TMR4243 - Marine Control Systems II

Case study C: Maneuvering

This exercise investigates maneuvering and dynamic positioning (DP) for CS Enterprise I (CSEI).

The case study contains a theory, a simulation and a model scale part. The theory and simulation part will be conducted during the semester and you will be supplied with the necessary equipment to conduct the simulation part of the exercise

System model

Two models are used in this case:

- The kinematics model of the horizontal motion of a vessel,

$$\dot{\eta} = R(\psi)\nu, \tag{1}$$

where $\eta := (p, \psi) \in \mathbb{R}^3$ and $p = (x, y) \in \mathbb{R}^2$ is the vessel position and heading vector, and the body-fixed velocity vector $\nu \in \mathbb{R}^3$ is viewed a control input. The rotation matrix is

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

and note that $\dot{R} = R(\psi)S(r)$ where

$$S(r) = \begin{bmatrix} 0 & -r & 0 \\ r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = -S(r)^\top.$$

- The complete C/S Enterprise I mathematical model in the ship's handbook, Chapter 3.

Problem statement

The maneuvering problem is comprised of the two tasks, in prioritized order:

Geometric task: For some absolutely continuous function $s(t)$, force the output η to converge to the desired parametrized path $\eta_d(s)$, that is,

$$\lim_{t\to\infty} |\eta(t) - \eta_d(s(t))| = 0.$$

Dynamic task: Control $\dot{s}$ to converge to a desired speed assignment $U_s(s, t)$, that is,

$$\lim_{t\to\infty} |\dot{s}(t) - U_s(s(t), t)| = 0.$$

Part I

Theory

1 Task: Path parametrization

In this task we will always assume that the desired heading ψ_d is tangential to the path. (other choices could also be made, but we do not investigate this in this text). In the following (in accordance with the notation of the course) the term $g(\cdot)^{ks}$ means that g is derivated k times respect to s

Task 1.1 Denoting the desired position by $\eta_d = \begin{bmatrix} p_x & p_y & \psi_d \end{bmatrix}^T$, derive expressions for η_d, η_d^s and η_d^{2s} as a function of $p_x, p_y, p_x^s, p_y^s, p_x^{2s}$, p_y^{2s} p_x^{3s} and p_y^{3s}

Task 1.2 Propose a speed assignment $U_s(s,t)$ for $\dot{s}$ that makes the vessel follow the path at constant reference speed U_{ref}.

Using this speed assignment, express $U_s(s,t)$ and $U_s(s,t)^s$ as a function of $p_x, p_y, p_x^s, p_y^s, p_x^{2s}$ and p_y^{2s}.

Task 1.3a Propose a straight-line parametrization for $p_d(s) = (x_d(s), y_d(s))$ with:

- initial position (x_0, y_0), and
- final position (x_1, y_1).

-Express $p_x, p_y, p_x^s, p_y^s, p_x^{2s}$, p_y^{2s} p_x^{3s} and p_y^{3s}

Task 1.3b Propose an ellipsoidal parametrization for $p_d(s) = (x_d(s), y_d(s))$ with:

- center in (c_x, c_y), and
- radii (r_x, r_y).

-Express $p_x, p_y, p_x^s, p_y^s, p_x^{2s}$, p_y^{2s} p_x^{3s} and p_y^{3s}

Hint: *Hint: Use Wolfram Alpha for help. Example: type d/ds ((y(s) /x(s))) to find the derivative of $y(s)/x(s)$ with respect to s*

2 Task: Kinematic model control design

For the kinematic model, let $z_1 := R(\psi)^\top (\eta - \eta_d(s))$ be the position/heading error vector composed in the body-fixed reference frame.

Task 2.1 Differentiate z_1 and show that the CLF

$$V_1(\eta, s) = \frac{1}{2} z_1^\top z_1 = \frac{1}{2} \left(\eta - \eta_d(s)\right)^\top \left(\eta - \eta_d(s)\right) \tag{2}$$

gives

$$\dot{V}_1 = z_1^\top \left(\nu - R(\psi)^\top \eta_d^s(s) \dot{s}\right) \tag{3}$$

Task 2.2 Assuming $\nu = \alpha_1$ is our control law, show that

$$\alpha_1(\eta, s, t) = -K_p z_1 + R(\psi)^\top \eta_d^s(s) U_s\left(s, t\right), \qquad K_p = K_p^\top > 0 \tag{4}$$

results for (2) in

$$\dot{V}_1 \leq -\lambda_{\min}\left(K_p\right) |z_1|^2 - V_1^s\left(\eta, s\right) \left(U_s\left(s, t\right) - \dot{s}\right).$$

Task 2.3 Derive the expression for $V_1^s(\eta, s)$ for the two cases of the desired heading.

3 Task: Update laws

(needs to updates)

Task 3.1 Tracking update law:

1. Show that
$$\dot{s} = U_s\left(s, t\right)$$
solves the Maneuvering Problem.

   Task 3.1b Explain the reason for the name of this update law**.**

Task 3.2 Gradient update law:

1. Show that
$$\dot{s} = U_s\left(s, t\right) - \mu V_1^s\left(\eta, s\right)$$
where $\mu \geq 0$ is a gain, solves the Maneuvering Problem.

