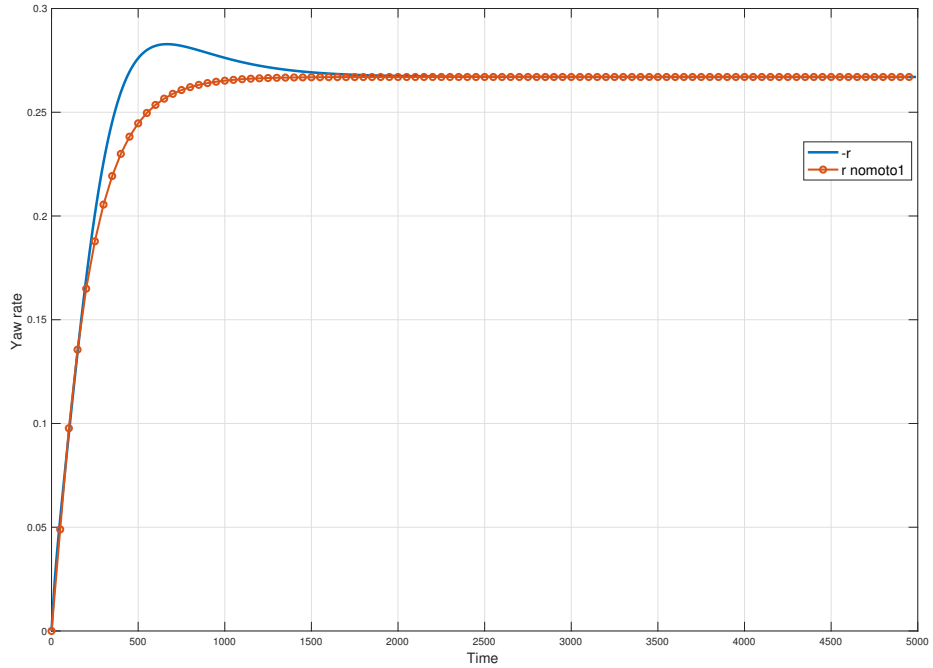


Figure 1: 1st order Nomoto model response compared to ship dynamics with a rudder step of 3°.

Seeing as the 1st order Nomoto model fails to accurately replicate the ship transient dynamics, one can instead try the 2nd order Nomoto model, defined as

$$\begin{aligned} \frac{r}{\delta}(s) &= \frac{K(1+T_3 s)}{(1+T_1 s)(1+T_2 s)} \\ \dot{\psi} &= r\,, \end{aligned} \tag{2}$$

where T_1, T_2, T_3 are constants and K is the gain. The comparison between the 1st and 2nd order Nomoto model and the ship response is shown in fig. 2. From this it is clear that the 2nd order model more accurately captures the characteristics of the nonlinear ship dynamics, due to the extra zero and pole on the system, imitating the sway coupling effect on yaw rate. The 2nd order model very accurately replicates the actual ship dynamics [1], covering even the complications arising right after applying a rudder change (sway-yaw rate coupling), and is thus a good alternative. This, combined with the relative ease of which the parameters can be estimated (e.g. using `MATLAB`), results in an overall good model for controlling the ship heading. Seeing as the linear models together cover the necessary dynamics, no nonlinear models will be considered. The 2nd order Nomoto is our model of choice, and will be used going forward.

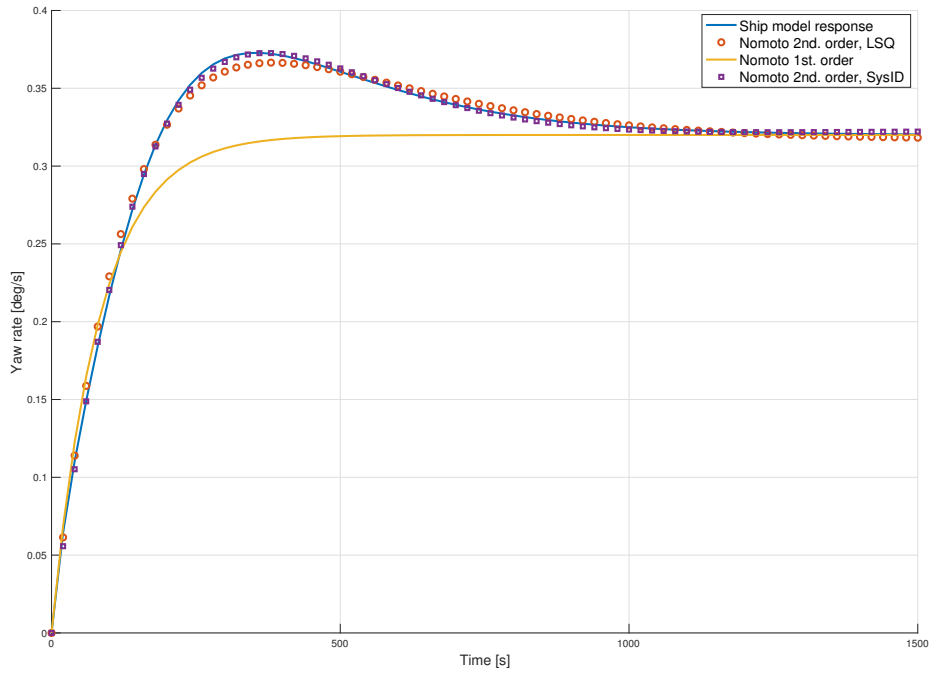


Figure 2: 1st. and 2nd order Nomoto responses compared to the ship dynamics with a rudder step of 8°. The 2nd order parameters were estimated using least squares (○) and system identification (□) from the `Matlab` system identification toolbox.

[1] The 1st order model, when estimated using the system identification toolbox in `Matlab`, had roughly a 60% coverage of the actual ship response, whereas the 2nd order model with estimated parameters covered approximately 97% of the ship response.

1.2: Identification of Model Parameters

Selecting the 2nd order Nomoto model means having to estimate the constants T_1, T_2, T_3 and K. This was done using both a least squares curve fitting function (`lsqcurvefit`) and a system identification-based method. The least squares method yielded the results shown in fig. 3 for different rudder inputs. The overall performance increased, however, when basing the parameter estimation on the system identification toolbox in MATLAB. This method resulted in a coverage of roughly 97% compared to the ship model dynamics. The plot of this result is shown in fig. 2. The estimated parameters from the system identification approach was the ones used for the heading model, since the system identification approach resulted in a slightly better model fit, both regarding the transient and the steady-state response (fig. 2). Since the parameters change depending on the input, as presented in fig. 3, the transfer function for one given rudder input ($\delta_c = 8^\circ$, $n_c = 7.3$) is shown in eq. (3). $\delta_c = 8^\circ$ was exemplified as it is well within the area of operation for both 1st and 2nd order heading models while still keeping away from the saturation limits.

$$\frac{r}{\delta}(s) = \frac{-3.793 \cdot 10^{-4} s - 1.294 \cdot 10^{-6}}{s^2 + 9.606 \cdot 10^{-3} s + 3.237 \cdot 10^{-5}} \tag{3}$$

1.3: Designing the Controller

A simple PID controller was initially chosen to control the ship heading. Defining the error term as $\tilde{\psi} = \psi_d - \psi$, in contrast to the assignment text, and keeping in mind that $\dot{\psi} = r$, the control law becomes

$$\tau = K_p \tilde{\psi} + K_i \int \tilde{\psi} \, d\tilde{\psi} + K_d \tilde{r} \ .$$

To ensure that the stated controller produces the wanted system convergence to zero, i.e.

$$\lim_{t \to \infty} \tilde{\psi}(t) = 0 \ ,$$

the closed-loop system can be analysed in the Laplace domain. The closed-loop transfer function becomes

$$\frac{\psi}{\psi_d}(s) = \frac{K(1 + T_3 s)(K_p s + K_i + K_d s^2)}{s^2(1 + T_1 s)(1 + T_2 s) + K(1 + T_3 s)(K_p s + K_i + K_d s^2)}. \tag{4}$$

From the closed-loop transfer function, it can be verified that the poles all lie in the left half plane for appropriately chosen controller parameters, indicating an asymptotically stable system (as the Nomoto model is a linear model). Thus, since the system is GAS, the PID controller will ensure zero error as $t \to \infty$.

Furthermore, defining the deviation ratio, $N(s)$, as the transfer function from desired value to state error

$$N(s) = \frac{\tilde{\psi}}{\psi_d}(s) = \frac{(1 + T_1 s)(1 + T_2 s)s}{(1 + T_1 s)(1 + T_2 s)s + K(T_3 s + 1)(K_p s + K_i + K_d s^2)} \ ,$$

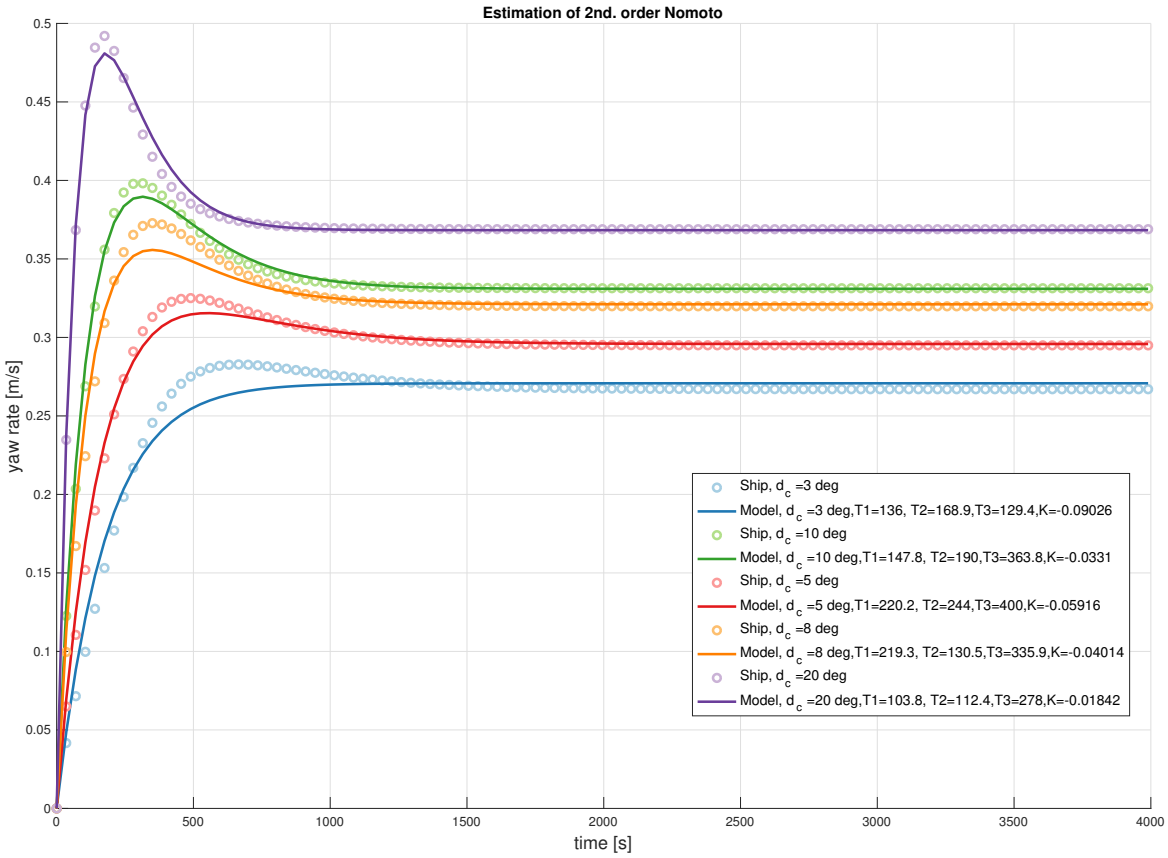


Figure 3: Response of second order Nomoto model with LSQ-estimated parameters compared to the ship model response for different rudder inputs.

it can be verified from the final value theorem that

$$\lim_{s \to 0} s\tilde{\psi}(s) = 0$$

for both impulse and step inputs. Thus the PID controller ensures that the error converges to zero.

After implementing the guidance system discussed in later sections, it became evident that a simpler PD controller yielded a better performance. Thus the control law becomes

$$\tau = K_p \tilde{\psi} + K_d \tilde{r} \ .$$

This results in an overall faster control response compared to a PID. This might indicate that the Nomoto model fails to capture essential parts of the ship model dynamics, as the closed loop transfer function argues that a PD controller would fail to ensure convergence. Some of the discrepancies between the model and the ship dynamics can of course be attributed to the assumptions done by the Nomoto model, simplifying the system.

Looking at the disturbances in the heading control system one can argue that there are none. We do have the presence of current, but with the assumption that this is homogeneous along the body of the ship and that the ship is in general symmetric, this would not provide any momentum to the ship as it would cancel itself out. It will however impact the velocity control of the ship and thus calculation of a reference heading, as will be discussed later in this report. One could argue that model errors might be considered disturbances, but they have not been taken them into account in this simulation.

All in all, since the steady-state error decreased when omitting the integral part of the controller, this seems to indicate that the PD controller is a better solution. This is further backed by Perez (2005)[5] / Fossen (2002)[4].

