Assignment 3 TTK4190
Autopilots and Path Following/Tracking

Fall 2015

Objective

The main challenges of this assignment are to develop heading and speed autopilots for the ship MS *Fartøystyring*, and subsequently use those autopilots to perform various path-following tasks. The simulations you will produce will have the greatest weight on the grade.

Deadline and Delivery details

The assignment must be handed in by **19:00 on Tuesday, 17 November**. Late submissions will not be accepted. You have to work in groups, and each group must deliver only one copy of the answer.

Preliminaries

The MSS Toolbox (available from the course website) is required for this assignment. Code should be handed in annoted and ready to run as-is, i.e. I don't have to modify any part of it to obtain the desired results and plots. This assignment counts for 10 % of the final grade.

Introduction

The MS *Fartøystyring* is restricted to movement in the plane, and is equipped with a main propeller and a main rudder. Position, heading, and linear and angular velocities are assumed perfectly measured. The ship is subjected to a constant (but unknown) current c. The input-output structure can be seen in Figure 1, where δ_c is the commanded rudder angle and n_c is the commanded shaft speed. The relationships between the commanded inputs and the true inputs are given by $\delta = \delta_{\max}\text{sat}(\delta_c/\delta_{\max})$ and $n = n_{\max}\text{sat}(n_c/n_{\max})$, where

$$\text{sat}(x) = \begin{cases} 1 & \forall \quad x \geq 1 \\ x & \forall \quad x \in [-1, 1] \\ -1 & \forall \quad x \leq -1 \end{cases}$$

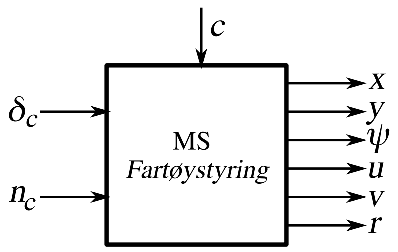

Figure 1: Input-output structure of the ship.

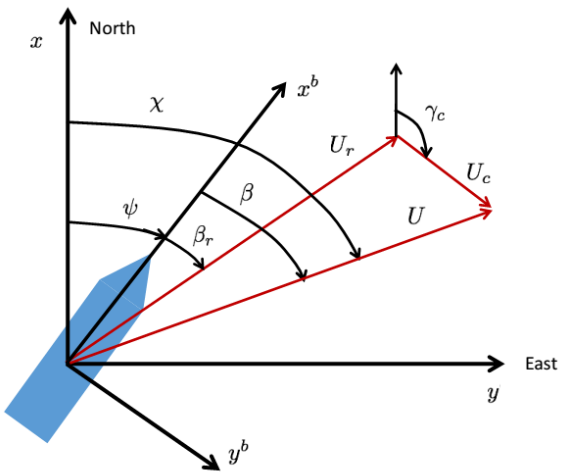

Figure 2: Variable definitions.

The ship position is given by $\mathbf{p} = [x, y]^{\mathrm{T}}$, represented in NED, with the linear velocity $\mathbf{v} = [u, v]^{\mathrm{T}}$, represented in body. The heading (yaw angle) is represented by ψ, with angular speed (yaw rate) represented by r. The direction of the linear velocity $\mathbf{v}$ is represented by the course χ, while the size of $\mathbf{v}$ (speed) is defined by $U \triangleq \sqrt{u^2 + v^2}$. In the absence of currents, the angle $\beta \triangleq \chi - \psi$ is called crab angle, while in the presence of currents the angle $\beta_r \triangleq \chi - \psi$ is called sideslip angle. The current velocity is identified by U_c, its direction by the angle γ_c, and U_r is the relative velocty. See Figure 2 for an illustration of these variables.

Simulink code similar to the block shown in Figure 1 is included with this assignment. The internal workings of the block and the exact direction and strength of c is hidden, and the relationships between inputs and outputs are nonlinear. The current can be switched on and off. Base your code on the supplied code.

1 Autopilot Design

In order to design a controller, a model of the involved process is usually helpful as an aid at a preliminary design stage. There are several different models that can be considered, both linear and nonlinear. Models for use in designing heading autopilots are described in detail in Chapter 7 of the textbook.

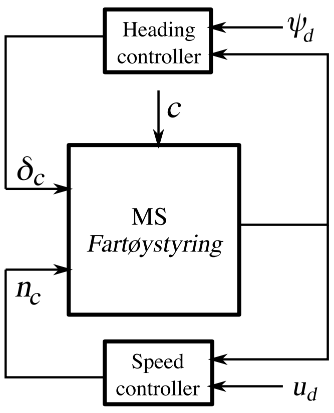

Figure 3: Controller structure for Section 1.

