NTNU

TTK4190

Guidance and Control of Vehicles

Autopilots and Path Following/Tracking

Group members:	*Student number:*
Kjetil B. Kjeka	730839
Ole N. Lyngstadaas	738006
Truls K. Olsen	737779
Even Ødegaard	730502

November 17, 2015

(This page is intentionally left blank)

Contents

1 Autopilot Design

1.1 Heading Autopilot

Task 1.1

The Nomoto model describe a relationship between δ and r. The second order model is [1, p. 143].

$$\frac{r}{\delta}(s) = \frac{K(1+T_3 s)}{(1+T_1 s)(1+T_2 s)} \tag{1}$$

which can be simplified into the first order model by choosing the time constant

$$T := T_1 + T_2 - T_3 \tag{2}$$

such that

$$\frac{r}{\delta}(s) = \frac{K}{1+Ts} \tag{3}$$

The second order model is able to capture more dynamics than the first order model, namely sway coupled on yaw motion [2]. This dynamic is needed for considering the overshoot when turning but not when designing a feedback controller. Since the second order model becomes ill conditioned as the difference between T_2 and T_3 decreases the first order model is preferred for designing feedback control. This is why we have chosen the first order Nomoto model for controlling MSFartøystyring's heading.

Task 1.2

The first order Nomoto model can never capture the full dynamics of MS Fartøystyring. Meaning that we can not be looking for parameters that will make the Nomoto model behave like the ship but instead we should find parameters making the Nomoto model behave similar to the ship. The problem is defining a metric of similarity appropriate for our application. Fossen[1, Example 12.2] simply uses a quadratic error integrated over some time[1]. The parameters that gives the most similar Nomoto function will

[1]Fossen[1] doesn't say anything about which time frame that should be used for this quadratic estimation. He also states in Example 12.2 that Turning circle method with $\delta_0 = 5°$ is beeing used, but on p. 355 it states that the rudder angle should be at least $15°$ when using this method. Fossen seems to lack consistency and thoroughness when discussing how to find the Nomoto gains.

then greatly vary in regards to integration limits (time). This may not seem very comforting until we remind ourselves that the Nomoto model is going to be used for feedback control and doesn't need accuracy for simulation. Still we would like the systems to at least have the same steady state for some reasonable rudder angle.

The way the Nomoto gains are going to be found is similar to Fossen's example considering a step response of the ship $\delta_c = 10°$ and requiring that the steady state gain of the Nomoto model matches the actual ship.

$$K = \lim_{t \to \infty} \frac{r}{\delta}|_{\delta=10°}$$

And then using the lsqcurvefit Matlab approximation function finding the Tk, giving:

$$\begin{aligned} K &= -0.0331 \\ T &= 100 \end{aligned}$$

The step response of the Nomoto model and the full system is shown in Figure 1. The position while doing the test is shown in Figure 2. The Nomoto model's capability of tracking a time varying signal (in this case a sine) compared to the full model's capability is shown in figure 3, and we can see that the Nomoto model tracks quite well.

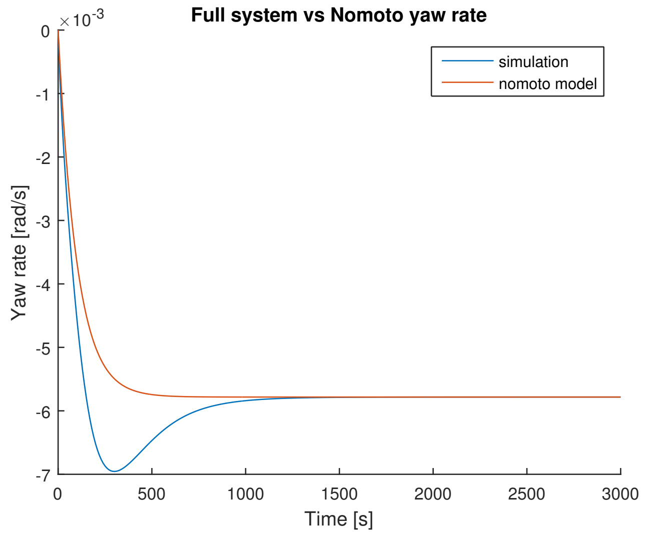

Figure 1: Step response of the system with a rudder input equal to 10°

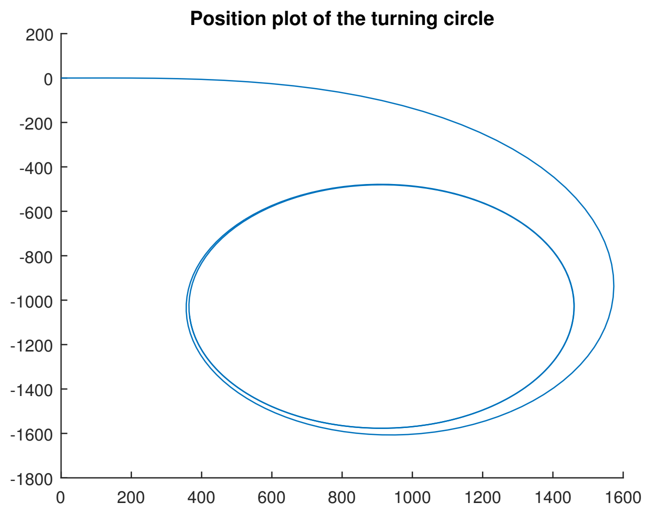

Figure 2: Position of ship while finding the step response of rudder input off 10°

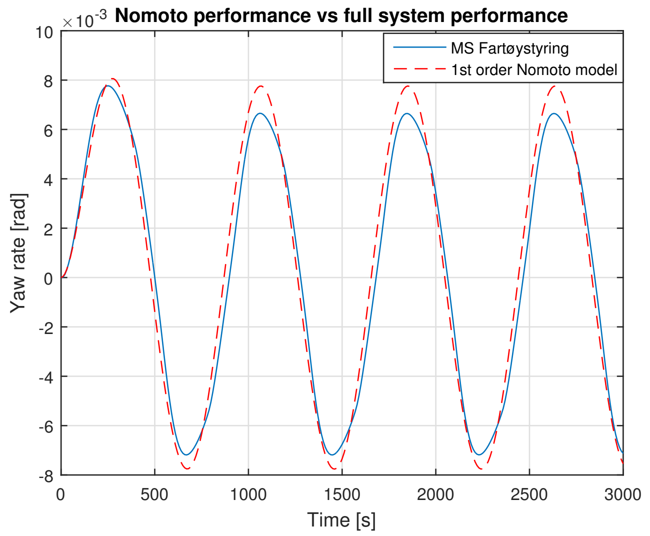

Figure 3: Nomoto performance vs full system performance

Task 1.3

We choose a PID-controller to control the heading, giving

$$\delta_c = -K_p\tilde{\psi} - K_d\dot{\tilde{\psi}} - K_i\int_0^t \psi(\tilde{\delta_c})d\delta_c = -K_p\tilde{\psi} - K_d\tilde{r} - K_i\int_0^t \psi(\tilde{\delta_c})d\delta_c \quad (4)$$

We have to ensure that $lim_{t\to\infty}\tilde{\psi} = lim_{t\to\infty}\tilde{r} = 0$, and also that the controller will work in the presence of current. The transfer function for the Nomoto system is given by

$$h_n(s) = \frac{K}{Ts+1} \quad (5)$$

and the transfer function for the PID controller is given by

$$h_r(s) = \frac{K_d s^2 + K_p s + K_i}{s} \quad (6)$$

If we assume that the heading error $\tilde{\psi_e}$ is caused by a change in the reference heading $\tilde{\psi_r}$ and also the current $\tilde{\psi_c}$, which is assumed to be constant or slowly varying, we have

$$\tilde{\psi_e} = \tilde{\psi_r} + \tilde{\psi_c} \quad (7)$$

We can then express the heading error caused by the reference as

$$\tilde{\psi_r}(s) = \frac{h_r(s)h_n(s)\frac{1}{s}}{1 - h_r(s)h_n(s)\frac{1}{s}}\tilde{\psi_d}(s) \quad (8)$$

where we assume that the heading change is a sine wave and can be modeled as $\tilde{\psi_d}(s) = \psi_d\frac{a}{s+a^2}$ Applying the finale value theorem gives

$$lim_{s\to 0}s\tilde{\psi_r}(s) = 0 \quad (9)$$

We can express the heading error caused by the current as

$$\tilde{\psi_c} = \frac{\frac{1}{s}}{1 - h_r(s)h_n(s)\frac{1}{s}}c(s) \quad (10)$$

where we assume that the current c can be modeled as a step $c(s) = \frac{c}{s}$ Applying the finale value theorem gives

$$lim_{s\to 0}s\tilde{\psi_c}(s) = 0 \quad (11)$$

Thus we have

$$lim_{s\to 0}s\tilde{\psi}_e(s) = lim_{s\to 0}s\tilde{\psi}_r(s) + lim_{s\to 0}s\tilde{\psi}_c(s) = 0 \tag{12}$$

and

$$lim_{s\to 0}s\tilde{r}_e(s) = lim_{s\to 0}s^2\tilde{\psi}_e(s) = 0 \tag{13}$$

We have then proved that the PID controller will be able to track a sine wave based reference signal when the current is constant or slowly varying without having a steady state offset.