1.4: Tuning the Controller

When tuning the controller, the PID and acceleration feedback pole-placement algorithm from Table 12.2 in [3] was used to obtain some starting values for the controller gains. The bandwidth of the system was specified to be $\omega_b = 0.125$, approximately three times the crossover frequency w_c from the closed loop transfer function. The relative damping ratio was chosen as $\zeta = 1$, the acceleration feedback gain set to $K_m = 0$ and as we are tuning a PD controller, K_i was set to zero. The gains of the controller were then computed to be

$$K_p = \frac{T}{K}\omega_n^2 = 8.6786$$
$$K_d = 2\zeta\omega_n\frac{T}{K} - \frac{1}{K} = 195.3422,$$

where

$$\omega_n = \frac{\omega_b}{\sqrt{1 - 2\zeta^2 + \sqrt{4\zeta^4 - 4\zeta^2 + 2}}} = 0.1051.$$

T and K were found from using the assumption $T = T_1 + T_2 - T_3$ with the estimated values $T_1 = 147.8$, $T_2 = 190$ and $T_3 = 363.8$, as well as $K = -0.0331$, from the 2nd order Nomoto model shown in fig. 3, when $d_c = 10$. Using the model for $d_c = 10$ was done simply because we expect a rudder angle between $\pm 25°$ and the model for $d_c = 10$ looked nice.

The reference signal was shifted to $\psi_d(t) = -0.3\sin(0.008t)$ rad with $r_d(t) = \dot{\psi}(t)$, and the initial conditions were set to $u(0) = 6.63$ m/s, $v(0) = \psi(0) = r(0) = 0$ and $n_c = 7.3$ rad/s. The current was switched on. The system was then simulated, and the gains manually tuned until the system showed satisfactory behaviour, shown in fig. 4. The final gains used throughout the assignment was found to be

$$K_p = 6$$
$$K_d = 250.$$

From fig. 4 it can be seen that the simulated signal follows the desired signal both for *psi* and $\dot{\psi}$ pretty well. By turning up the gains the error in ψ would be even less, but this would affect the error in $\dot{\psi}$ and make the error greater, specially during the first 100 s. The final gains were therefore chosen as a compromise, to make both errors as small as possible.

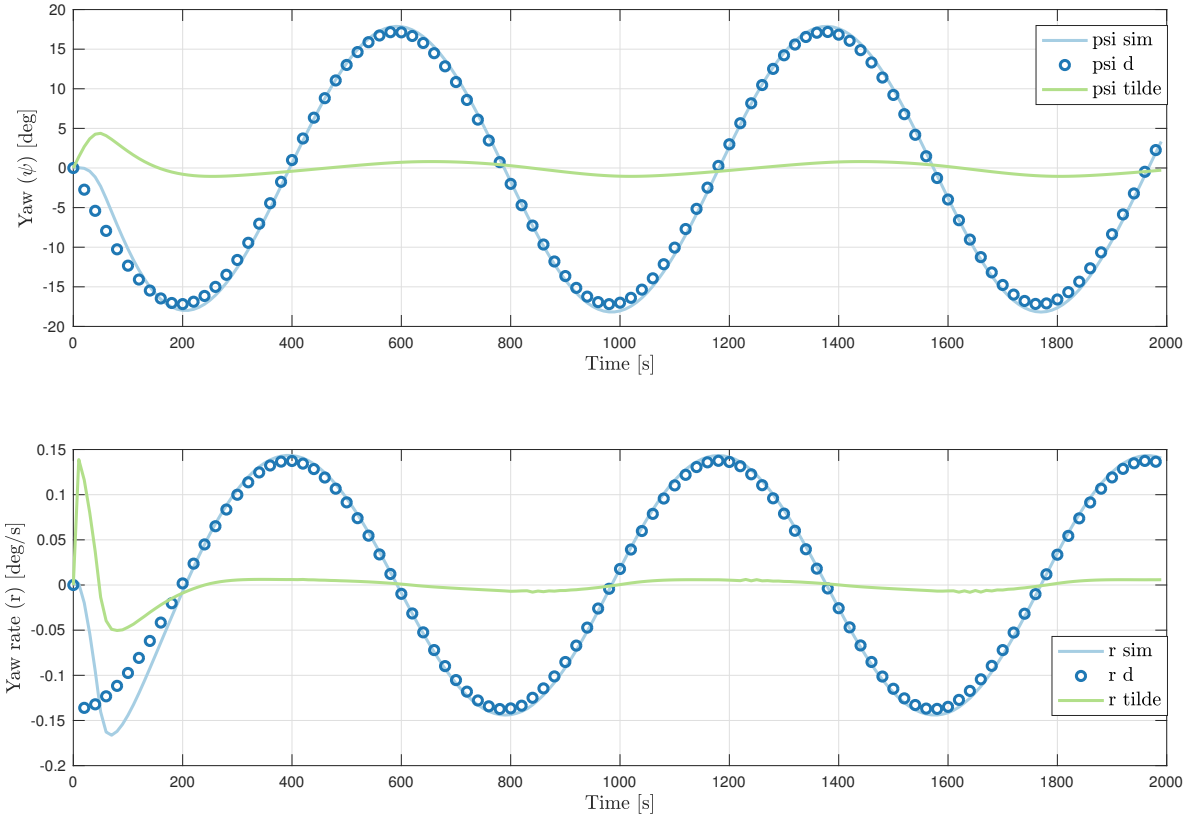


Figure 4: Tracking of a time-varying sinusoidal desired heading ($\psi_d(t)$) and heading rate ($r_d(t)$) with $K_p = 6$ and $K_d = 250$.

The rudder angle when tracking the sinusoidal desired heading reference signal can be seen in fig. 5. Before manually tuning the controller gains, the rudder angle would oscillate in the turning when following the sinusoidal reference signal. After manually tuning these oscillations disappeared, which proves that the final gains are better for the ship than the ones found using the PID and acceleration feedback pole-placement algorithm in our case.

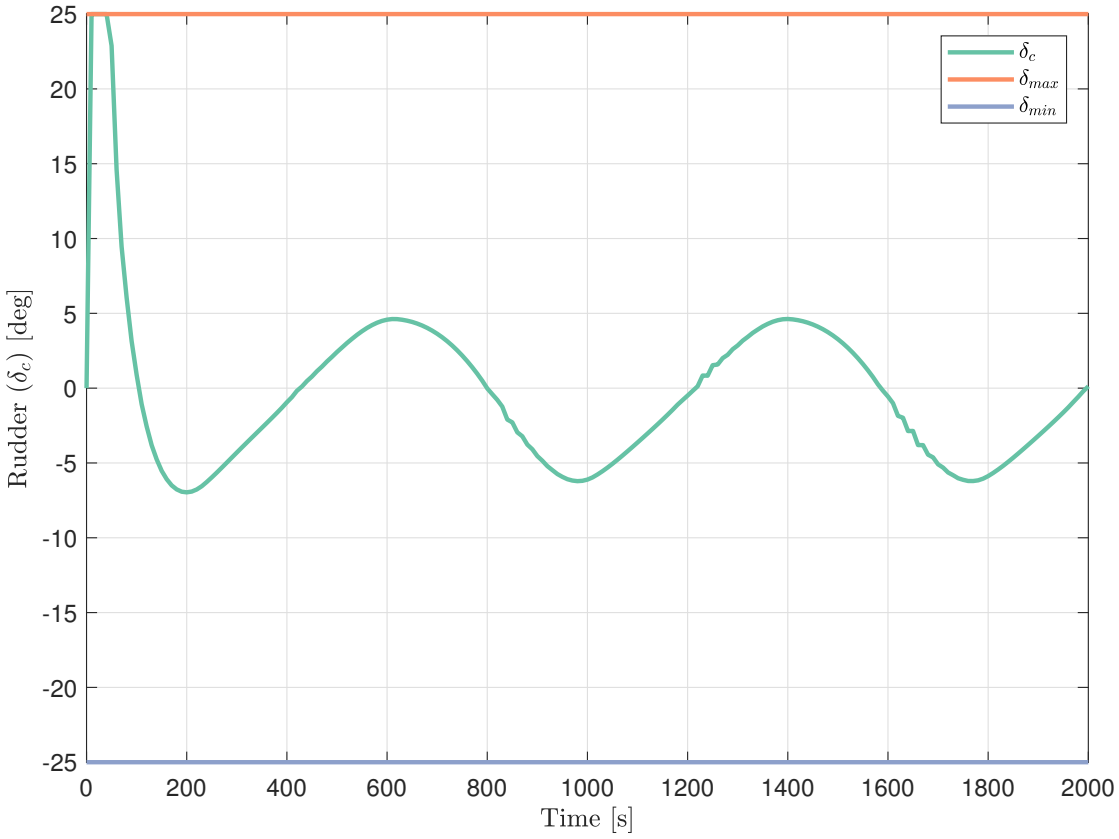


Figure 5: The rudder angle when tracking a time-varying sinusoidal desired heading ($\psi_d(t)$) and heading rate ($r_d(t)$) with $K_p = 6$ and $K_d = 250$.

Speed Autopilot

1.5: Model of Choice

The 3 DOF horizontal plane models for maneuvering are based on the rigid body kinetics.

$$\mathbf{M}_{RB}\dot{\nu} + \mathbf{C}_{RB}\nu = \boldsymbol{\tau}_{hyd} + \boldsymbol{\tau}_{hs} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave} + \boldsymbol{\tau} \tag{5}$$

With hydrostatic forces $\boldsymbol{\tau}_{hs} = 0$.

In 3 DOF manoeuvring the dynamics associated with motion heave, roll and pitch are neglected, meaning $w = p = q = 0$. It is also assumed that the craft has homogenous mass distribution and xz-plane symmetry such that $\mathbf{I}_{xy} = \mathbf{I}_{yz} = \mathbf{0}$.

The classical maneuvering model makes use of the zero-frequency models for surge, sway and yaw, stating that the horizontal motions (surge, sway and yaw) of a marine craft moving at forward speed can be described using a zero-frequency model where $\mathbf{M}_A$ and $\mathbf{D}_p$ are constant

matrices. This technique is frequently applied when deriving maneuvering models for ships in a seaway. Using equations of motions not depending on frequency reduces the models complexity. A normal feedback control system stabilized in surge, sway and yaw have natural periods in the range of 100 - 200 s, meaning natural frequency of 0.03 - 0.10 rad/s. ([3], p. 113).

If we let the body-frame coordinate origin be set at the centerline of the craft at the point CO, then $y_g = 0$ The added mass matrix is assumed to be computed in CO. This reduces the kinetic equation.

$$\mathbf{M}_A = \begin{bmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & A_{26} \\ 0 & A_{62} & A_{66} \end{bmatrix} \mathbf{D}_p = \mathbf{0} \tag{6}$$

with $A_{11} = m - X_{\dot{u}}$. An example of nonlinear maneuvering model based on surge resistance and cross flow drag is:

$$\mathbf{N}(\nu_r)\nu_r = \mathbf{C}_A(\nu_r)\nu_r + \mathbf{D}\nu_r + \mathbf{d}(\nu_r) \tag{7}$$

This model consists of the linear damper $\mathbf{D}$, which is important for low-speed maneuvering and station keeping, while $\mathbf{d}(\nu_r)$ is more dominant at higher speeds. ([3], p. 136). Linear damping will always be present for marine craft operating in waves due to potential damping and linear skin friction.

For a marine craft moving at slowly varying forward speed

$$U = \sqrt{u^2 + v^2} \approx u \tag{8}$$

The 3 DOF maneuvering model can be decoupled in a forward speed (surge) model and a sway-yaw subsystem for manoeuvring.The surge equation, equation (7.32) in [3] is:

$$(m - X_{\dot{u}})\dot{u} - X_u u_r - X_{|u|u}|u_r|u_r = \tau_1 \tag{9}$$

with τ_1 as the sum of the control and external forces in surge. As stated before, the effects of the nonlinear damping are more important at higher speeds, meaning that the nonlinear term at lower speeds may be approximated with a linear term. Approximated with a linear term, the transfer function from τ_1 to u is of first order, meaning

$$a\dot{u} + bu = \tau_1 \tag{10}$$

with the constants a and b, and no current.

The relationship between the control forces and the velocity of the shaft, is not stated in the assignment. Consequently it was attempted to use a linear model for the relationship between the control force and the shaft velocity, meaning

$$\tau_1 = k_{nc} n_c \tag{11}$$

with the constant k_{nc}. The summarised linear model of the surge, consisting of surge and n_c, is:

$$a\dot{u} + bu = k_{nc}n_c \tag{12}$$

The conclusion is that with a linear model of the system the modelling is less complex, but also associated with more assumptions.

1.6: Identification of Model Parameters

The parameters may be found using a curve fitting tool in `MATLAB`. In sections 1.5 it was suggested to find a linear model of the surge dynamics, decoupled from the dynamics in yaw and sway. To find the parameters of the linear model it is therefore necessary to limit changes in the sway/heading dynamics. Therefore a controller on yaw is used during the identification of the system parameters, setting $\psi_d = 0$. The current is also turned off, and we assume that the only force on the system are therefore the controller input n_c.

With the `MATLAB`-app system identification, a transfer function of the dynamics in surge may be found. The functionality of the app that finds the transfer function is also manifested as a `MATLAB`-function `tfest(data,numberOfPoles)`. This function may find a linear transfer function for a given data set, for a given number of poles and zeros. The first order model will be on form shown in eq. (13), and the second order model will be on form shown in eq. (14).

$$H_1(s) = \frac{a_1}{b_1 s + c} \tag{13}$$

$$H_2(s) = \frac{a_2 s + b_2}{c_2 s^2 + d_2 s + e_2} \tag{14}$$

The parameters of the fitting are shown for the first order model in table 1 and for the second order model in table 2. The tables include the step made in n_c, the constants from equation (13) or (14) and the fitting percentage of the estimated model compared to the data from MS *Fartøystyring*.

Table 1: Parameters for fitting with transfer function with one pole, with u(0) = 6.63 m/s, from equation (13).