1.1 Heading Autopilot

Task 1.1. *If you were to control the MS* Fartøystyring*'s heading, what kind of model would you use? Justify your answer, and write down the model structure.*

Task 1.2. *By employing δ_c and recording the corresponding ψ and r, identify the parameters of the model you chose in Task 1.1. Write down the parameters, and describe the method you used to find them. (Hints: You may switch off the current for this task. Note that a negative δ_c gives a positive r, opposite of what is assumed for, e.g., the Nomoto model.)*

We define $\tilde{\psi} \triangleq \psi - \psi_d$ and $\tilde{r} \triangleq r - r_d$, where ψ_d and r_d are desired, time-varying yaw and yaw rate reference signals.

Task 1.3. *Design a control law that ensures* $\lim_{t\to\infty}\tilde{\psi} = \lim_{t\to\infty}\tilde{r} = 0$ *(see Figure 3). The controller must work in the presence of current.*

Justify your choice of controller, and write down the controller structure. Implement the controller in Matlab/Simulink.

Task 1.4. *Consider* $\psi_d(t) = -0.3\sin(0.008t)$ rad *and* $r_d(t) = \dot{\psi}_d(t)$*. Use* $u(0) = 6.63$ m/s*,* $v(0) = \psi(0) = r(0) = 0$ *and* $n_c \equiv 7.3$ rad/s*. Tune the controller until it behaves satisfactorily. Write down the final choice of controller parameters. Provide four plots to show the closed-loop behavior. The plots should be of*

1. $\tilde{\psi}$
2. ψ *and* ψ_d
3. $\tilde{r}$
4. r *and* r_d.

For this task, have the current switched on*.*

Code hand-in: *Hand in a file with the title run_task_1_4.m that will initialize the simulation, run the simulation, and generate the necessary plots. Include any other files necessary to run run_task_1_4.m. Place all Matlab/Simulink files in a folder named "task 1.4".*

1.2 Speed Autopilot

Task 1.5. *If you were to control the MS* Fartøystyring*'s surge speed, what kind of model would you use? Justify your answer, and write down the model structure.*

Task 1.6. *By employing* n_c *and recording the corresponding* u*, identify the parameters of the model you chose in Task 1.5. Write down the parameters, and describe the method you used to find them. (Hint: You may switch off the current for this task.)*

We define $\tilde{u} \triangleq u - u_d$, where u_d is a desired, time-varying surge reference signal.

Task 1.7. *Design a control law that ensures* $\lim_{t\to\infty}\tilde{u} = 0$ *(see Figure 3). The controller must work in the presence of current.*

Justify your choice of controller and write down the controller structure. Implement the controller in Matlab/Simulink.

Task 1.8. *Consider* u_d *as a step from 4 to 7* m/s *at* $t = 700$*. Use* $u(0) = 4$ m/s*,* $v(0) = \psi(0) = r(0) = 0$ *and the controller from Task 1.4 to keep* $\psi \approx 0$*. Tune the control law until it behaves satisfactorily. (Hint: It might* *be advantageous to also include a reference model. If you do, write down the structure and parameters used, and justify your choice of both.)*

Write down the final choice of controller parameters. Provide two plots to show the closed-loop behavior. The plots should be of

1. $\tilde{u}$
2. u *and* u_d.

For this task, have the current switched on.

Code hand-in: *Hand in a file with the title run_task_1_8.m that will initialize the simulation, run the simulation, and generate the necessary plots. Include any other files necessary to run run_task_1_8.m. Place all Matlab/Simulink files in a folder named "task 1.8".*

2 Path Following and Path Tracking

2.1 Path Generation

One of the overall objectives of the control laws is to be employed for the purpose of following a path. Based on the current location of the ship and available path information, a guidance system computes a desired velocity that, if properly tracked, ensures path following for the ship. Useful guidance laws for planar path following can be found in the textbook.

Included with the assignment is a set of waypoints, WP. These waypoints can be considered as a sampled selection from an infinite amount of possible paths. Here, you will consider three basic ways of path construction based on WP:

1. **Continuous interpolation:** Using for instance hermitian interpolation or splines, the waypoints can be converted to a continuous, smooth path.
2. **Piece-wise continuous interpolation:** By defining a series of straight-line segments between pairs of consecutive waypoints, a piece-wise continuous, non-smooth path can be generated.
3. **Circles and straight lines:** This method is described in Chapter 10.3.1 in the textbook.