Task 1.4

We first look at the PD part of the controller. We have the model

$$T\dot{r} + r = K\delta_c \tag{14}$$

$$T\ddot{\psi} + \dot{\psi} = K(-K_p\tilde{\psi} - K_d\dot{\tilde{r}}) = K(-K_p(\psi - \psi_d) - K_d(\dot{\psi} - r_d)) \tag{15}$$

$$T\ddot{\psi} + (1 + KK_d)\dot{\psi} + KK_p\psi = K(K_p\psi_d + K_d\dot{\psi}_d) \tag{16}$$

Meaning that for the characteristic polynomial we have

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = s^2 + \frac{1 + KK_d}{T}s + \frac{KK_p}{T} \tag{17}$$

which gives

$$\omega_n = \sqrt{\frac{KK_p}{T}} \tag{18}$$

$$\zeta = \frac{1 + KK_d}{2\sqrt{KK_pT}} \tag{19}$$

Since we want the controller to be faster than the input dynamics, which is a sine where $\omega_d = 0.008$, we choose $\omega_n = 10\omega_d = 0.08$. By also choosing critical damping, meaning $\zeta = 1.0$, we can solve the equation set and find K_p and K_d.

$$K_p = \frac{\omega_n^2 T}{K} = -14.48 \tag{20}$$

$$K_d = \frac{2\zeta\omega_n T - 1}{K} = -323.69 \tag{21}$$

and by a rule of thumb we choose

$$K_i = \frac{\omega_n}{10} K_p = -0.12 \tag{22}$$

The plots that show the closed-loop behavior is shown in figure 4, 5, 6 and 7.

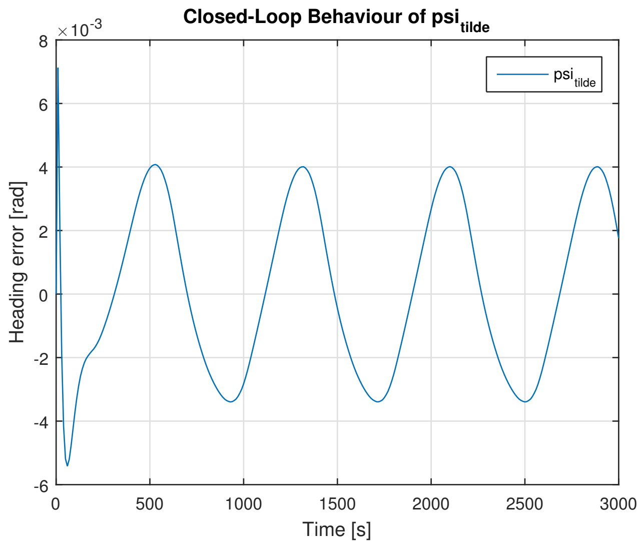

Figure 4: Closed-loop behavior of $\tilde{\psi}$

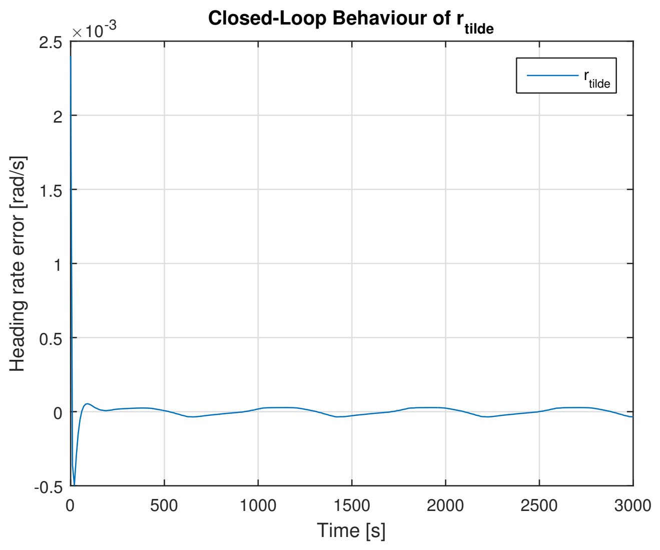

Figure 5: Closed-loop behavior of $\tilde{r}$

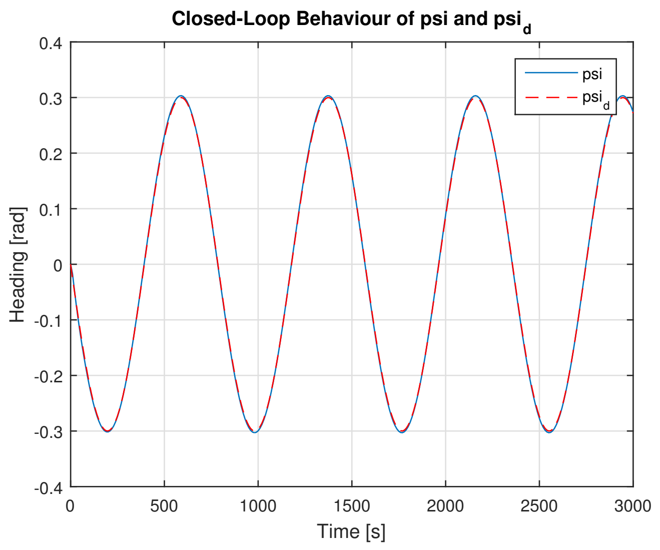

Figure 6: Closed-loop behavior of ψ and ψ_d

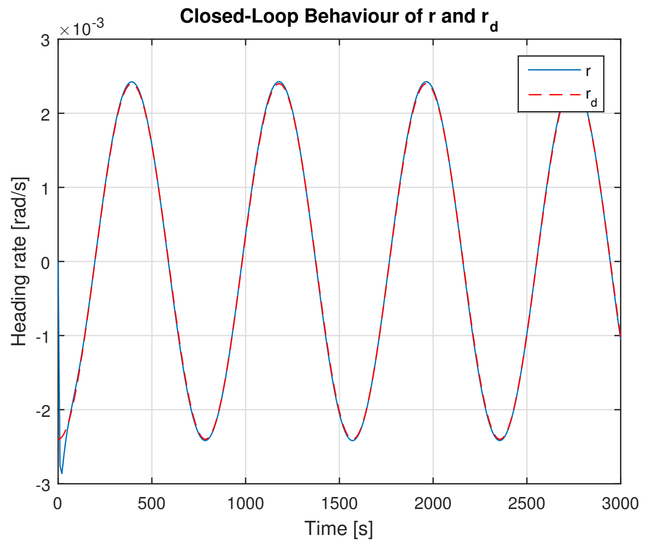

Figure 7: Closed-loop behavior of r and r_d

1.2 Speed Autopilot

Task 1.5

If we assume that the craft is moving at a constant or slowly varying forward speed, we can use the forward speed model

$$(m - X_{\dot{u}})\dot{u} - X_{\dot{u}}u_r - X_{|u|u}|u_r|u_r = \tau_1 \tag{23}$$

We can use this model because we have what is called starboard-port symmetry, which implies that surge is decoupled from sway and yaw.

Task 1.6

We reduce the decoupled model from Task 1.5 to

$$m_u\dot{u} + d_1u + d_2|u|u = n_c|n_c| \tag{24}$$

where

$$m_u = \frac{m - X_{\dot{u}}}{K} \tag{25}$$

$$d_1 = -\frac{X_u}{K} \tag{26}$$

$$d_2 = -\frac{X_{|u|u}}{K} \tag{27}$$

where we have used (12.265) from [1] to model the thrust from the variable-speed FP propeller as

$$F(n) = Kn|n| \tag{28}$$

where K is a constant thrust coefficient and n is the propeller rpm. Using this, we can then simulate the system with two different values of n_c and find the speed of the craft in each scenario. Since then the speed is constant, the acceleration is zero and we have two equations with two unknowns.

$$d_1 u + d_2 |u| u = \tau_1 \tag{29}$$

$$d_1 u + d_2 |u| u = \tau_2 \tag{30}$$

where τ_1 and τ_2 is given by $\tau = n_c|n_c|$ where n_c is the propeller rpm (in rad/s) chosen by us for the simulation.

We find

$$d_1 = -0.0634 \tag{31}$$

$$d_2 = 1.0463 \tag{32}$$

To find m_u we looked at the step responses of the Forward Speed model and the full ship model and compared. We found

$$m_u = 6000 \tag{33}$$

to be a good value. The step response comparison between the full ship model and the Forward Speed model is shown in figure 8.