$\mathbf{n}_c$	$\mathbf{a}_1$	$\mathbf{b}_1$	$\mathbf{c}_1$	**Fit percentage**
2.0944 rad/s	-0.0001	1.0000	0.0002	57.2993
4.1888 rad/s	0.0016	1.0000	0.0016	98.2184
6.8068 rad/s	0.3120	1.0000	0.3180	20.7147
7.3304 rad/s	0.0163	1.0000	0.0167	29.9601
8.9012 rad/s	0.0028	1.0000	0.0029	99.1487

Looking at the tables, the fitting of the first order models ranges from $20\% - 99\%$, depending on n_c. Compared to the second order model the fitting ranges from $95.5\% - 99.5\%$, as shown in table 2. We also tried to fit the model with different initial conditions. Fitting with the initial condition $u(0) = 2$ instead of $u(0) = 6.63$ gave table 3.

Table 2: Parameters for fitting with transfer function with two poles,with u(0) = 6.63 m/s, from equation (14).

$\mathbf{n}_c$	$\mathbf{a}_2$	$\mathbf{b}_2$	$\mathbf{c}_2$	$\mathbf{d}_2$	$\mathbf{e}_2$	**Fit percentage**
2.0944 rad/s	0.0525	0.0001	1.0000	0.0673	0.0001	95.4870
4.1888 rad/s	0.0764	0.0002	1.0000	0.1019	0.0002	98.7201
6.8068 rad/s	0.0041	0.0000	1.0000	0.0065	0.0000	99.9905
8.9012 rad/s	0.1663	0.0005	1.0000	0.1821	0.0005	99.5198

Table 3: Parameters for fitting with transfer function with one pole with u(0) = 2.0 m/s, from equation (13).

$\mathbf{n}_c$	$\mathbf{a}_1$	$\mathbf{b}_1$	$\mathbf{c}_1$	**Fit percentage**
2.0944 rad/s	0.0992	1.0000	0.1018	7.5233
4.1888 rad/s	0.0012	1.0000	0.0013	98.1862
6.2832 rad/s	0.0018	1.0000	0.0018	97.4049
7.3304 rad/s	0.0021	1.0000	0.0021	97.1786
8.9012 rad/s	0.0025	1.0000	0.0025	96.9417

In table 3 the fitting ranges from $96.9417 - 98.1862\%$, with an outlier with a percentage of 7.5233%. The function `tfdata(sys)` returns the numerator(s) and denominator(s) of the transfer function for TF, SS or ZPK model of tye system. The accuracy of computations using high-order transfer functions is at times poor, particularly for MIMO or high-order systems [6]. Conversion to a transfer function representation can incur a loss of accuracy [2]. From table 1 and table 3 it seems that the fitting was less successful when the initial velocity is close to the systems steady state velocity with specified n_c. We therefore tried a different method using `MATLAB`-function `lsqurvefit` and the first order model in time domain:

$$u(t) = Kn_c - (Kn_c - u(0))e^{-t/T} \tag{15}$$

With T being the time constant of the system, and K being the system gain. In this fitting $u(0) = 6.63$. `lsqurvefit` is a nonlinear least-squares solver.

Approximated step response of the first order model is plotted together with the response from MS *Fartøystyring* in fig. 6.

The figures also show the estimated values from T and K found using `lsqurvefit`. From the figure it is apparent that the fitting of the first order model yields quite good estimates for $n_c >= 4$ rad/s, and also quite good fitting of the steady state for $n_c = 2$. The model does not fit well when $n_c = 0$. The reason for this may be that with $n_c = 0$ the change in u from $u(0)$ to steady state value is large. With larger steps in u it is harder for a first order model to model the transient response, as displayed in fig. 6.

The 1st order Nomoto model is valid under the assumption that the ship operates with small rudder angles, at low velocity and only in the XY-plane.

Even though the first order model fitted quite well with the response from MS *Fartøystyring*, we wanted to test the fitting with the forward speed model, from eq. (9). The fitting was

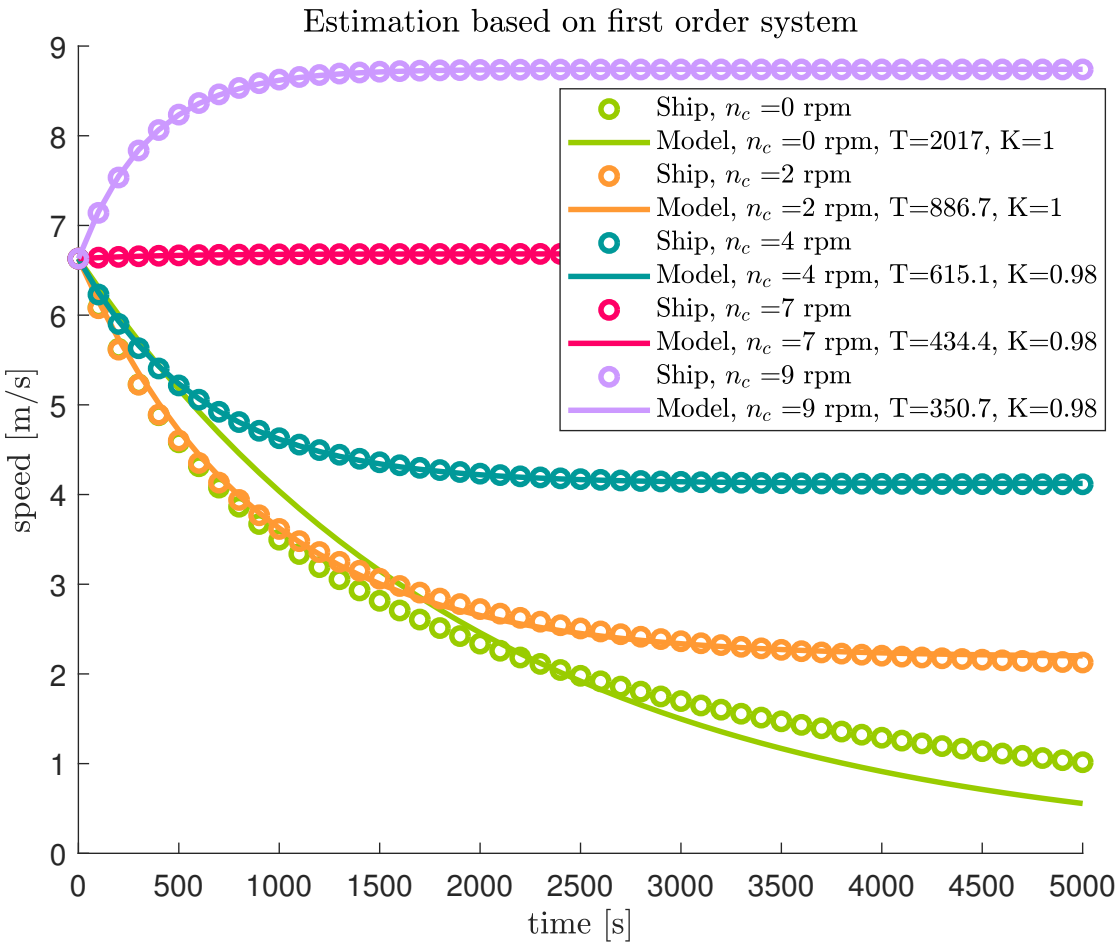


Figure 6: Compared results of the estimation of the first order model response vs. MS *Fartøystyring* response. The different responses are from different steps in n_c.

implemented using `lsqurvefit` and a simulink implementation of equation eq. (9). With $k_1 = (m - X_{\dot{u}})/k_{nc}$, $k_2 = X_u/k_{nc}$ and $k_3 = X_{|u|u}/k_{nc}$, from eq. (9) and eq. (11). The result of the curve fitting is shown in figure 7. The plots shows better estimation than the linear first order model for lower shaft speed n_c. This is because of the lack of complexity in the first order model, making it harder to model the transient behaviour as discussed before.

Still the first order model is a simple model which gives relatively good approximations with higher shaft velocities, and is therefore the model we choose as an estimation of the system. The model will not be used for dynamic positioning, meaning that a good model for lower speeds are not necessary in this case.

In figure 6 the T and K were found through `lsqurvefit`. The figures also shows that T varies from approximately 300 to 2000. In later assignments the region of operation for the ship is approximately $3 - 9$ m/s. Therefore to find the parameters for the region of operation, the model from the fitting with $n_c = 4$, 7 or 9 rad/s will be used. The gain coefficient K is quite stable in this region, and therefore K = 0.98. The time constants on the other hand are quite

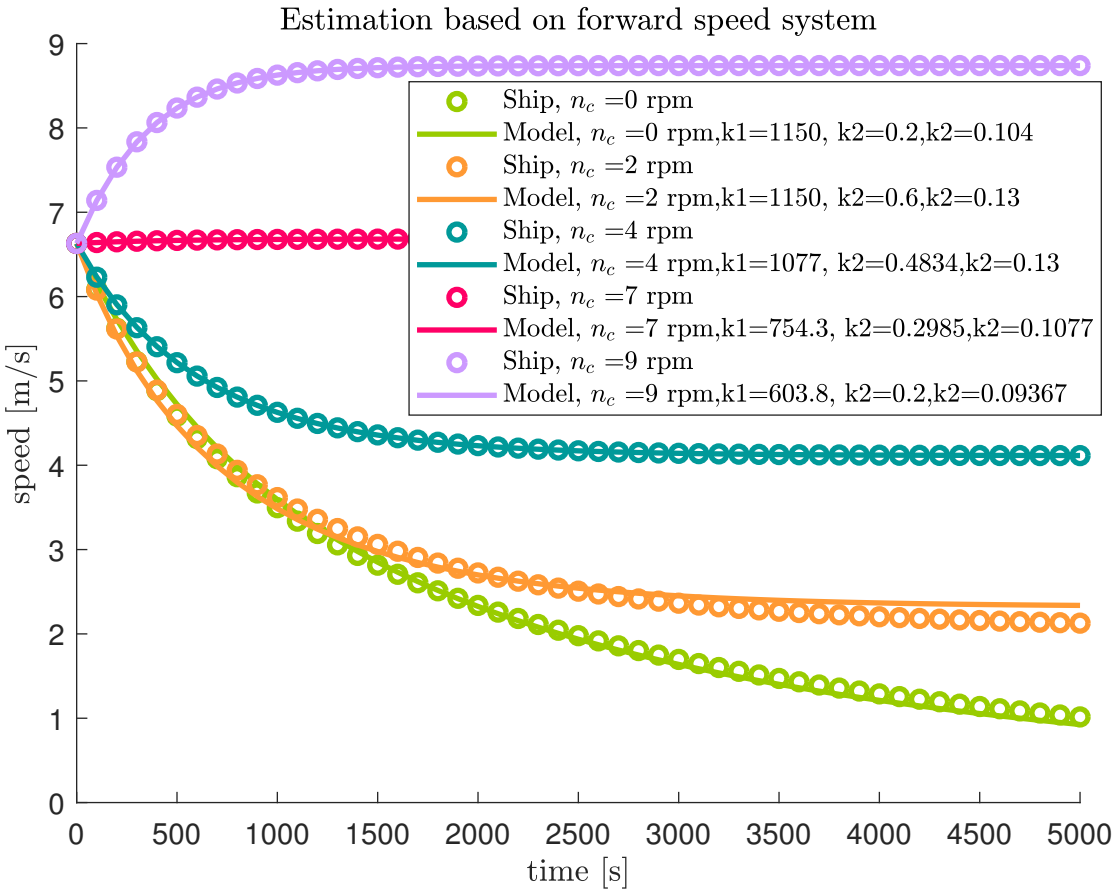


Figure 7: Compared results of the estimation of the linearized speed forward model response vs. MS *Fartøystyring* response. The different responses are from different steps in n_c.

dependent on n_c, ranging from $T = 350.7$ to $T = 615$, for $n_c = 4$ to $n_c = 9$. To get a controller which works best in the region of $3 - 8$ m/s the median value is used, giving a time constant of T = 434.4. This results in the following transfer function:

$$H_u(s) = \frac{u(s)}{n_c} = \frac{0.98}{1 + 434.4s} \tag{16}$$

In section 1.5 it was stated that a normal feedback control system stabilised in surge, sway and yaw have natural periods in the range of 100 - 200 s. This values will depend on the ship, and since our values are in close proximity of 200 s, they seem reasonable.

1.7: Designing the Controller

The PID controller is an industry favourite. In this case however, we do not have access to any measurements of the linear acceleration of the ship, i.e. the function block that represents

the ship does not provide us with $\dot{u}$. The derivative part of the PID controller would thus require estimation of $\dot{u}$ based on the measured state u. As we are using a 1st order linear model however, a PI controller should suffice for the ships forward speed control. As we will see shortly, this may assure convergence for our linearised model.

It is worth noting that had we chosen to move forward with the nonlinear model of the forward speed we could have attempted cancelling out the nonlinearities with a suitable nonlinear control law. Here, however, we have chosen to proceed with the linearised model.