   Task 3.2b Explain the reason for the name of this update law**.**

Task 3.3 Modified gradient update law:

To account for the varying magnitude of $V_1^s\left(\eta, s\right)$ along the path, due to a possible nonlinear parametrization, a normalizing modification can be applied.

1. Show that
$$\dot{s} = U_s\left(s, t\right) - \frac{\mu}{|\eta_d^s\left(s\right)|} V_1^s\left(\eta, s\right) \tag{5}$$
where $\mu \geq 0$ is a gain, solves the Maneuvering Problem.

2. Explain mathematically how this normalizes $V_1^s(\eta, s)$ to rather use the unit tangent vector along the path.

Task 3.4 Filtered gradient update law:

Let $\omega_s := U_s(s,t) - \dot{s}$ be the speed assignment error and

$$W_1(\eta, s, \omega_s) := V_1(\eta, s) + \frac{1}{2\lambda\mu}\omega_s^2$$

be a new CLF (considering ω_s as an additional state in the system).

Task 3.5 Show that

$$\begin{aligned} \dot{s} &= U_s(s,t) - \omega_s \\ \dot{\omega}_s &= -\lambda(\omega_s - \mu V_1^s(\eta, s)) \end{aligned}$$

where $\mu > 0$ is a gain, gives

$$\dot{W}_1 \leq -\lambda_{\min}(K_p)|z_1|^2 - \frac{1}{\mu}|\omega_s|^2,$$

and solves the Maneuvering Problem.

Task 3.6 Explain the reason for the name of this update law**.**

4 Task: DP maneuvering control design

In this task we will NOT change the update law (5) to also include the z_2 dynamics, but rather we will keep the update law (5) as above, depending only on (η, s, t). This is achieved if the control law for τ compensates $\dot{\alpha}_1$ directly.

Task 4.1 Let $z_2 = \nu - \alpha_1(\eta, s, t)$ be the error between ν and its desired control α_1, and calculate again $\dot{V}_1$ for (2) with z_2 and α_1 included.

Task 4.2 Calculate $\dot{\alpha}_1$ and $\dot{z}_2$.

Task 4.3 Let $V_2(\nu, \eta, s, t) = V_1(\eta, s, t) + \frac{1}{2} z_2^\top M z_2$ be the Step 2 CLF. Calculate $\dot{V}_2$.

Task 4.4 Propose a control law for τ that renders $\dot{V}_2$ negative definite. Make sure that τ cancels $\dot{\alpha}_1$ directly (and not by $\sigma_1(\eta, s, t) + \dot{\alpha}$), so that the modified gradient update law (5) in previous task can be used as is.

Do not apply integral action directly in the control law. Instead, try to compensate for the unknown bias by using the estimated bias from the DP observer developed in Case B.

Note also that states should be free from noise. Otherwise the term $V_1^s(\eta, s)$ in (5) may make s jump around when μ is large and η is noisy. To avoid this, use the estimated $\hat{\eta}$ from the observer. One can also try to use the filtered gradient update law with cutoff frequency λ set appropriately to attenuate noise.

Hint 1 To simplify, it may be smart to simulate first without the observer and thruster mapping. (i.e. use τ_d directly onto the vessel model and get η and ν, directly from vessel model). Once you are able to do this, take your implementation with outputs from observer and thrust mapping.

Hint 2 You might need to tune your gains. μ should probably be low (<0.1).

Part II

Simulation

5 Task: Hardware-in-the-loop simulation

Task Implement the control law together with the CSEI model.

Task Adjust and compile the control system (without the CSEI control plant model) from Tasks 4 and for use on cRIO. Add input and output ports as necessary.

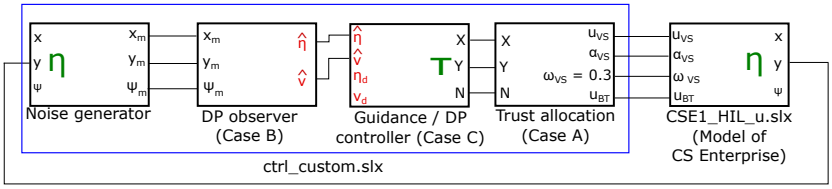

Figure 1: Simulation system structure

Task Set up a VeriStand simulation project connecting your control system to the provided CSEI model.

Task Develop a suitable graphical user interface.

Task 5.1 Simulate straight line maneuvering. Tune μ.

Task 5.2 Simulate ellipsoid maneuvering. Tune μ.

Task Tune controller gains.

Note that in practice in the lab, you start the trajectory from a given initial position (η_0) after a certain amount of time t_0. Therefore, after designing your method in Simulink try the following:

1. See that you can start from any initial position η_0. In Simulink you can test this by changing the initial position of the "fossen-model".
2. See that you can start the trajectory line after a certain amount of time. (i.e. you switch to straight line mode after 10 seconds). In Simulink you can simulate this by using a transport-delay block.

Part III

Scale model test

Task 6.1 Test relevant scenarios corresponding to Task 5. See if you are able to test both straight line and ellipsoid. If not, why ?