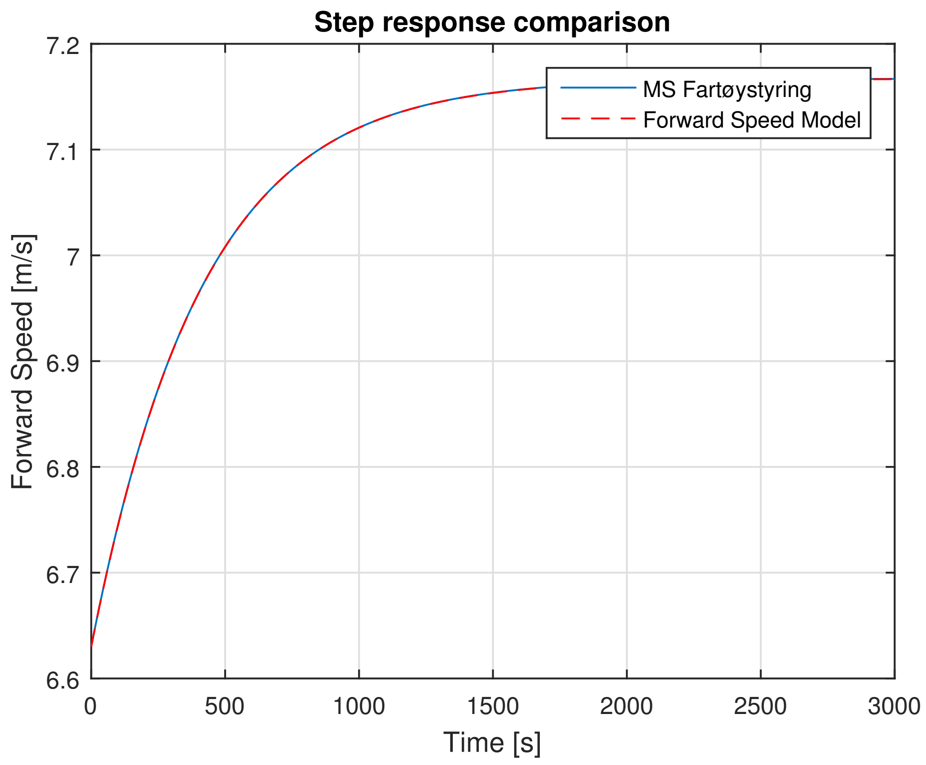

Figure 8: Step response comparison between the full ship model and the forward speed model

Task 1.7

We define

$$\tilde{u} = u - u_d \tag{34}$$

From section 13.2.1 in [1] we know that the acceleration can be calculated as

$$\dot{u} = \dot{u}_d - K_p(u - u_d) - K_i \int_0^t (u - u_d)(\tau) d\tau = \dot{u}_d - K_p\tilde{u} - K_i \int_0^t \tilde{u} d\tau \tag{35}$$

By using this in our model from task 1.5 and 1.6, we get the following speed controller

$$m_u[\dot{u}_d - K_p\tilde{u} - K_i \int_0^t \tilde{u}(\tau)d\tau] + d_1 u + d_2|u|u = \tau_1 \tag{36}$$

which is a feedback linearizing speed PI controller with feedforward acceleration. We then know that the linear system's equilibrium point given by

$$\dot{\tilde{u}} + K_p\tilde{u} + K_i\int_0^t \tilde{u}d\tau = 0 \tag{37}$$

is globally exponentially stable if the gains are chosen as

$$K_p = 2\lambda \tag{38}$$

$$K_i = \lambda^2 \tag{39}$$

$(\lambda > 0)$, ensuring that $lim_{t\to\infty}\tilde{u} = 0$

Using a third order reference model to smoothen out the reference signal u_d found in [1, p. 250], we get

$$\frac{u_d}{u_r}(s) = \frac{\omega_{n,speed}^3}{s^3 + (2\zeta_{speed}+1)\omega_{n,speed}s^2 + (2\zeta+1)\omega_{n,speed}^2 s + \omega_{n,speed}^3} \tag{40}$$

This prevents oscillations in speed while converging to the reference value.

Task 1.8

By trial and error we found

$$\lambda = 0.23 \tag{41}$$

$$\omega_{n,speed} = 0.01 \tag{42}$$

to be good values for the speed controller and reference model. Also choosing

$$\zeta_{speed} = 1.0 \tag{43}$$

to get critical damping gave good results. This means that the reference model is tuned to be a bit slower than the ship, and should therefore reduce the amount of oscillations in the system. The closed-loop behavior of the system is shown in figure 9 and 10 .

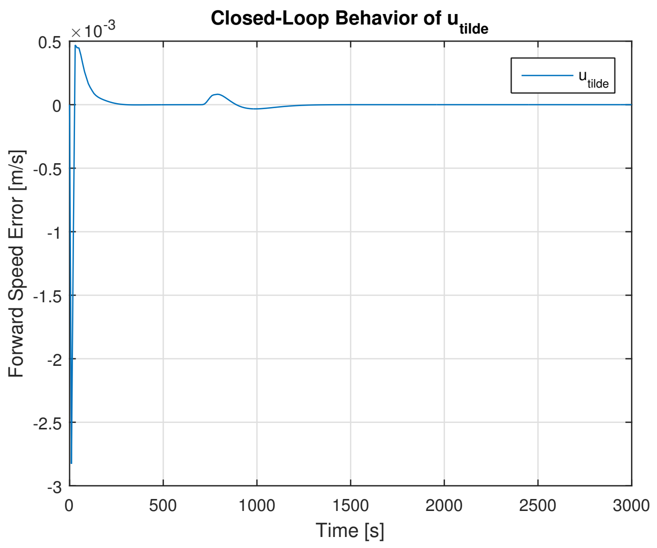

Figure 9: Closed-loop behavior of $\tilde{u}$

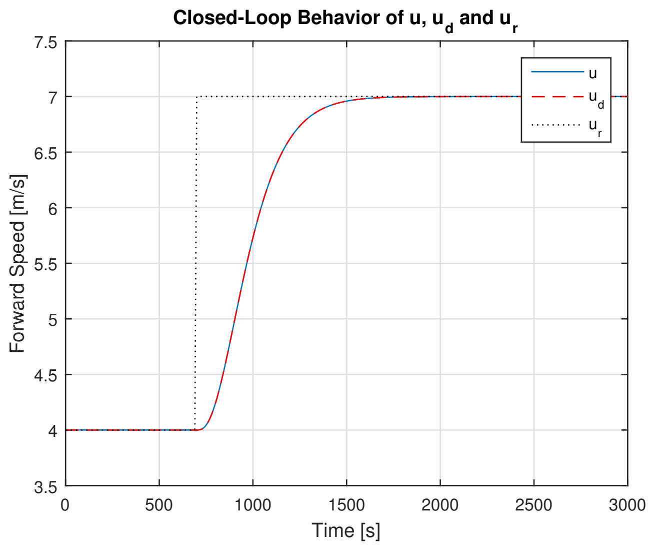

Figure 10: Closed-loop behavior of u and u_d

2 Path Following and Path Tracking

2.1 Path Generation

Task 2.1

For the continuous interpolation we plotted the smooth path for both hermitian interpolation and splines. The piece-wise continuous interpolation was plotted by drawing lines between the way points.

To be able to plot the Dubin's path we needed to know the ship's turning radius. By simulating the ship's model with a constant maximum allowed shaft speed, $n_c = 8.9 rad/s$ and the maximum rudder angle $\delta_c = 25°$ without having disturbances enabled, we found the minimum turning radius to be just under 400 meters. The simulation result is shown in figure 11 . However, since the ship can't get maximum turn radius instantly and to get a smooth turn, we've chosen a turn radius $R = 800m$ for the Dubin's path.

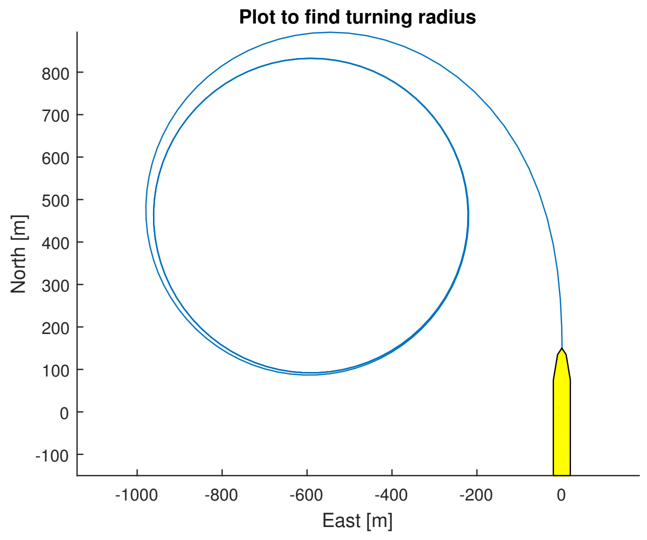

Figure 11: The ship's minimum turning radius with maximum shaft speed and rudder angle

The two continuous interpolation methods both ensures that the ship hits the way points exactly while being smooth and easy to track. The downside to these methods is that they both take a longer route than the other methods.