We define $\tilde{u} := u_d - u$. Note that this is different than what the assignment text says, as this is the form which we are accustomed.

$$\tau_c = K_p\tilde{u} + K_i \int \tilde{u}\, d\tilde{u} \tag{17}$$

Examining the closed loop transfer function

$$\frac{\tilde{u}}{u_d} = \frac{1}{1 + h_p h_c} = \frac{K(K_p s + K_i)}{Ts^2 + (K_p K + 1)s + KK_i} \tag{18}$$

we concluded that for the poles to be in the left half plane, and thus the system to be stable, we would need

$$K_i K T > 0 \tag{19}$$

As we found earlier $T > 0$ and $K > 0$, we thus need $K_i > 0$ for this system to be stable. As this is a linear system we can conclude that it is in fact global asymptotically stable (GAS) and thus converges towards the origin. As we have no poles in the origin less $K_i = 0$, analysis of the convergence of the system with the controller stated in equation (17) may also be done using the final value theorem:

$$\lim_{t\to\infty} \tilde{u}(t) = \lim_{s\to 0} \tilde{u}(s)s \tag{20}$$

Laplace transforming the controller law and the plant to find their corresponding transfer functions yields

$$h_c(s) = K_p + \frac{K_i}{s} \tag{21a}$$

$$h_p(s) = \frac{K}{1 + Ts} \tag{21b}$$

Using this to find the transfer function from u_d to $\tilde{u}$, then inserting the resulting equation for $\tilde{u}$ into the final value theorem from equation (17), results in:

$$\lim_{t\to\infty} \tilde{u}(t) = \lim_{s\to 0} \frac{u_d(s)(1 + Ts)s^2}{(1 + T)s + K(K_p s + K_i)} \tag{22}$$

We can see that for both the impulse response and the step response (i.e. $u_d = s$ and $u_d = \frac{1}{s}$ respectively) the limit in equation (22) goes to 0. Thus we can conclude that in both cases $\tilde{u}$ converges to zero with this controller.

Current at sea is a slowly varying disturbance and is therefore not so different from a constant disturbance. Constant disturbances may cause a stationary error to appear if the controller lacks sufficient integral action. This will however not be of concern to us here as we are using a PI controller. The integral part sums up the error in the system until the constant error has been cancelled. In the case of the current, which is comparable with a constant error, this will be enough to sufficiently cancel its effects.

We have here assumed a linear relationship between τ and n_c, (i.e. $\tau = k_{nc} n_c$) which of course is highly unlikely to be the case. This simplification have however not caused any noticeable damage in the controller results. For simplicity the value of k has been set to 1 and its effects have thus been baked into the other values of the system. This way our controller will correct for the error this assumption introduces.

1.8: Tuning the Controller

When the time came to tune our controller parameters we first took a shot at `MATLAB`s own tuner function `pidtune`. By first running `tfest`, the function version of the `MATLAB` app 'System Identification', to find the parameters of the model for different values of n_c. We then ran the PI controller with both the MS *Fartøystyring* block and the model with the given parameters for each of the previous n_c values in order to use `pidtune` to find suitable K_i and K_p values for each case based on the respective model parameters.

The resulting plot and controller parameters suggested by `pidtune` are shown in fig. 8 and table 4 respectively.

Table 4: Table of controller values suggested by `pidtune`.

n_c	K_p	K_i
4.1888	1.4550	0.0042
6.8068	1.4590	0.8409
8.9012	1.4579	0.0076

By doing this we could conclude two things we already suspected to be true: Our model and the true ship were not the same and thus `pidtune`, which uses our model, did not result in particularly well tuned parameters. This can be concluded from the plot in fig. 8 as we can see the controller works way better on the model than the true ship. This makes sense as `pidtune` relies on the model to provide parameters for good tuning. We can conclude that in order to be able to use these model based tools for e.g. tuning we would need a more accurate model of our system. Even though the model for which we proved convergence is faulty, the controller may still be sufficient with the right tuning, although this has to be done manually.

As per ordered by the assignment text the desired speed was kept stationary until a step was introduced at $t = 500s$. All initial values were kept as stated in the assignment text and ϕ_d was kept at 0. The current was kept on.

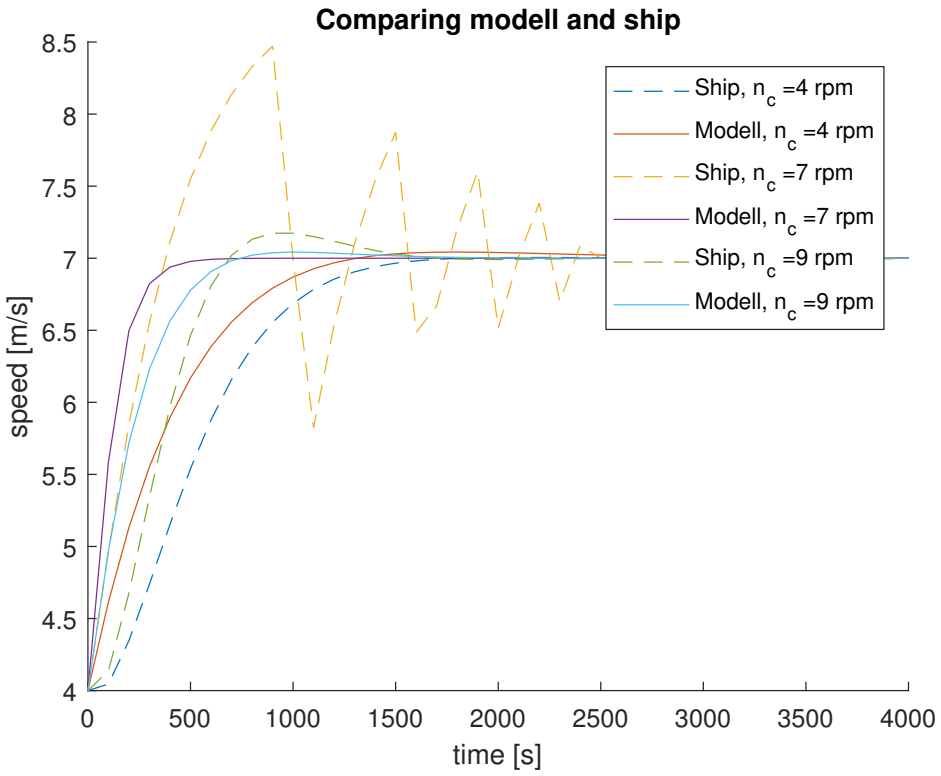


Figure 8: Step response of model and "true" ship for different values of n_c and controller parameters given by `pidtune`.

Manual tuning resulted in the following parameters:

$$K_p = 80,\ K_i = 0.08 \tag{23}$$

which were much larger than the ones returned by `pidtune`. K_p for instance, is approximately 10 times the size of what `pidtune` suggested.

To avoid overshoot by integrator wind up, we introduced a switch in the integral action such that the integral action would only be active when the state was close to its desired value. This way the integral part would be able to cancel out the stationary error while leaving the bigger transitions to the proportional part. The chosen switching value on the error was 0.7 radians.

As we can see from the plot of ψ in Figure 10, ψ is initially not able to keep its value stationary seen by the jump from its zero value before returning to zero once more before time 500s. This might be due to weaknesses in model (e.g. nonlinearities) or a hidden trait of the ship simulation. Either way we presume this might be the reason why we were asked to wait until $t = 500$ before activating the step. As seen in fig. 9, the speed is slightly affected by this behaviour as turning violate the assumption of purely forward speed in the speed model, but in this task it has no large impact because of the high proportional gain in the speed controller. The reality of the speed and heading being coupled will however become more

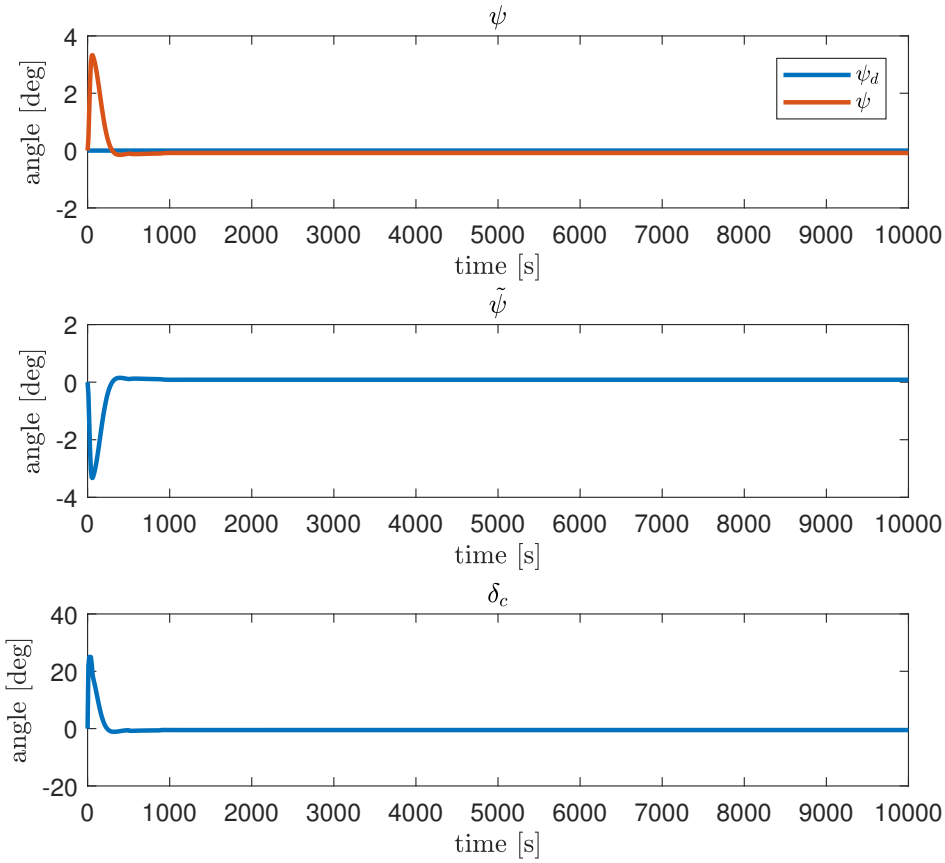


Figure 9: Simulation of ψ when step in u and $\psi_d = 0$, using speed and heading controller.

present in later tasks.

When the step in desired speed is introduced, we can see the speed approaches the new desired value with a relatively steep slope before quickly converging to the desired value without overshooting. From the plot of n_c, in fig. 9, we can see that the $K_p u$ does the heavy lifting of bringing the the state close to the the desired value, while $K_i u$ keeps it at the desired value, compensating for stationary errors. From the same plot we can see how most of the controllers proportional part is lost due to the saturation of n_c. This suggest that the value of K_p could have been less. A smaller value in K_p did however cause the speed to dip down before increasing towards the desired value, slowing down the convergence.

In order to keep the speed at 7, $K_i u$ must keep a stationary value. This suggests the current is a constant (or at least very slowly varying) which coincides with our previously stated assumptions.

Also worth noting is the fact that the sudden change in speed seems to not affect the heading when it keeps the constant value, as seen in fig. 10. This is also a desirable trait.

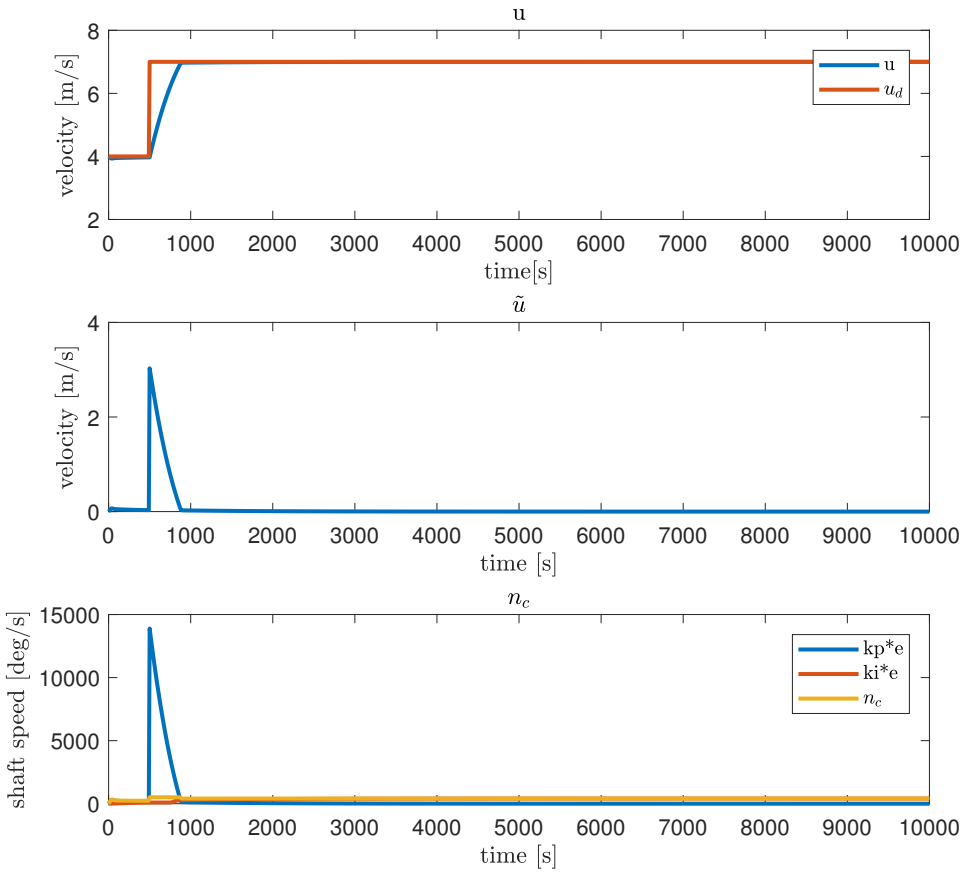


Figure 10: Simulation of u when step in u and $\psi_d = 0$, using speed and heading controller.