Task 2.1. *Using WP, generate paths using each of the three above-mentioned methods. Plot all of them in an xy-plot according to the NED-frame convention, where the x-axis points north (i.e. upwards) and the y-axis points east (i.e. to the right). Plot all three methods in a single figure. For method 3, you do not have to draw the curve described by the circles and lines, the circles and lines are sufficient. Justify your choice of circle size(s).*

Which path would you choose for path-following purposes, and why?

2.2 Path Following

In the following tasks, your goal is to get the MS *Fartøystyring* to follow the piece-wise linear path generated by method 2 in Task 2.1. The ship is starting in the origin, with $u(0) = 6.63$ m/s and $v(0) = \psi(0) = r(0) = 0$. Use the straight line defined by the first two waypoints as the first active path segment.

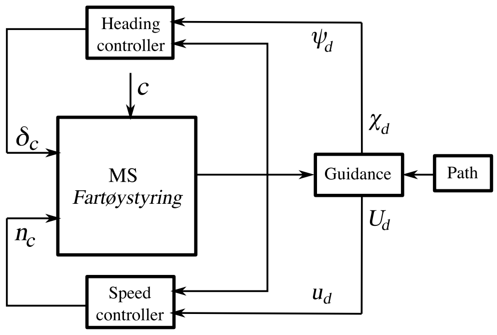

Figure 4: Controller structure for Section 2, Tasks 2.1–2.4.

Task 2.2. *Design a guidance system that employs the ship states and relevant path information associated with method 2 of Task 2.1 as inputs and generates time-varying reference signals χ_d and U_d which kinematically guarantee path following. Implement the guidance system in Matlab/Simulink.*

Task 2.3. *Use the guidance system outputs as reference signals for the heading and speed autopilots, by directly assigning $\psi_d = \chi_d$ and $u_d = U_d$ (see Figure 4). Simulate the system until you think the MS* Fartøystyring *is sufficiently close to the final way-point. Plot the behavior of the closed-loop system with the included Matlab function pathplotter.m. Include both figures in your hand-in.*

For this task, have the current switched on.

Code hand-in: *Hand in a file with the title run_task_2_3.m that will initialize the simulation, run the simulation, and generate the necessary plots. Include any other files necessary to run run_task_2_3.m. Place all Matlab/Simulink files in a folder named "task 2.3".*

Task 2.4. *Is the path-following behavior observed in the previous task as expected? Explain the results.*

Task 2.5. *Design transformations that purposefully convert χ_d to ψ_d and U_d to u_d, see Figure 5. Justify your conversion methods, and implement them in Matlab/Simulink.*

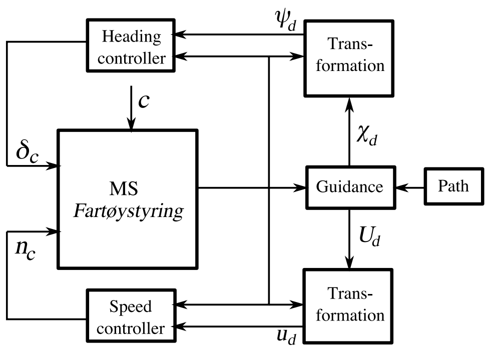

Figure 5: Controller structure for Section 2, Tasks 2.5–2.7.

Task 2.6. *Repeat Tasks 2.3 and 2.4 with ψ_d and u_d as the outputs from the transformation blocks.*

Code hand-in: *Hand in a file with the title run_task_2_6.m that will initialize the simulation, run the simulation, and generate the necessary plots. Include any other files necessary to run run_task_2_6.m. Place all Matlab/Simulink files in a folder named "task 2.6".*

2.3 Path Tracking

A target is moving with constant speed $U_t = 3$ m/s along the straight line passing through the first two waypoints of WP, with initial position at the second waypoint. It is heading away from the first waypoint.

The goal for the following task is to track this target. The ship is starting in the origin, with $u(0) = 6.63$ m/s and $v(0) = \psi(0) = r(0) = 0$.

Task 2.7. *In addition to your path-following system from Task 2.6, include feedback in the speed assignment U_d so that the MS* Fartøystyring *can track the target. Implement and simulate this path-tracking system in Matlab/Simulink from $t = 0$ until you think the ship is sufficiently close to the target, plus a further 200 seconds to show that it remains on target. Make sure you clearly specify the desired distance to target. Plot the behavior of the closed-loop system with the included Matlab function pathplotter.m. Include all figures in your hand-in.*

For this task, have the current switched on.

Code hand-in: *Hand in a file with the title run_task_2_7.m that will initialize the simulation, run the simulation, and generate the necessary plots. Include any other files necessary to run run_task_2_7.m. Place all Matlab/Simulink files in a folder named "task 2.7".*