The piece-wise continuous interpolation method is a short and direct route. However, it is unrealistic to be able to follow this path exactly, especially since our ship has no thrusters, making it unable to turn with low turning radiuses, even at low speed.

Dubin's path is a modification of the continuous interpolation methods. It follows the same direct route as the piece-wise continuous interpolation method until it gets within a certain range of the way point, then it starts turning following the circles, as shown in figure 12. This method represents the shortest path in minimum time, and is the preferred method in most cases. The downsides of Dubin's path are that when turning off the circles and onto the the direct route, a jump in the desired yaw rate is expected. It also cuts the corners, and does not hit the way points exactly. Therefore, if its very important to hit the way points exactly, you are better off with an interpolation method. Otherwise, the Dubin's path is the preferred method.

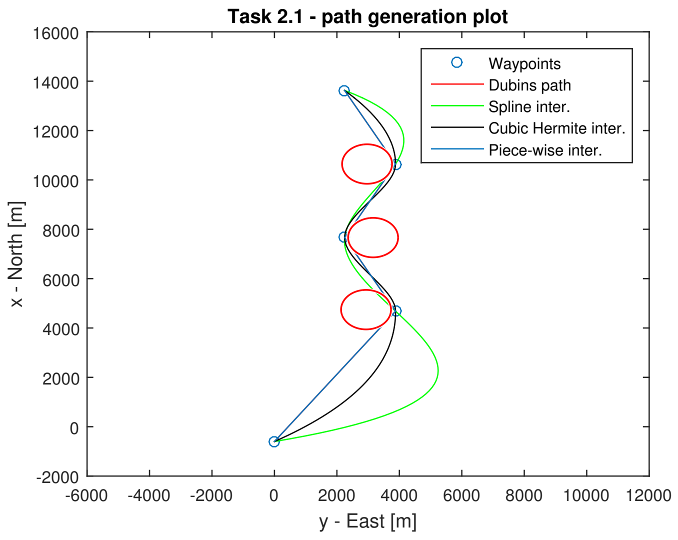

Figure 12: Path generation methods

2.2 Path Following

Task 2.2

We are to design a guidance system that makes the ship follow a piece-wise linear path. We do so using Line-of-sight steering laws based on lookahead-based steering, which is a computationally cheaper method than its rival, the enclosure-based steering method, and is also valid for all cross-track errors.

For lookahead-based steering, the course angle assignment is separated into two parts:

$$\chi_d(e) = \chi_p + \chi_r(e) \tag{44}$$

where

$$\chi_p = \alpha_k = arctan(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}) \tag{45}$$

is the path-tangential angle and

$$\chi_r(e) := arctan(\frac{-e}{\Delta}) = arctan(-K_p e - K_i \int_0^t e(\tau) d\tau) \tag{46}$$

is a velocity-path relative angle, which ensures that the velocity is directed toward a point on the path that is located a lookahead distance $\Delta(t) > 0$ ahead of the direct projection $p^n(t)$ on to the path. The integral part of this controller is there to avoid having sideslip and to make sure the ship follows the exact path. If this isn't necessary, it is possible to just use a P-controller instead, resulting in a faster response.

The cross-track error is found by

$$e(t) = -[x(t) - x_k] sin(\alpha_k) + [y(t) - y_k] cos(\alpha_k) \tag{47}$$

The ship is to move along a piece-wise linear path made up of straight line segments connected by way points. To do so we need a switching mechanism for when to select the next way point. Way point (x_{k+1}, y_{k+1}) can be selected on the basis of whether or not the craft lies within a circle of acceptance with radius R_{k+1} around (x_{k+1}, y_{k+1}). Meaning that if the ship's position (x, y) at a given time t satisfy

$$[x_{k+1} - x(t)]^2 + [y_{k+1} - y(t)]^2 \leq R_{k+1}^2 \tag{48}$$

the next way point should be selected. Choosing R_{k+1} equal to two ship lengths, as suggested in the book, we have $R_{k+1} = 609.6m$.

We have no restrictions on speed, therefore the guidance for speed was set to a constant value, $U_d = 6.63$, which is the same speed that was used in the first part of this assignment, and also the same speed we used to tune our speed controller with.

Task 2.3

Choosing first to go with a P-controller for a faster ship response, we chose the tuning parameters for the resulting guidance controller as

$$K_p(t) = \frac{1}{\Delta(t)} \tag{49}$$

$$K_i = 0 \tag{50}$$

where

$$\Delta(t) = \sqrt{R^2 - e(t)^2} \tag{51}$$

Since our heading controller takes the derivative of the desired heading as input, it would get a step in the desired heading every time we changed way points. To fix this we implemented a simple 2nd order reference model

$$\frac{\psi_d}{\psi_r} = \frac{\omega_{heading}^2}{s^2 + 2\zeta_{heading}\omega_{heading} + \omega_{heading}^2} \tag{52}$$

where we once again chose $\zeta = 1.0$ to have critical damping, and by trial and error found $\omega = 0.05$ to give good results.

While simulating we noticed how both autopilots suffered from integral windup. Both the rudder and shaft speed has a saturation their controller is not aware of. This was especially noticeable in the speed controller, since it was not able to keep the desired surge speed while turning, causing large integral windup. We fixed this by adding back-calculation to both controllers, and thus improved performance.

The ship's path is shown in figure 13, and the ship's closed loop behavior is shown in Figure 14, 15, 16, 17, and 18.

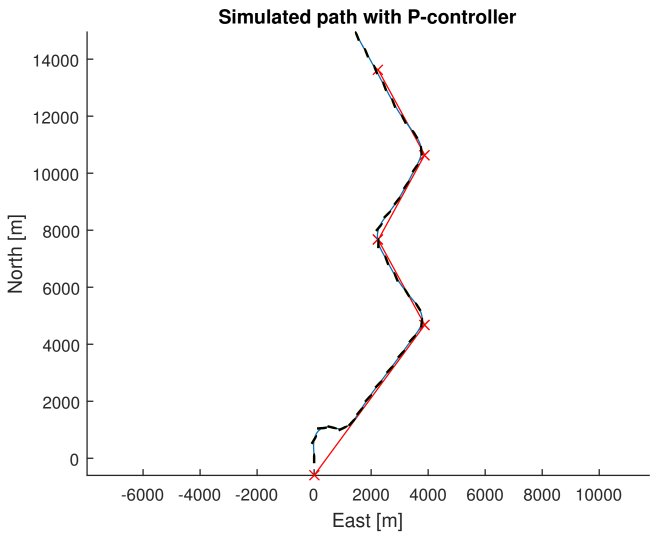

Figure 13: The ship's simulated path with a P-controller guidance system

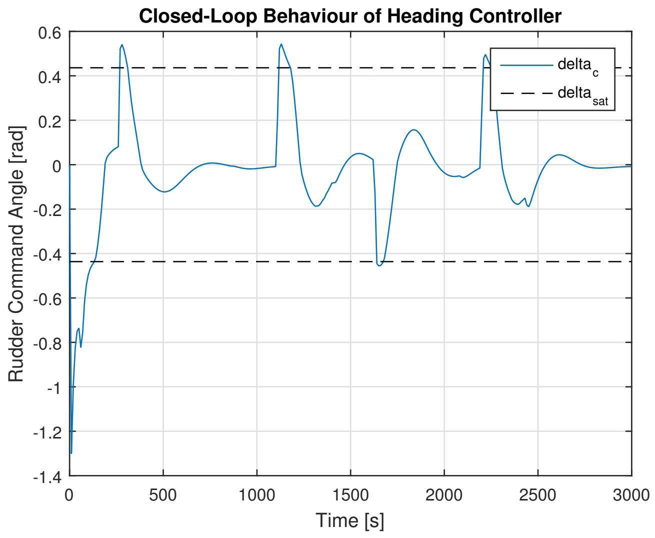

Figure 14: The ship's simulated rudder command angle with a P-controller guidance system