Path Following

2.1: Designing the Guidance System

To have the MS *Fartøystyring* follow a piece-wise linear path generated by a set of waypoints, a guidance system needed to be implemented. The initial values for the ship was:

$$\text{Position: } [x(0) = 1500\,\text{m}, \quad y(0) = 500\,\text{m}]^\top$$
$$\text{Velocity: } [u(0) = 6.63\,\text{m/s}, \quad v(0) = 0\,\text{m/s}]^\top$$
$$\text{Heading: } \psi(0) = 50^\circ \quad \| \quad \text{Yaw rate: } r(0) = 0\,\text{rad/s}$$

The guidance system implemented for MS *Fartøystyring* has its roots in lookahead-based steering. To guide the ship, the course angle is divided into two parts: the path-tangential angle ($\chi_p = \alpha_k$) and the velocity-path relative angle ($\chi_r(e)$). The path-tangential angle between two waypoints is defined as

$$\alpha_k := \text{atan2}(y_{k+1} - y_k \,, x_{k+1} - x_k)\,,$$

where a waypoint is expressed as a point $\mathbf{p}_j^n = [x_j, y_j]^\top$ in the NED-frame. (See fig. 11 for a graphical representation.) From this, one can write the coordinates of the craft in the path-fixed reference frame as

$$\boldsymbol{\epsilon} = \mathbf{R}_p(\alpha_k)^\top(\mathbf{p}^n(t) - \mathbf{p}_k^n) \ ,$$

where $\mathbf{p}^n(t)$ is the ship position at time t and $\boldsymbol{\epsilon} = [s(t), e(t)]^\top$, where $s(t)$ is the distance along the path and $e(t)$ is the cross-track error normal to the path. To follow a given linear path exact, one would then aim to have $\lim_{t\to\infty} e(t) = 0$. The desired course angle then becomes

$$\begin{aligned} \chi_d &= \chi_p + \chi_r(e) \\ &= \alpha_k + \arctan(-K_p e) \end{aligned} \tag{24}$$

Due to some problems with the angle mapping, causing the ship to take unnecessary turns when changing waypoints, the calculations of χ_d was slightly modified, resulting in the desired course angle now being

$$\chi_d = \mod\{\alpha_k + \arctan(-K_p e),\ 2\pi\} \tag{25}$$

This lookahead-based steering is based on a circle of acceptance: a circle centred on the ship with a radius R of sufficient size as to intersect with the path to be followed. This radius is given by

$$R^2 = e(t)^2 + \Delta(t)^2 \ ,$$

where $\Delta(t)$ is the lookahead distance. These parameters were instead chosen constant; the radius of acceptance was chosen to be $R = 500$m whereas the lookahead distance was chosen $\Delta = 2.5L_{\text{ship}}$, where L_{ship} is the ship's length. The line of sight vector (LOS vector) is then defined as the distance from the ship to the intersection closest to the next waypoint between the circle of acceptance and the path. The updating of the waypoints happens when the ship is within a radius of acceptance centred on the current next waypoint ($\mathbf{p}_{k+1}^n$).

In addition to the guidance system itself, linear reference model was added for trajectory generation. Reference models are usually based on physical models and tend to give smoother actuator input signals. This is preferable on a real ship as it leads to less wear and tear on the actuators. In this case, the attitude reference model takes the form of a 1st order low-pass filter in cascade with a mass-spring-damper system, resulting in a 3rd order model. This gives the model

$$\frac{\eta_{d_i}}{r_i^n}(s) = \frac{\omega_{n_i}^2}{(1 + T_i s)(s^2 + 2\zeta_i\omega_{n_i}s + \omega_{n_i}^2)} \ , \tag{26}$$

where T_i is the filter time constant, ω_{n_i} is the natural frequency, η_{d_i} is the desired state and r_i^n is the reference signal. To acquire satisfactory performance and stability, the bandwidth of the reference model, $\omega_{b_{ref}}$, must be chosen lower than the bandwidth of the control system. Implementing the reference model, the frequency was chosen to be around 0.09 rad/s to give a non-oscillatory response for attitude. This smoothened the rudder input somewhat, but not to a very significant level. To further improve the rudder input, the reference model had to be altered in such a way that it resulted in a fairly inaccurate tracking. Evidently a compromise

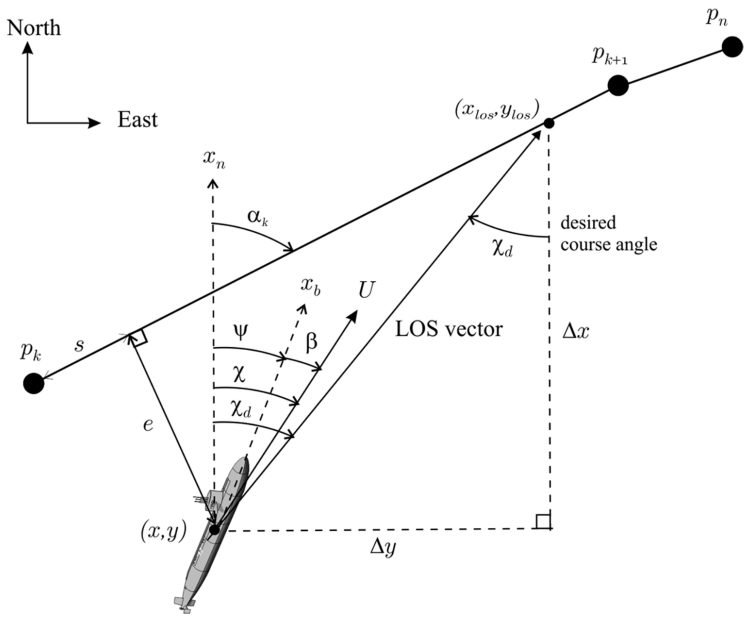


Figure 11: Illustration of the guidance system fundamentals. [Image credit: T.I., Fossen (2011)[3]

between robustness and accuracy has to be done. In our case, working with a simulated ship, the focus was put on accuracy, weighting the path following over input usage.

Reference model on velocity should be at least of order two so as to obtain smooth reference signals for the desired velocity u_d. The equation for a reference model of order two is:

$$\ddot{u}_d + 2\zeta_u \omega_{n,u} \dot{u}_d + \omega_{n,u}^2 u_d = \omega^2 r^b \tag{27}$$

With u_d as the desired velocity, $\dot{u}_d$ as the desired acceleration, $\ddot{u}_d$ as the desired "jerk", $\omega_{n,u}$ as the natural frequency and ζ_u as the relative damping ratio. This reference model will give a step in $\ddot{u}_d$ and low-pass filtered values for $\dot{u}_d$ and u_d, giving a smooth signal for tracking [3].

The poles of the the surge closed loop system are:

$$T_u s^2 + (1 + K_u K_{p,u})s + K_u K_{i,u} \tag{28}$$

Comparing the poles of the closed loop surge system to the poles of a mass-spring-damper system gives the natural frequency $\omega_{n,u} = 0.0115$. The frequency of the reference model was then chosen to be around 0.005 rad/s with a relative damping ratio to 1 to give a non-oscillatory response for attitude. This smoothened the rudder to a significant level, giving less wear and tear. A slow reference model leads to slower regulation in velocity. Since this is a simulation, the natural frequency of the reference model may be higher, resulting in a more accurate tracking.

2.2: Simulation without Crab angle Compensation

The results from running the simulation with the implemented lookahead-based guidance system, with a lookahead distance $\Delta = 2.5 \cdot L_{\text{ship}}$, is shown in figs. 13 to 12. The radius of acceptance centred at each waypoint indicating when the waypoints should be updated, was set to 500 m and the desired speed was set to 5 m/s. In this task it is assumed that the ship has no crab angle, i.e. that $\psi_d = \chi_d$.

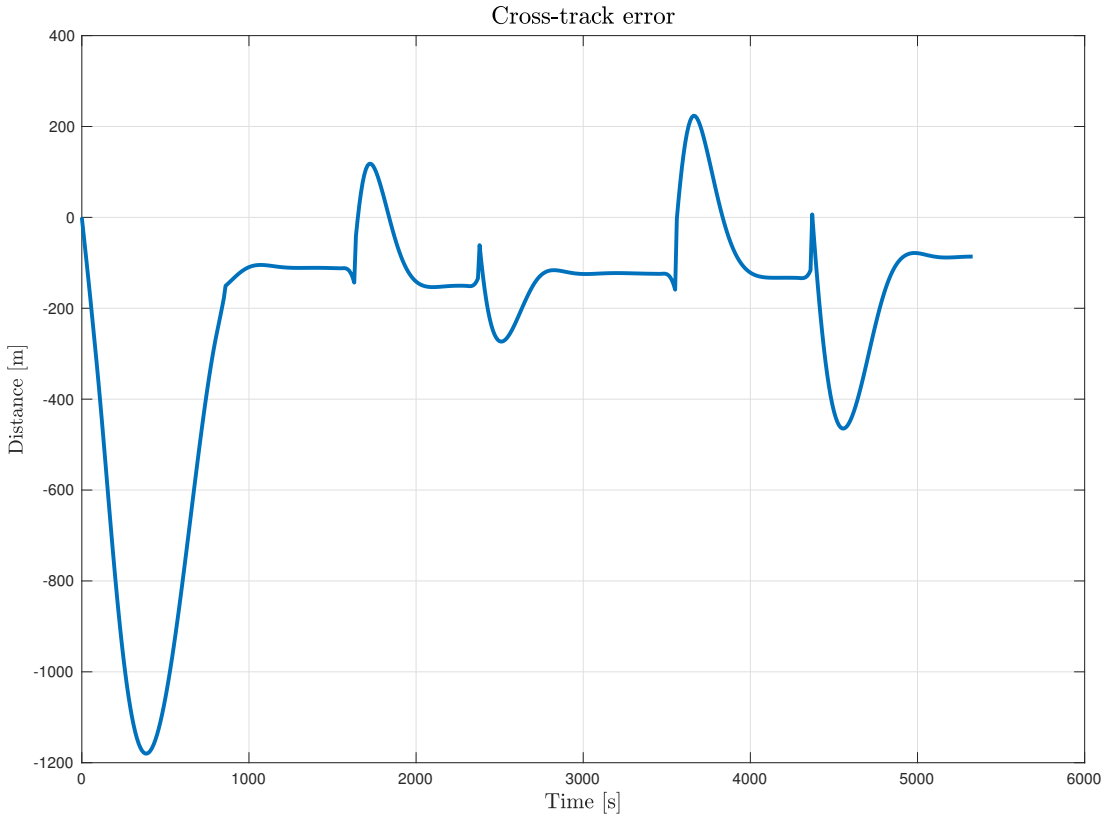


Figure 12: Cross-track error of MS *Fartøystyring* when following the piece-wise linear path.

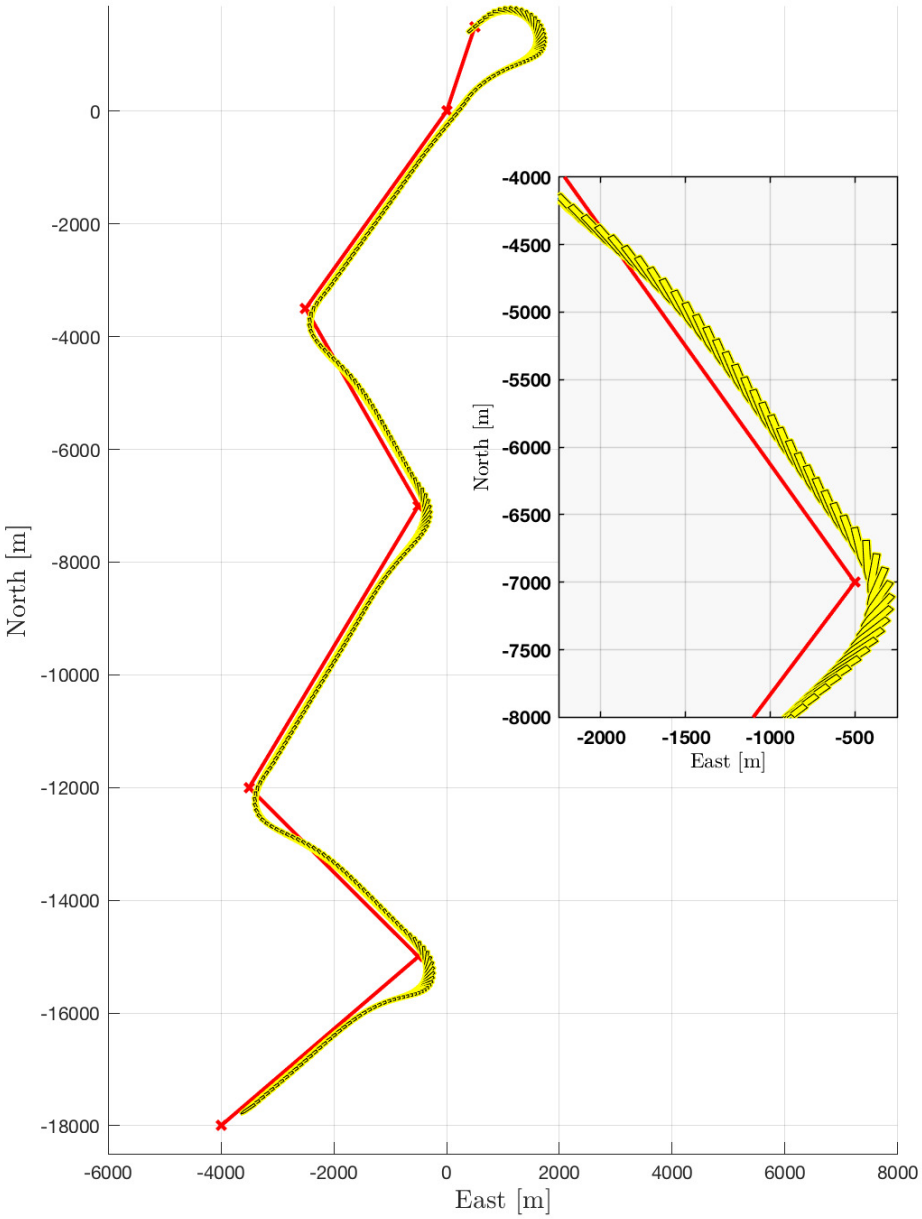


Figure 13: Path followed by MS *Fartøystyring* with lookahead-based guidance system and no crab angle compensation.

