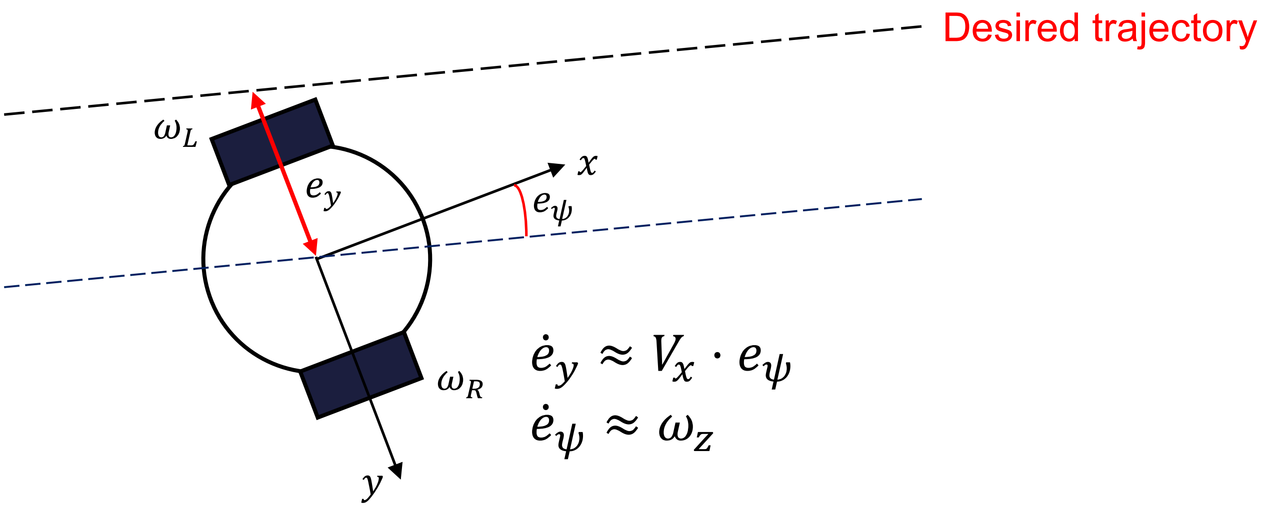
Guideline for the Intern Program

Joohwan Seo
Jan. 2021

Control 개요 - Model

- Mobile robot error-kinematics in body-frame (local) coordinate



$$\dot{e}_y \approx V_x \cdot e_\psi$$
$$\dot{e}_\psi \approx \omega_z$$

- Where V_x is longitudinal velocity and $\omega_z = \frac{R_{wheel}}{R_{robot}}(\omega_R - \omega_L)$.

Control 개요 – Control Derivation

- Note that our control inputs are ω_R and ω_L.

- Since there are no control inputs in our system equation, we take the derivative of $\dot{e}_y$, assuming that V_x is fixed. (Similar to feedback-linearization)

$$\begin{aligned}\ddot{e}_y &= V_x \dot{e}_\psi \\ &= V_x \omega_z\end{aligned}$$

- Take ω_z as
$$\omega_z = \frac{1}{V_x}\left(-K_d(\dot{y} - \dot{y}_r) - K_p(y - y_r)\right)$$

- Then, its closed-loop dynamics become

$$\begin{aligned}&\ddot{e}_y + K_d(\dot{y} - \dot{y}_r) + K_p(y - y_r) \\ &= \ddot{e}_y + K_d(\dot{e}_y) + K_p(e_y) \\ &= 0\end{aligned}$$

- The closed-loop error dynamics become 2nd – order ODE, that you might be very familiar with.

- Also note that ω_z rule is equivalent as the PD control, if we don't have the information of $\dot{y}_r$.

Control 개요 – Problem

1. Assume that we don't want any oscillation in e_y.
   Then, find gain K_p and K_d such that e_y does not oscillate, with the fastest convergence possible.
   (Hint : Select ω_n (natural frequency) on your own. You may need the knowledge of mechanical vibration.)

2. Assume that we have obtained the control gains K_p and K_d which guarantee fastest convergence and no oscillation.
   Then, derive the control signals of ω_R and ω_L.
   (Use any assumption that you might need.)
   (Hint : Do not forget V_x.)

3. Write the pseudo-code for ω_R and ω_L.

4. Assume that we want closed-loop system response with designated pole location. Define the pole location yourself and derive the control signal ω_R and ω_L.

5. Simulate the whole process with Matlab using ODE45.
   (Hint : It is mathematical simulation.)

Simulation Example

- Consider a system

$$\ddot{x} = -x + 2\dot{x} + u$$

- The state variable is $x = x_1, \dot{x} = x_2$, then the state-space equation is

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_1 + 2\dot{x} + u$$

- For the pole location to be (-1,-1), select control input as

$$u = -4\dot{x}$$

Main file

```
clear; close all; clc
%%
init = [4,0]';
[t,X] = ode45(@(t,X) system_fun(t,X), [0,10],init,'.');

figure(1)
plot(t,X(:,1)); hold on; plot(t,X(:,2));
legend('x','dx');
xlabel('time (s)')
```

ODE Function file

```
function dX = system_fun(t,X)
    dX = zeros(2,1);
    x = X(1);
    dx = X(2);

    u = -4*dx;

    dX(1) = dx;
    dX(2) = -x + 2*dx + u;
end
```

See matlab ode45 Function user manual for the details.

System parameter for the problem & Simulation

- $V_x = 0.15$ m/s
  $R_{wheel} = 0.03$ m, $R_{robot} = 0.075$ m

- Initial condition of $y = 0.1, \dot{y} = 0, e_\psi = 0$
- Reference trajectory $y_r = 0.05\cos(t)$

- Use the control signal $u = [\omega_R, \omega_L]^T$
  Simulate for 50 seconds

- Use simulation states $X = \left[y, \dot{y}, e_\psi\right]^T$