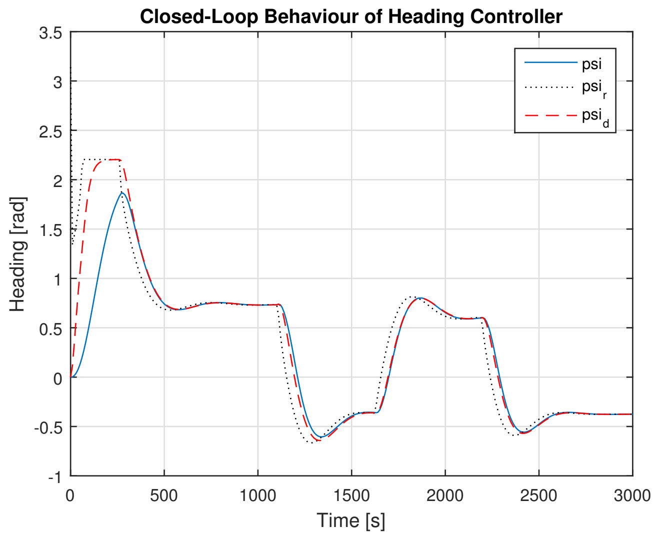

Figure 15: The ship's simulated heading with a P-controller guidance system

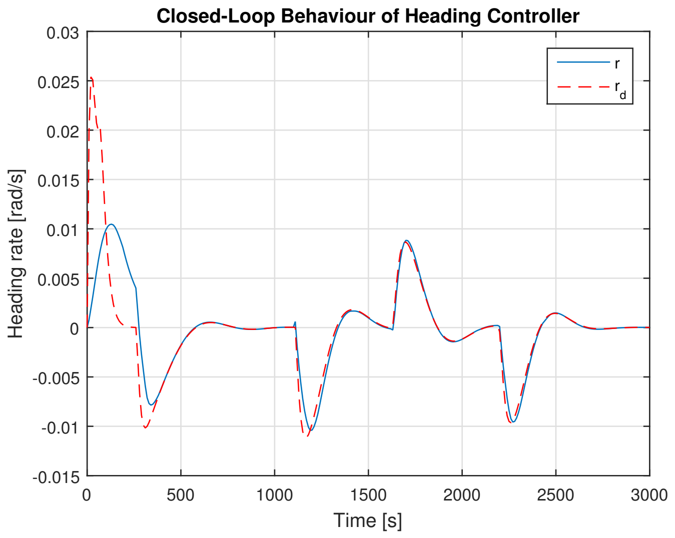

Figure 16: The ship's simulated heading rate with a P-controller guidance system

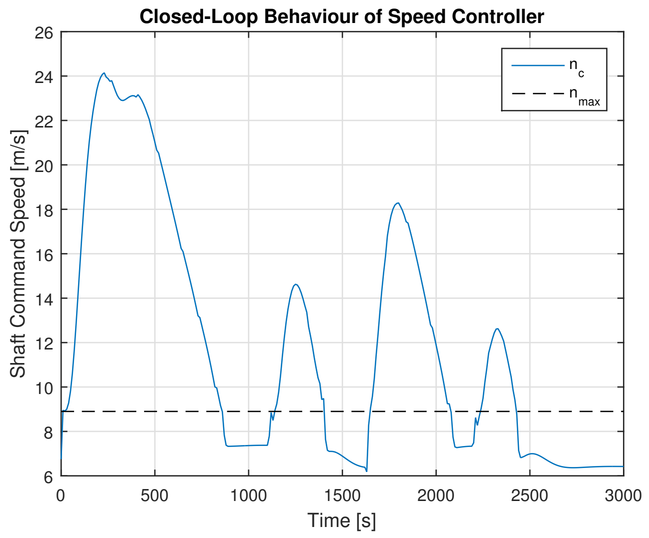

Figure 17: The ship's simulated shaft command speed with a P-controller guidance system

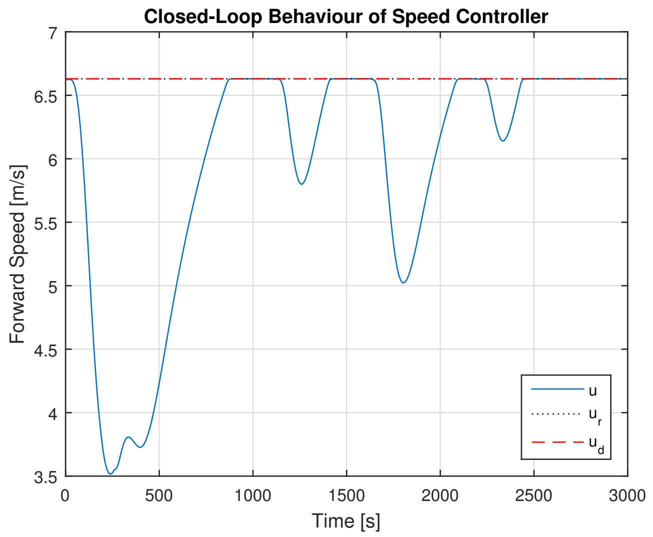

Figure 18: The ship's simulated forward speed with a P-controller guidance system

Now extending our system to include integral action in the guidance system, we get the following tuning parameters for our guidance controller:

$$K_p(t) = \frac{1}{\Delta(t)} \tag{53}$$

$$K_i(t) = \frac{1}{300} K_p(t) \tag{54}$$

where $K_i(t)$ was chosen using trial and error. The ship's new path is shown in figure 19, and the closed loop behavior in figure 20, 21, 22, 23 and 24.

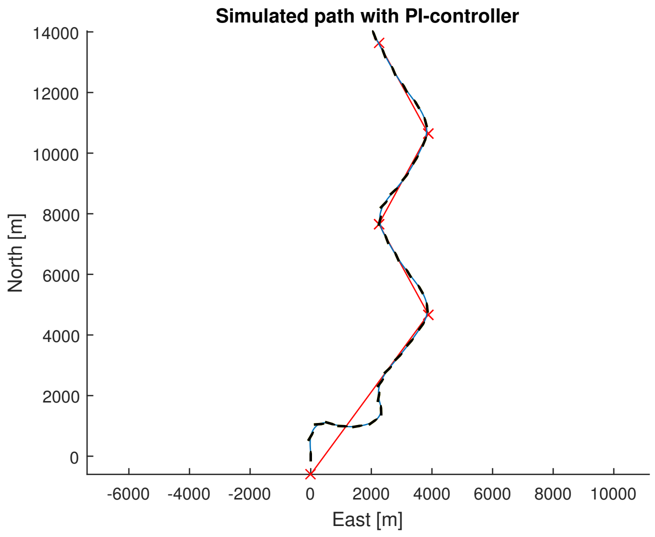

Figure 19: The ship's simulated path with a PI-controller guidance system

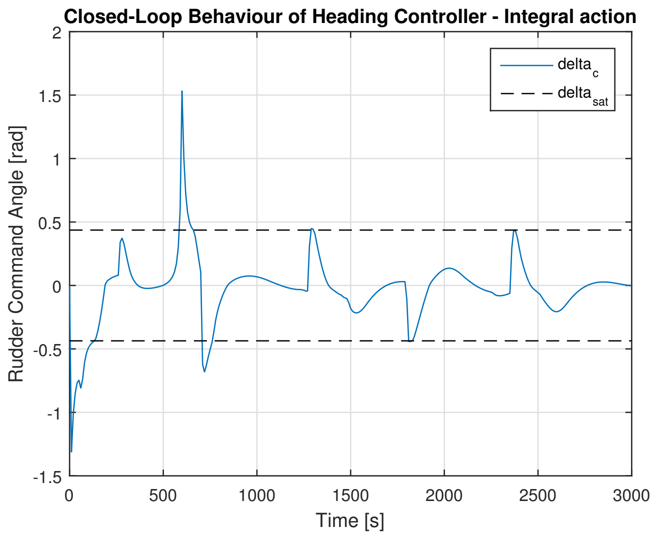

Figure 20: The ship's simulated rudder command angle with a PI-controller guidance system

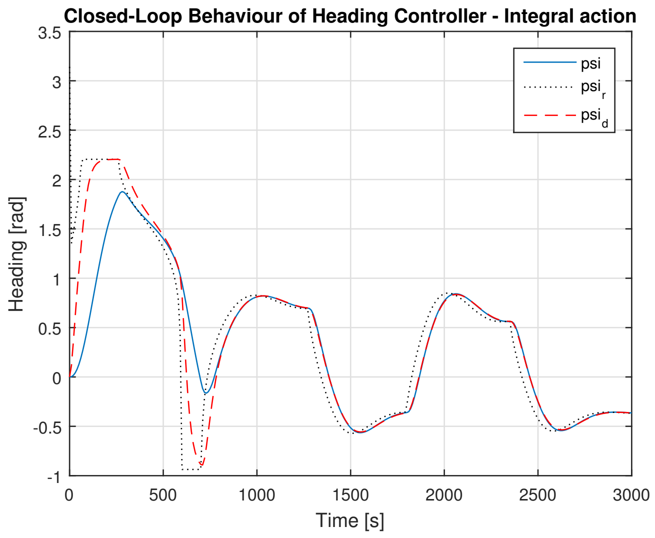

Figure 21: The ship's simulated heading with a PI-controller guidance system

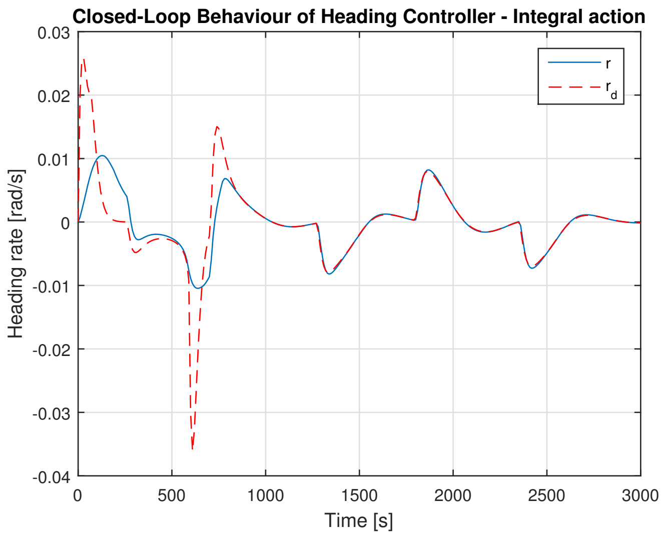

Figure 22: The ship's simulated heading rate with a PI-controller guidance system