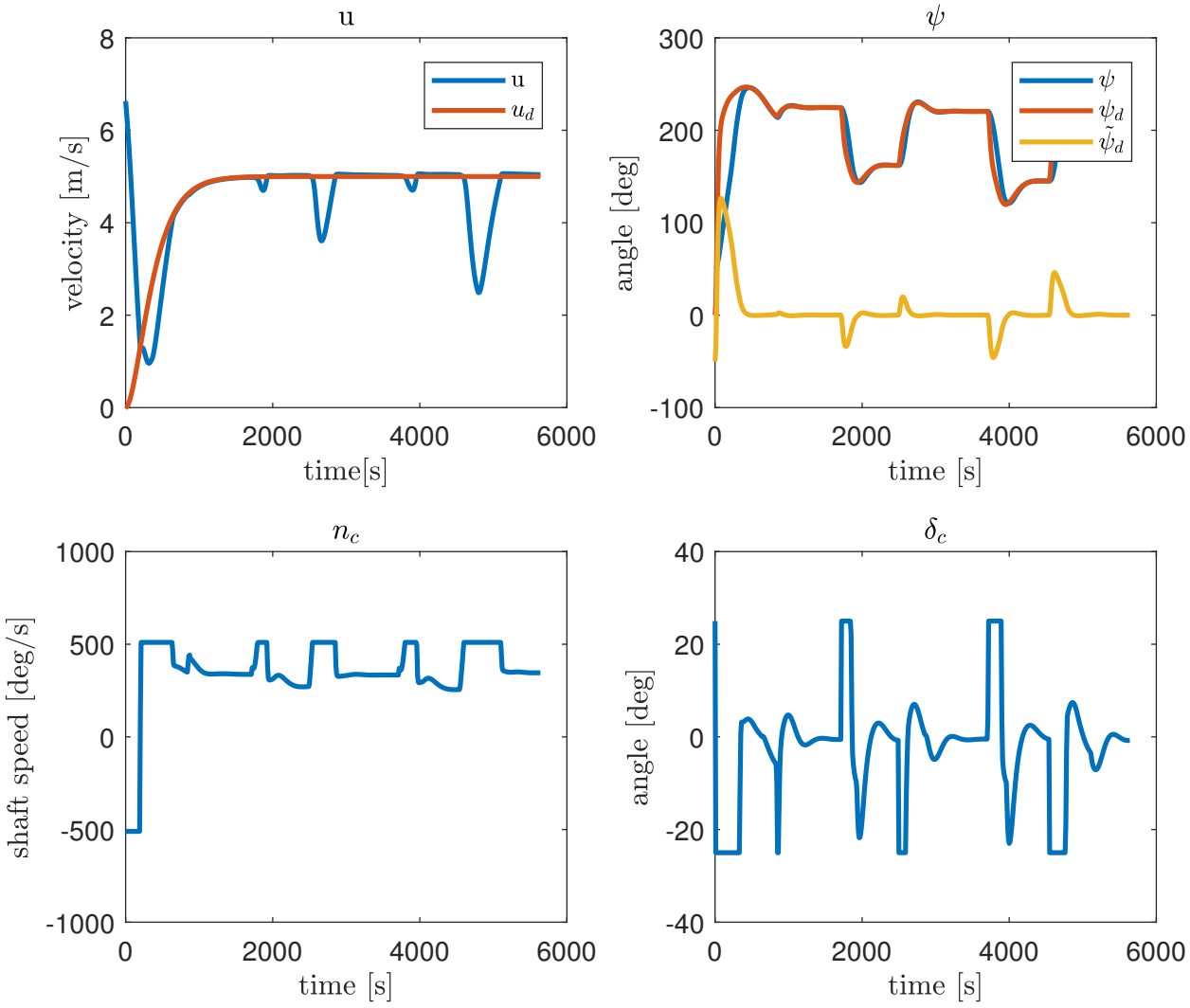


Figure 14: Speed, heading, shaft speed and rudder angle results when following the piece-wise linear path.

2.3: Explanation of Results

From fig. 13 it can be seen that the ship follows the path pretty well, except from a small constant error and some delays in the turning points. This is also noticeable when looking at the cross track error in fig. 12.

In fig. 14 the error in the heading angle can be seen as almost equal to zero, except from the small spikes from when the ship is turning. From this one could believe that the ship would stay on track, but because of not having any crab angle compensation the controller gets the wrong input reference value and will therefore produce an offset in the path. The ship is not able to keep up at the desired surge velocity in the turns. The surge model is also made for the ship following a straight line, and is not optimised for making turns like we do. When looking at the shaft speed one can also notice that it saturates during the turns, giving full speed when turning the ship. So trying to give the shaft speed a higher reference value would not help since it is already maxed out.

The cross-track error when following the path is shown in fig. 12. The spikes come from the greater distance from the waypoints when the ship is turning. One can also notice that the cross track error centres at about -100 m from the path. This is most likely caused by being exposed to current.

Overall this is the expected behaviour of the ship. Without any crab angle compensation a constant error from the path is expected. Some errors from the models and the fact that the controllers are tuned to behave nicely for a ship following a straight path makes these results satisfactory for our ship following a path generated from the waypoints.

Path Following with Crab angle Compensation

2.4: Illustrating the Problem

When following a path without crab angle compensation, the crab angle is assumed zero and the course angle is then assumed equal to the heading angle of the ship. If this is the case, and the crab angle really is zero, then one can assign the desired course directly from the heading angle without any problems. However, when the crab angle is nonzero, if the reference signal sent into the heading controller is the desired course, it will include an offset (the crab angle, β) and no longer be suited as a reference signal. The surge controller will also be affected by the crab angle. The total desired speed will no longer be only in the surge direction, but will also have some component in the sway direction. The desired speed in surge direction will therefore be a function of the total desired speed and the crab angle.

From fig. 15 it can be seen that the crab angle in this case is nonzero. This results in the heading angle, instead of the course angle, following the desired course χ_d, as the desired course is used as the heading reference signal. Without crab angle compensation also $u_d \neq U_d$. This is not the desired behaviour of the system. Crab angle compensation should therefore be included so that the heading reference signal is the actual desired heading ($\psi_d = \chi_d - \beta$). This should make the heading follow the desired course with an offset equal to the crab angle, and

make the course angle the one to follow the desired course χ_d. Crab angle compensation will also solve the problem with $u_d \neq U_d$, when the system is exposed to current.

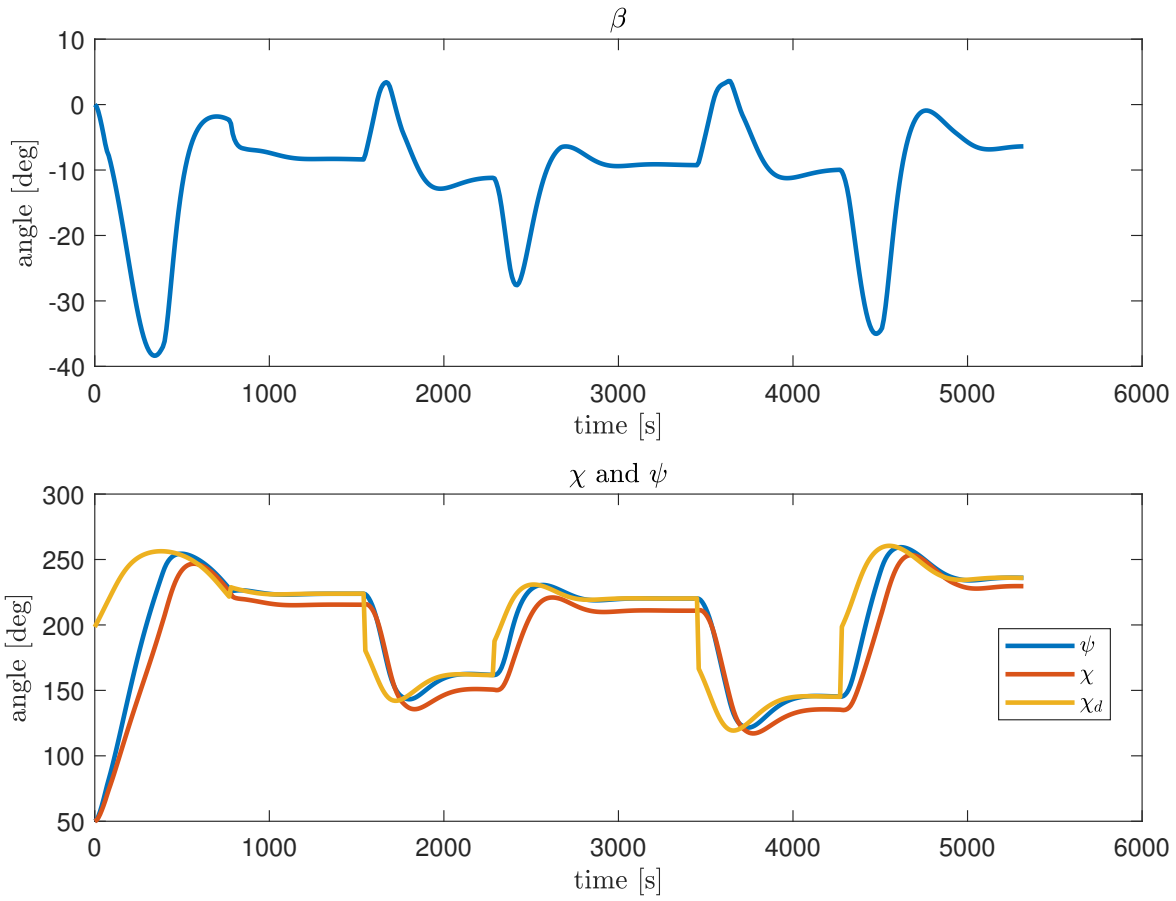


Figure 15: A nonzero crab angle, β, gives a difference in the heading angle, ψ, and the course angle, χ.

2.5: Adding Crab angle Compensation

To account for some of the experienced drift, crab angle compensation are added. There are two ways of defining the crab angle for at 3 DOF system with $w = 0$:

$$\psi_d = \chi_d - \beta \tag{29}$$

$$\beta = \arcsin\left(\frac{v}{\sqrt{u^2 + v^2}}\right) \tag{30}$$

The equations stated above may be be used to transform from χ_d to ϕ_d and from U_d to u_d. The transformations becomes:

$$\psi_d = \chi_d - \beta \tag{31}$$
$$u_d = \cos(\beta) U_d \tag{32}$$

To calculate ψ_d it will be needed to use the the estimated beta and the desired values calculated in the guidance system.

2.6: Simulation with Crab angle Compensation

With crab angle compensation it is expected that the the LOS algorithm should follow the path better, compensating for the current.

The resulting path tracking is plotted in fig. 16, demonstrating that the ship now better follows the path, due to the effect of crab compensation. Looking at the zoomed-in section in figure 16, there is now little or no discrepancy between the path and the boat regulation in steady state. However there is still some offset between the ship's planned path and the ship's actual path in big turns, due to the fact that the vessel can not turn instantaneously. In order to reduce observed overshoots when turning, one may increase the acceptance radius. This would give smoother transitions between the different waypoints, since the ship would shift α_k earlier. This will also affect the proximity of the ship and the waypoints to be smaller.

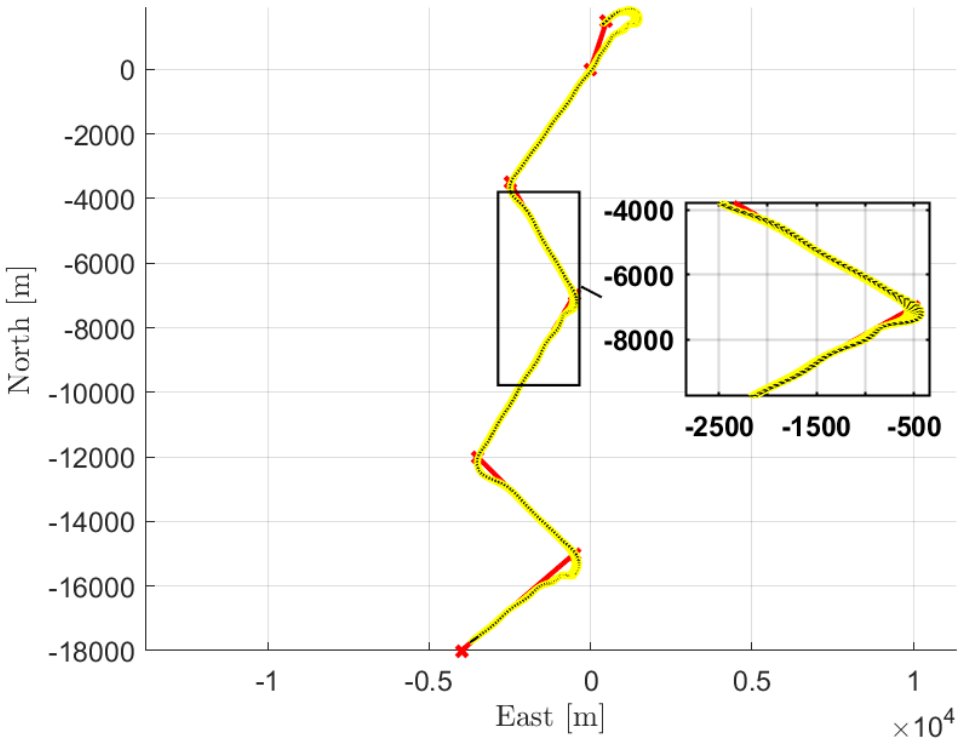


Figure 16: Plot of the waypoint following, with crab compensation.