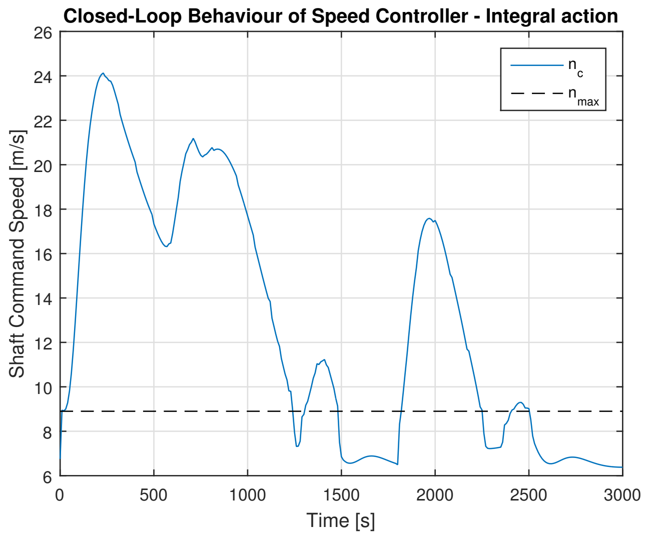

Figure 23: The ship's simulated shaft command speed with a PI-controller guidance system

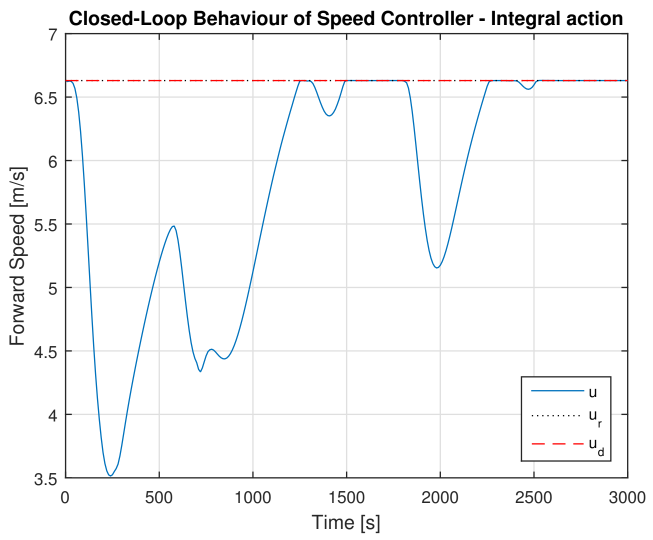

Figure 24: The ship's simulated forward speed with a PI-controller guidance system

Task 2.4

The ship's path-following behavior observed was more or less as expected. Using a guidance system based on a P-controller, we got a fast response but some deviation from the desired path, meaning we had a non-zero sideslip angle. This is because the desired course X_d is mapped directly to the heading reference ψ_r. Introducing integral action to the system somewhat fixed this, but the response got slower and more shaky, making the ship overshoot way points and use more time and potentially energy to find the correct path. Even compensating for integral windup in both controllers didn't fix this, and overall the ship's performance looked better without integral action in the guidance system. We therefore chose to continue with our P-controller based guidance system for the following tasks.

Task 2.5

We use transformations to improve the accuracy of ψ_r and u_r. To transform the course angle χ_d to ψ_r we use

$$\psi_r = \chi_d - \beta \tag{55}$$

where

$$\beta = asin(\frac{v}{U}) \tag{56}$$

where

$$U = \sqrt{u^2 + v^2} \tag{57}$$

From (57) we also get the transformation for u_r:

$$u_r = \sqrt{U^2 - v^2} \tag{58}$$

These transformations will help our system reduce the stationary error we had while using a P-controller based guidance system, and also help reduce the oscillations we had while using a PI-controller based guidance system.

Task 2.6

When simulating we noticed how our 3rd order speed reference model was too slow to follow the new u_r provided by the transformation block, which was more rapidly changing than before. By swapping to a 2nd order model given by

$$\frac{u_d}{u_r}(s) = \frac{\omega_{speed}^2}{s^2 + 2\zeta_{speed}\omega_{speed}s + \omega_{speed}^2} \tag{59}$$

which is the same reference model used for heading. Since u_r now is rapidly changing we had to increase ω_{speed} so that the reference model could keep up. We kept $\zeta_{speed} = 1$ to keep the critical damping, and, by trial and error, found $\omega_{speed} = 0.5$ to be a suiting value which made it possible to follow u_r a lot better.

The ship's path is shown in figure 25, and the ship's closed loop behavior is shown in figure 26, 27, 28, 29 and 30. We can see how the performance has improved, and the path following behavior is better than in previous tasks.

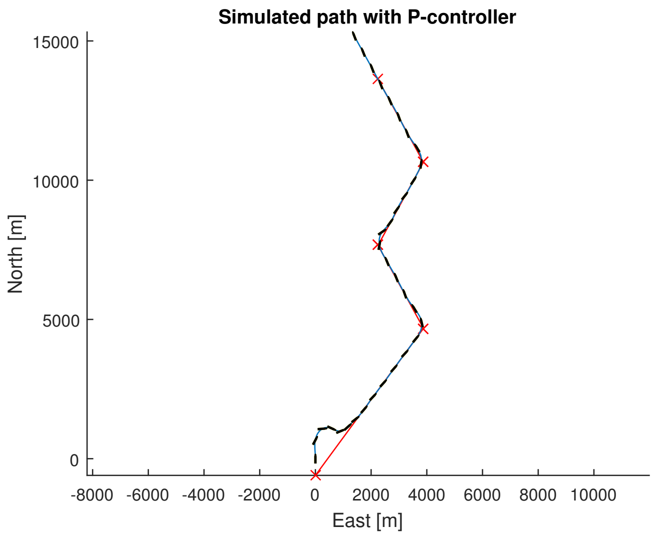

Figure 25: The ship's simulated path with a P-controller guidance system and transformations

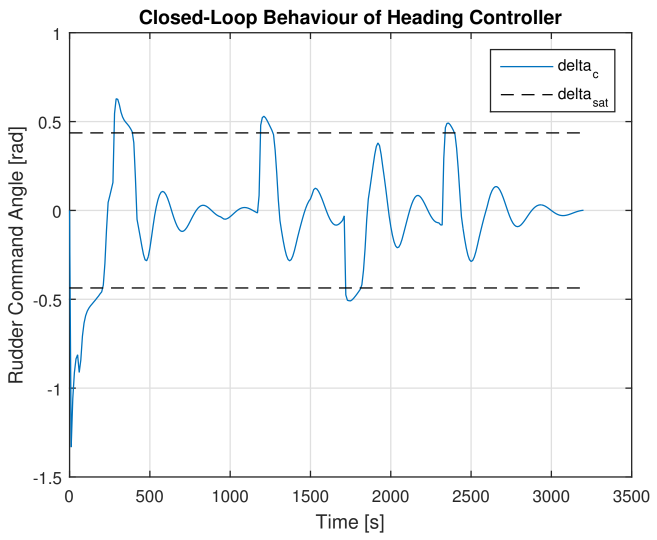

Figure 26: The ship's simulated rudder command angle with a P-controller guidance system and transformations

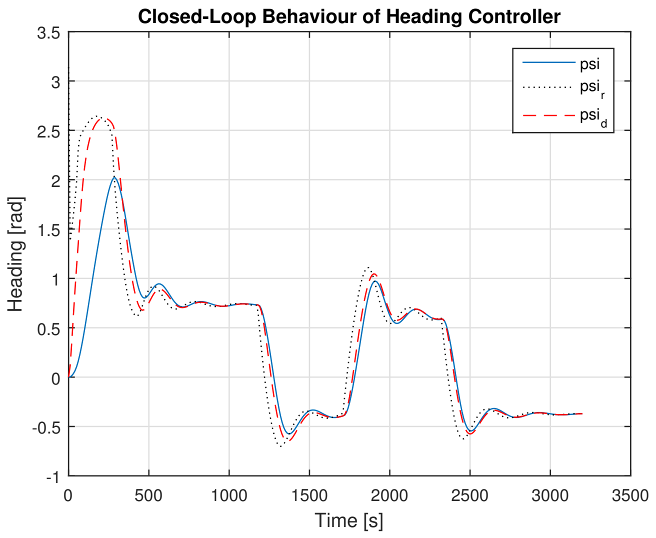

Figure 27: The ship's simulated heading with a P-controller guidance system and transformations

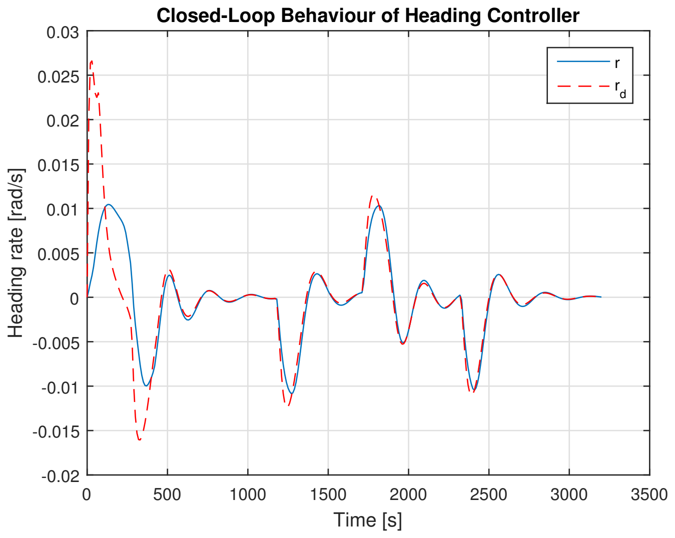

Figure 28: The ship's simulated heading rate with a P-controller guidance system and transformations

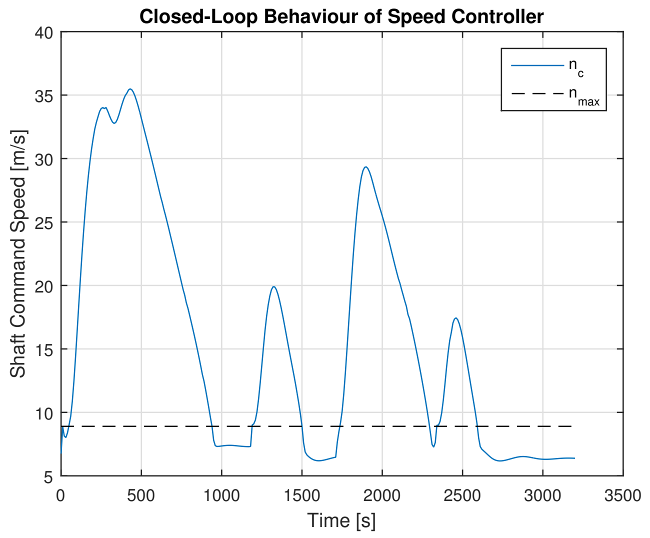

Figure 29: The ship's simulated shaft command speed with a P-controller guidance system and transformations

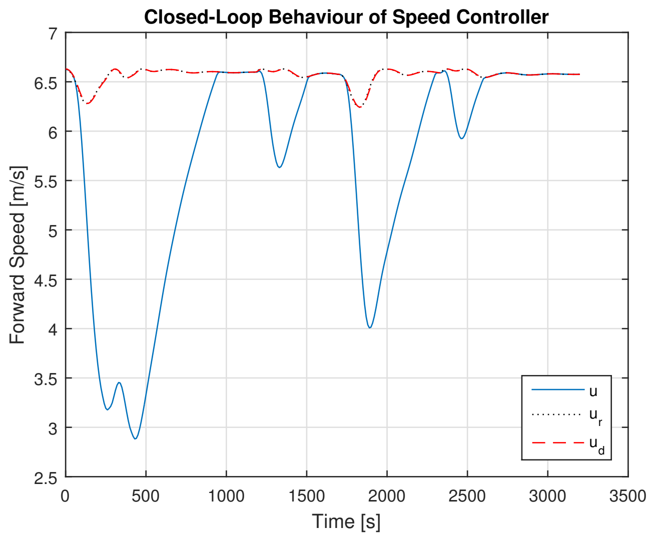

Figure 30: The ship's simulated forward speed with a P-controller guidance system and transformations

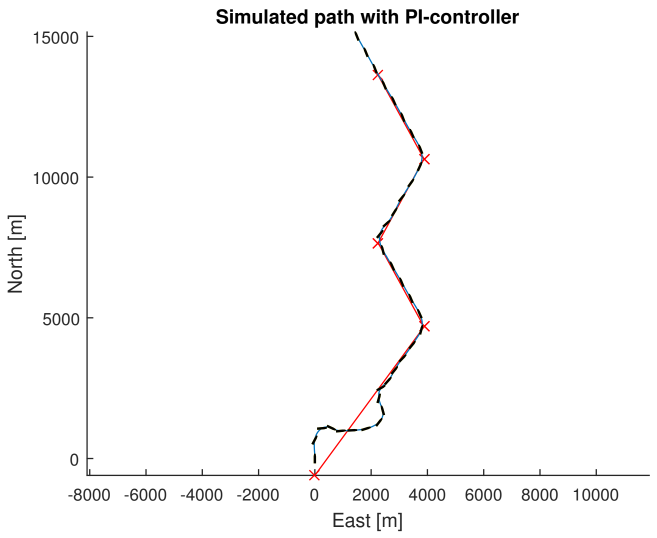

Figure 31: The ship's simulated path with a PI-controller guidance system and transformations

The ship's path using a PI-controller based guidance system with transformations is shown in figure 31, and we can also see how the performance have increased compared to PI-controller based guidance system in task 2.3. However, the P-controller based guidance system still has better performance, making our choice of using the P-controller based guidance system in task 2.3 the correct choice.

2.3 Path Tracking

Task 2.7

We are to track a target moving with constant speed $U_t = 3m/s$ along a straight line passing through the first two way points from the previous tasks. We want to catch up to the moving ship, follow it's heading and stay at a constant distance from it, namely $s_d = 100m$. To achieve this, we designed a speed guidance controller with feedback:

$$U_d = U_t - \kappa \frac{s}{\sqrt{s^2 + \Delta_s^2}} \tag{60}$$

where κ is given by

$$\kappa = U_{a,max} = u_{max} - U_t \tag{61}$$

and s is given by

$$s = \begin{bmatrix} cos(\chi_t) \\ sin(\chi_t) \end{bmatrix} (p^n - p_t^n) + s_d \tag{62}$$

where s_d is the desired tracking distance from the target, χ_t is the heading of the target, and $\Delta_s > 0$ is a speed tuning parameter which decides the behavior of the speed while the ship approaches the tracked target. Low values cause oscillations in U_d, while high values reduce oscillations but increases the approach time. Choosing $\Delta_s = 2000$ gave some oscillations, but by increasing ω_n in the heading controller to $\omega_n = 0.11$ solved this problem.

The path of the ship during tracking is shown in figure 32 and the total distance to target during tracking is shown in figure 34. The closed loop behavior can be seen in figure 35, 36, 37, 38 and 39.