The plot for cross track error confirms the observations from the path plotter. In fig. 17 it is evident that the cross track error is centered around 0m, after he vehicle has reached the first way point. Before crab angle compensation the cross error was typically centered around approximately -100m. As before there is some deviation from the path when the ship is turning, but this is because the boat is not able to turn immediately around a point.

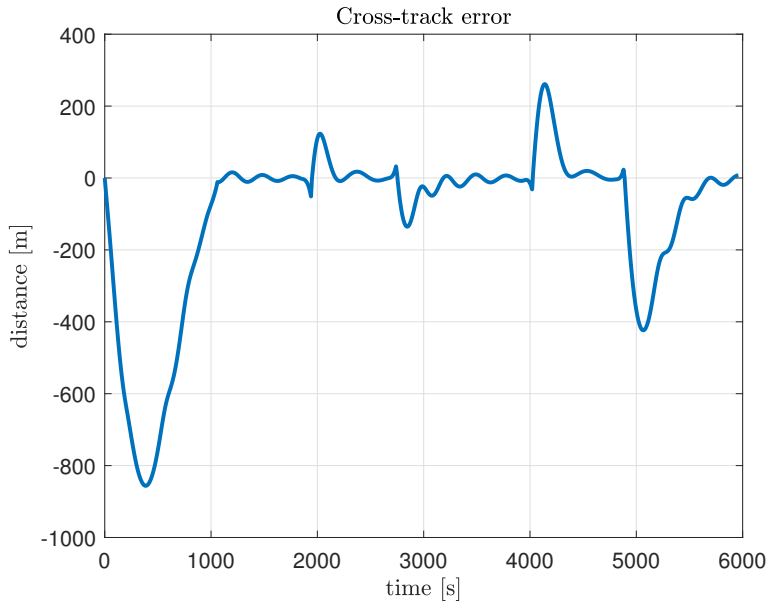


Figure 17: Cross track error after implementing crab angle compensation.

The fact that the vessel follows the path better is also noticeable in the figure 18, where ψ follows ψ_d , instead of χ following ψ_d as it did without crab angle compensation.

The compensation in crab angle also affected the regulation of u_d, which for this assignment is 5 m/s, as seen in Figure 19. After transforming U_d to u_d the system followed the desired value better. One interesting observation is that u retains a similar form as before the crab angle compensation, while the desired value experiences more change. u_d now depends on the crab angle, meaning the turning dynamics are now also included in the calculation of u_d. This is the same dynamics the ship experiences while turning, making u_d easier to follow.

As before crab angle compensation, the regulation of u and ψ does not indicate ideal decoupling during tuning of the system, meaning that it is not possible to both control u and ψ perfectly at the same time with only one propeller, with δ_c and n_c. Looking at the plot of the thruster values, it is noticeable that when d_c saturates, caused by a leap in the desired heading values, the value of u experiences a sudden change in value. This is consistent with the fact that the decoupled dynamics is not realised for MS *Fartøystyring*, which would also be unrealistic. If two controller forces were decoupled, it would hopefully be easier to follow the desired values.

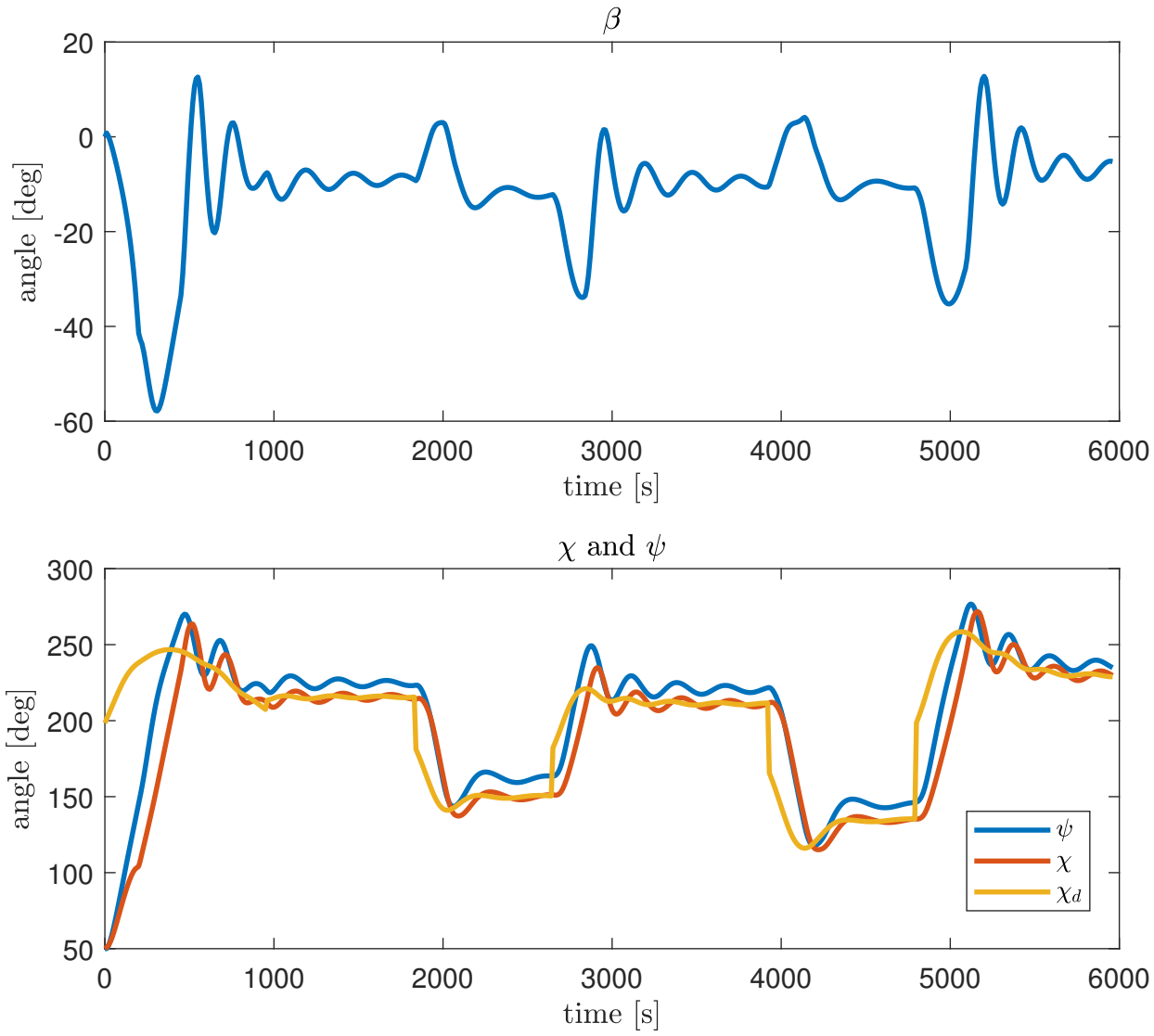


Figure 18: Plot of heading, course angle, desired course angle and crab angle with crab angle compensation.

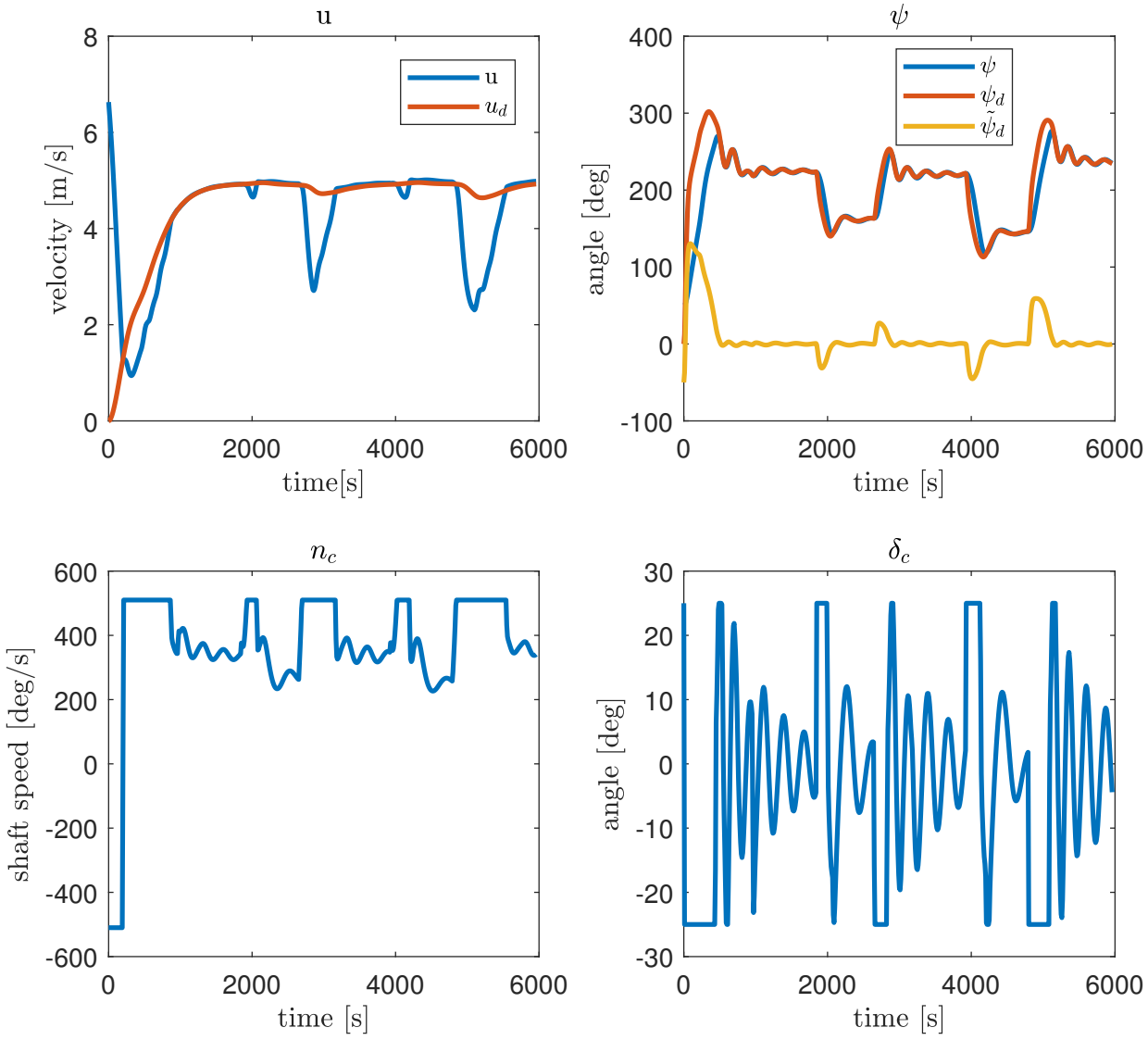


Figure 19: Plot of ψ, n_c, δ_a and u, with crab compensation.

2.7: Target Tracking

In this task we were assigned to follow another vessel at a certain distance. We chose this distance to be 400 m as this would be sufficiently close without crashing into the other ship, assuming equal length of the two ships. As we are situated on the sea, with the assumption of no other obstacles, a two-point guidance systems seemed like a satisfactory approach to the task of catching up to and follow a target. Hence, both Pure Pursuit (PP) guidance and Constant Bearing (CB) guidance was considered. PP is reminiscent of how a predator would chase its pray and is often used in guidance of air-to-surface missiles [1]. CB on the other hand is more comparable to convergence on a rendezvous point, where the velocity of the target is taken into account. CB is often referred to as parallel navigation and is more typically used for air-to-air missiles[1]. Of the two we chose to use CB as it is usually more energy efficient and less prone to a tale chase scenario.

As we are not intending on crashing into the targeted vessel, only to follow it, the target position was set to 400 m behind the targeted ship. This way we have moved the target to a spot that follows the same trajectory and speed as our target ship and we can use this as our targeted position without crashing into the other ship.

At first we reused our LOS algorithm for the calculation of the heading, this time using only two waypoints for the path, where the second of the two was the moving target position. When this was done, the desired speed U_d was calculated from CB (as described in [3]) by using the norm of v_d and then u_d from the resulting U_d multiplied by $\cos(\beta)$. This resulted in a chase of the target that was more reminiscent of PP navigation. The ship converged to the same path that the target followed and used CB calculations for control of speed to catch up and then keep the 400 m distance to the targeted ship as can be seen from fig. 21. This merge of method was motivated by the fact that we use heading and speed control and not velocity control as is the case with CB. Later however, we proceeded to instead calculate the heading directly from the CB velocity calculations, hereby fully implementing CB.

The equations used for CB was

$$\boldsymbol{v}_d^n = \boldsymbol{v}_t^n + \boldsymbol{v}_a^n \tag{33}$$

$$\boldsymbol{v}_t^n = [U_{target}\cos(\alpha),\, U_{target}\sin(\alpha)] \tag{34}$$

$$\boldsymbol{v}_a^n = \frac{-\kappa\tilde{\boldsymbol{p}}}{||\tilde{p}||} \tag{35}$$

where $\kappa = \frac{(U_{max}-U_{target})||\tilde{p}||}{\sqrt{\tilde{p}^\top\tilde{p}+\Delta_p^2}}$,$\tilde{p}$ is the vector from the ship to the target and $U_{max} = 7$. Δ_p was tuned and we used a switching mechanism to switch between two values depending on the distance from the target. Δ_p is changed from 1000 to 5600 when $|\tilde{p}|$ are less than 2500. This makes the system faster.

This helped optimising the results.

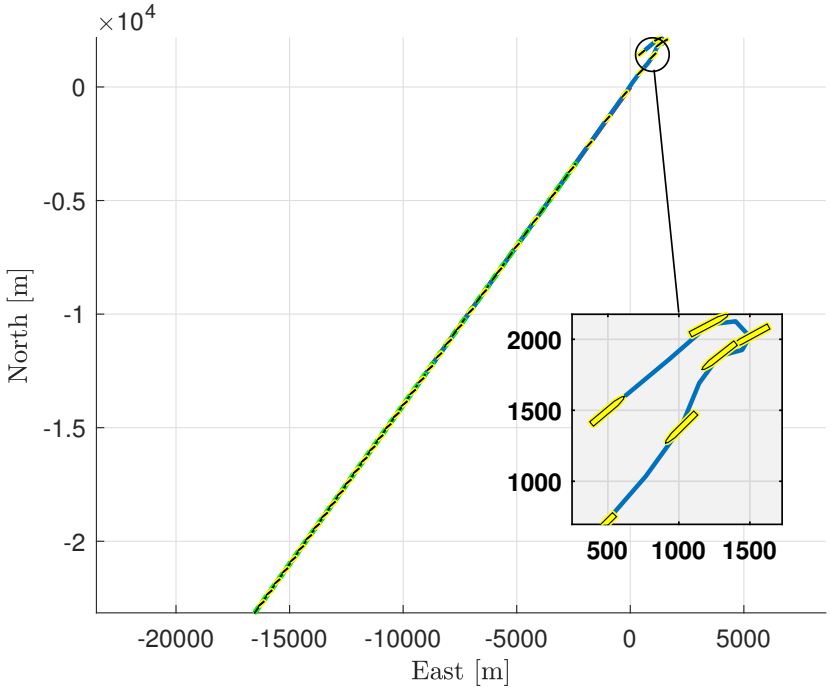


Figure 20: Simulation of the ships positions with LOS and CB. (Sampling time of 100 s.)

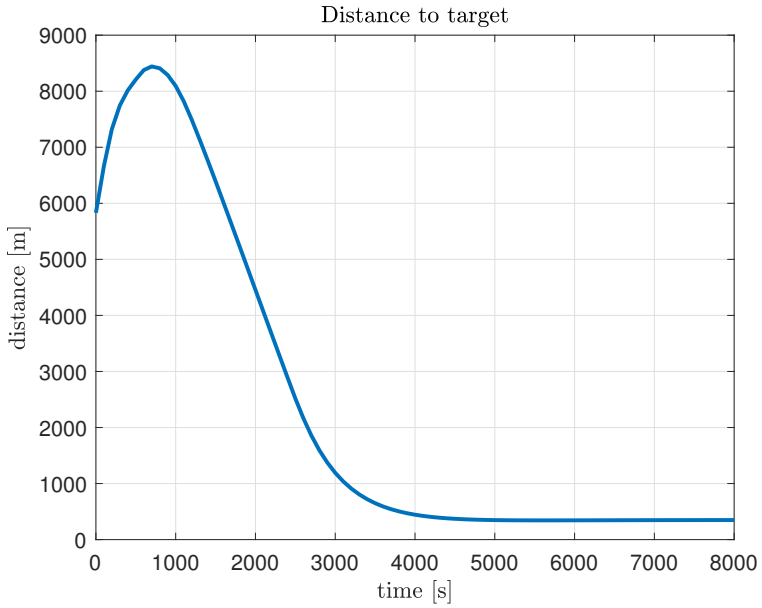


Figure 21: Distance from target ship with LOS and CB. (Sampling time of 100 s.)

As a two-point guidance system, the only positions needed are the position of the ship and the position of the target. The CB thus used the same target position as the second waypoint of the LOS algorithm used previously for its calculations. The desired speed for the reference model was still calculated as previously described, but now the heading would have to be calculated from the velocity. As the velocity given by CB was given in NED coordinates we could use regular trigonometric identities to calculate χ_d.

$$\chi_d = \text{atan2}(v/u) \tag{36}$$

With CB fully implemented the ship took a different approach (literally) to the target position, turning the other direction than before and moving on a straight line towards the calculated rendezvous point before keeping this target position. This implementation was only slightly faster in its convergence than the previous implementation. The rudder also had somewhat less alternations with pure CB than when LOS was included.

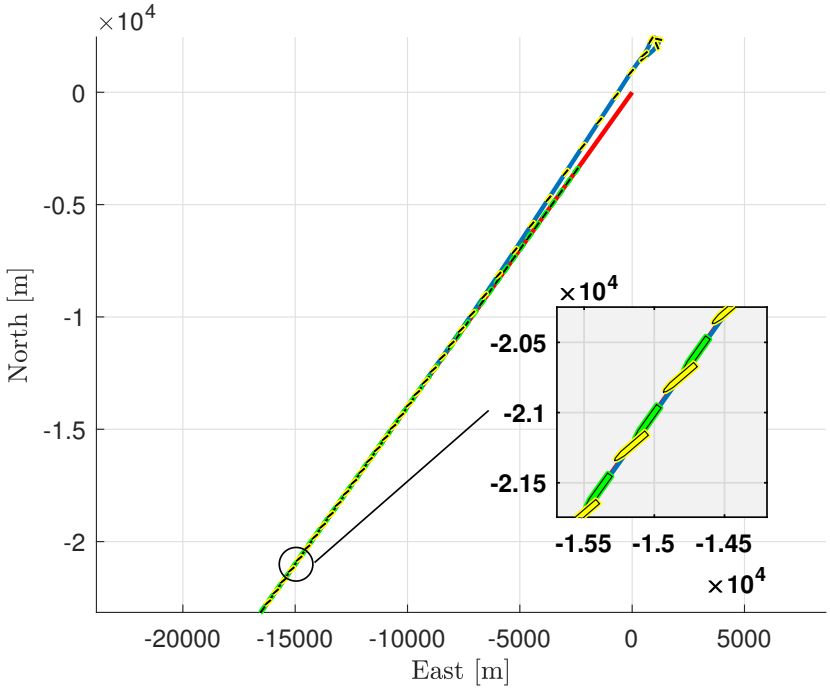


Figure 22: Simulation of the ships positions with CB. (Sampling time of 100 s.)

The results from the simulation with this target tracking algorithm are displayed from fig. 22 to fig. 25. From fig. 23 it is evident that our ship converges to a spot 400 m away from the targeted ship and that it keeps this placement. From fig. 22 and fig. 24 we confirm that the ship does not diverge from the path followed by the targeted ship after entering it and from fig. 25 we can see that the speed converges to the target speed, which is as expected. As before the speed follows its referenced value better when moving on a straight line devoid of turns. We also observe that both rudder and shaft speed still somewhat saturate in the initial sharp turn, while otherwise keeping below their respective maximum values. This behaviour is to be expected now that a larger chunk of the path is a straight line.

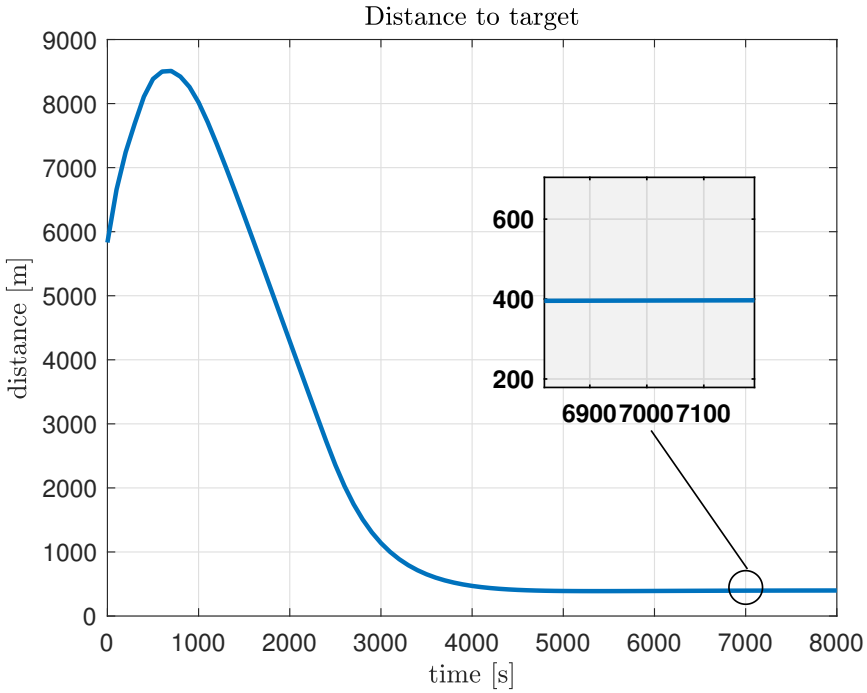


Figure 23: Distance from target ship with CB. (Sampling time of 100 s.)

From fig. 22 we see that even though we follow the targeted ship correctly on its path, the heading is different. This may be because the guidance algorithm specifies what heading is necessary for convergence on the target, but not how to keep the most efficient heading while on the moving target after that. This different heading makes for less streamlined travelling and less efficient power consumption. Additional restrictions on the heading when close to the target could be a possible improvement.

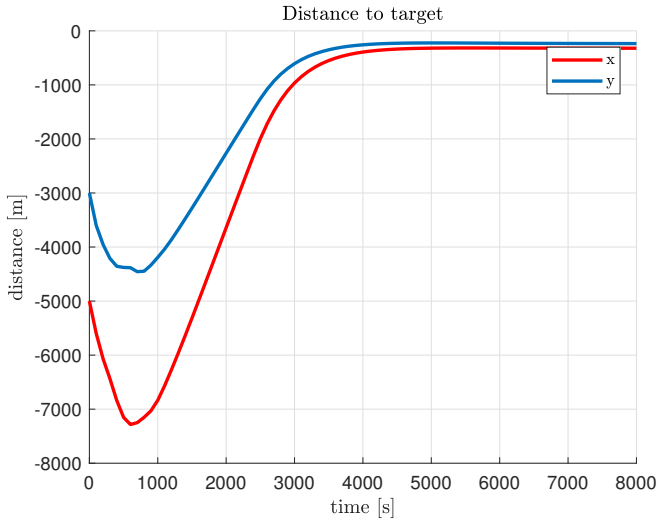


Figure 24: Distance from target with CB, decomposed into north and East components. Sampling time is 100 s.

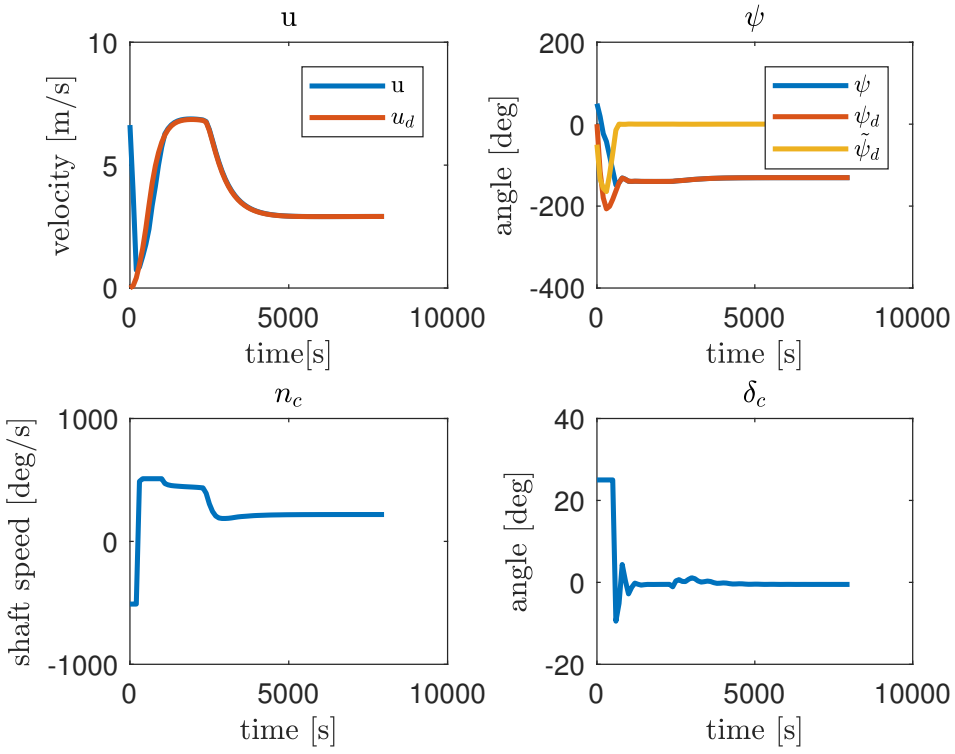


Figure 25: A selection of plots of the states and inputs from the target tracking simulation with CB. Sampling time is 100 s.

References

[1] Morten Breivik and Thor I. Fossen. "Guidance Laws for Planar Motion Control". In: *47th IEEE Conference on Decision and Contro* (2008), pp. 572–573.

[2] *Conversion between model types.* `https://se.mathworks.com/help/control/ug/conversion-between-model-types.html`. Accessed: 2017-11-6.

[3] T.I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, 2011.

[4] T.I. Fossen. *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles.* Springer, 2002.

[5] Tristan Perez. *Ship Motion Control: Course Keeping and Roll Stabilisation Using Rudder and Fins.*

[6] *tfdata.* `https://edoras.sdsu.edu/doc/matlab/toolbox/control/ref/tfdata.html`. Accessed: 2017-11-6.