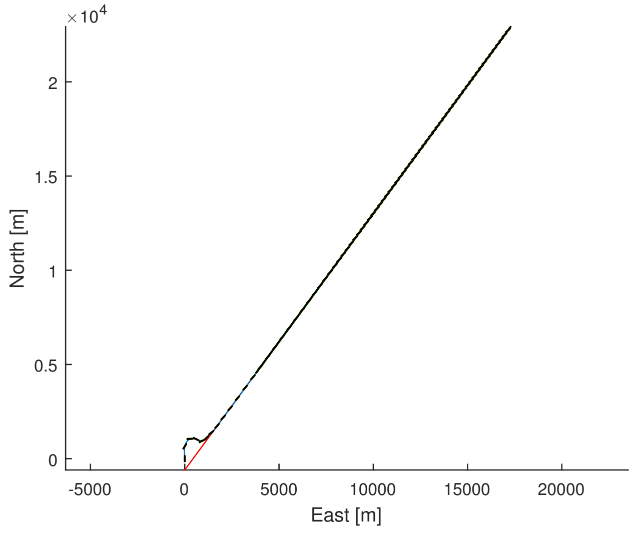

Figure 32: Path of ship during tracking

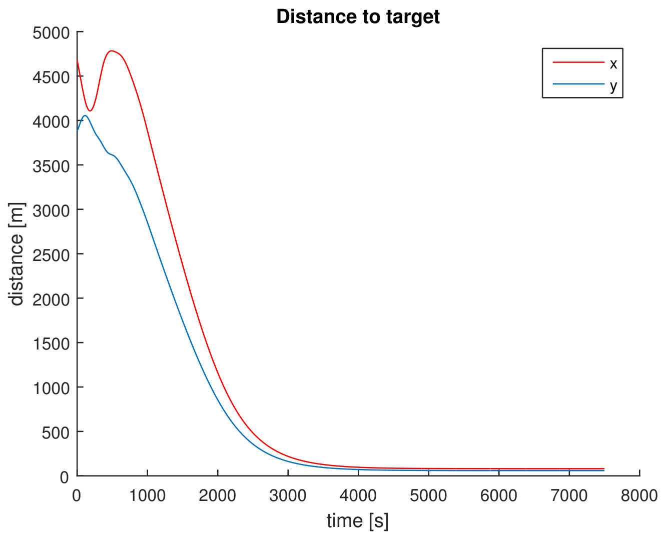

Figure 33: XY distance to target during tracking

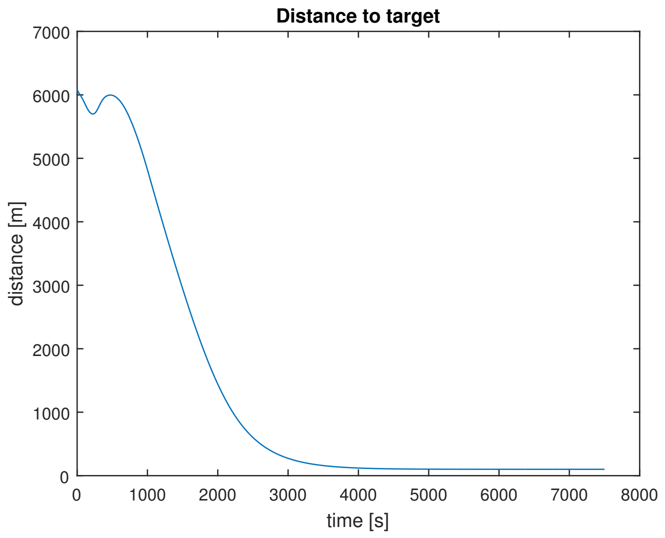

Figure 34: Total distance to target during tracking

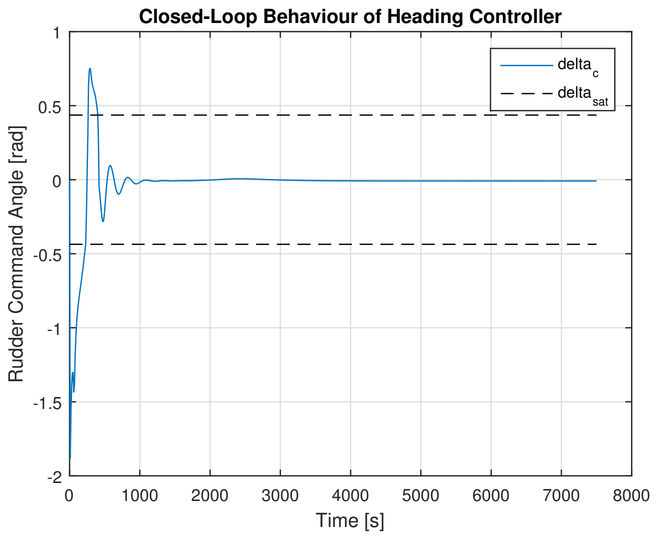

Figure 35: The ship's simulated rudder command angle during tracking

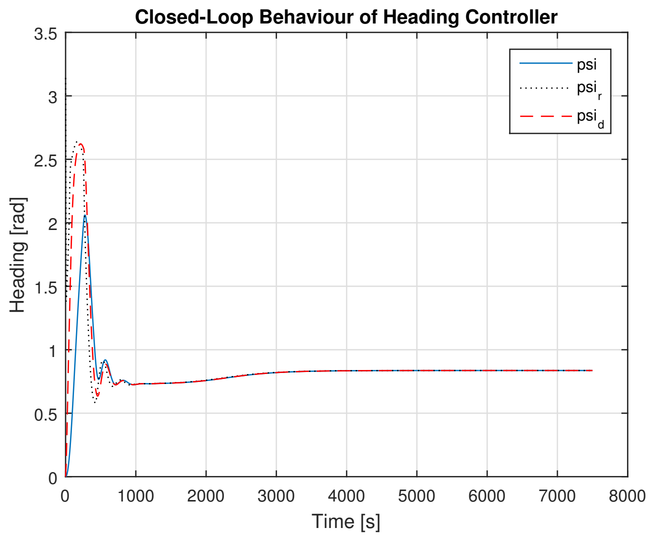

Figure 36: The ship's simulated heading during tracking

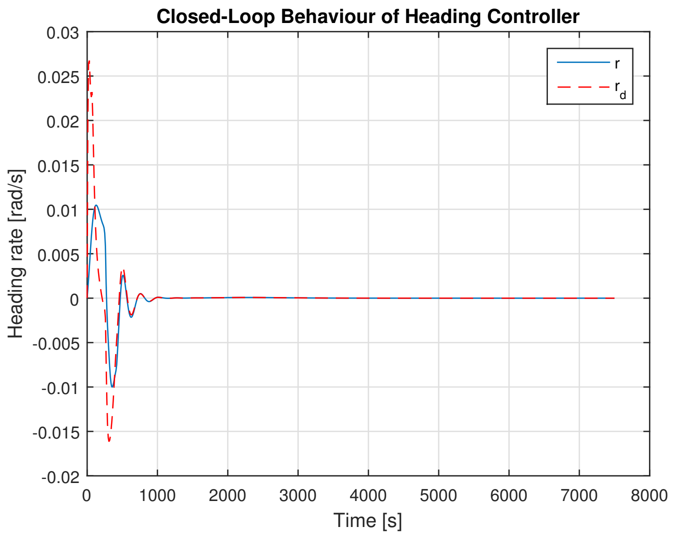

Figure 37: The ship's simulated heading rate during tracking

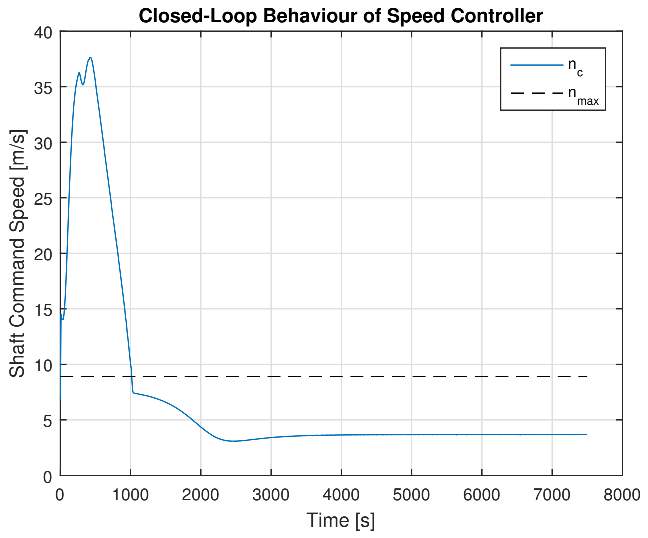

Figure 38: The ship's simulated shaft command speed during tracking

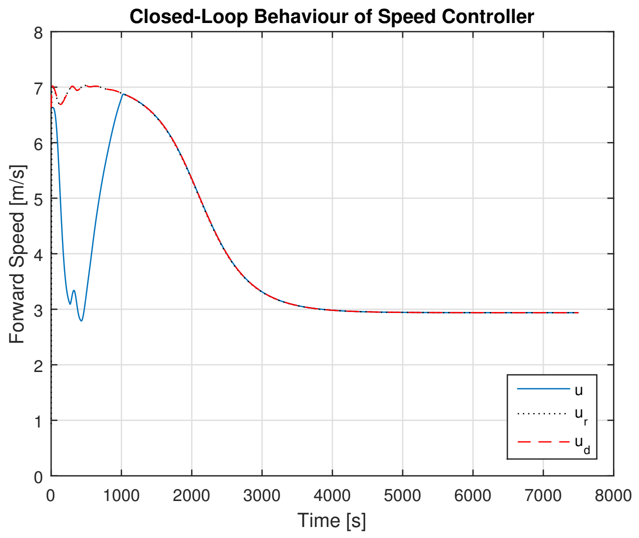

Figure 39: The ship's simulated forward speed during tracking

References

[1] Thor I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, Wiley, 2011.

[2] P. Mishra, "Ships steering autopilot design by Nomoto model", *ELK Asia Pacific Journals*, Special Issue, [online], Available at: http://www.elkjournals.com/microadmin/UploadFolder/41014.M-4.pdf (Accessed: 13.11.2015)