

Controllability_and_Observability

November 5, 2025

0.1 Lateral Dynamics

State Matrix (\mathbf{A}_{lat})

$$\mathbf{A}_{lat} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{4C_\alpha}{m\dot{x}} & \frac{4C_\alpha}{m} & -\frac{2C_\alpha(l_f-l_r)}{m\dot{x}} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_\alpha(l_f-l_r)}{I_z\dot{x}} & \frac{2C_\alpha(l_f-l_r)}{I_z} & -\frac{2C_\alpha(l_f^2+l_r^2)}{I_z\dot{x}} \end{bmatrix}$$

Input Matrix (\mathbf{B}_{lat})

$$\mathbf{B}_{lat} = \begin{bmatrix} 0 & 0 \\ \frac{2C_\alpha}{m} & 0 \\ 0 & 0 \\ \frac{2C_\alpha l_f}{I_z} & 0 \end{bmatrix}$$

Controllability matrix (\mathbf{P}_{lat})

$$\mathbf{P}_{lat} = [\mathbf{B}_{lat} \quad \mathbf{A}_{lat}\mathbf{B}_{lat} \quad \mathbf{A}_{lat}^2\mathbf{B}_{lat} \quad \mathbf{A}_{lat}^3\mathbf{B}_{lat}]$$

Observability matrix (\mathbf{Q}_{lat})

$$\mathbf{Q}_{lat} = \begin{bmatrix} \mathbf{C}_{lat} \\ \mathbf{C}_{lat}\mathbf{A}_{lat} \\ \mathbf{C}_{lat}\mathbf{A}_{lat}^2 \\ \mathbf{C}_{lat}\mathbf{A}_{lat}^3 \end{bmatrix}$$

Working

$$A_{lat}B_{lat} = \begin{bmatrix} \frac{2C_\alpha}{m} & 0 \\ -\frac{8C_\alpha^2}{m^2\dot{x}} - \frac{\frac{4C_\alpha^2 l_f (l_f - l_r)}{m I_z \dot{x}}}{m I_z \dot{x}} & 0 \\ \frac{2C_\alpha l_f}{I_z} & 0 \\ -\frac{4C_\alpha^2 (l_f - l_r)}{I_z m \dot{x}} - \frac{\frac{4C_\alpha^2 l_f (l_f^2 + l_r^2)}{I_z^2 \dot{x}}}{I_z^2 \dot{x}} & 0 \end{bmatrix}$$

$$A_{lat}^2 B_{lat} = \begin{bmatrix} -\frac{8C_\alpha^2}{m^2 \dot{x}} - \frac{4C_\alpha^2 l_f (l_f - l_r)}{m I_z \dot{x}} & 0 \\ \frac{32C_\alpha^3}{m^3 \dot{x}^2} + \frac{32C_\alpha^3 l_f (l_f - l_r)}{m^2 I_z \dot{x}^2} + \frac{8C_\alpha^3 l_f^2 (l_f - l_r)^2}{m I_z^2 \dot{x}^2} + \frac{16C_\alpha^3 l_f (l_f^2 + l_r^2) (l_f - l_r)}{m I_z^2 \dot{x}^2} & 0 \\ -\frac{4C_\alpha^2 l_f (l_f - l_r)}{I_z m \dot{x}} - \frac{4C_\alpha^2 l_f (l_f^2 + l_r^2)}{I_z^2 \dot{x}} & 0 \\ \frac{16C_\alpha^3 l_f (l_f - l_r)}{I_z^2 m \dot{x}^2} + \frac{16C_\alpha^3 l_f^2 (l_f^2 + l_r^2)}{I_z^3 \dot{x}^2} & 0 \end{bmatrix}$$

$$A_{lat}^3 B_{lat} = \begin{bmatrix} \frac{32C_\alpha^3}{m^3 \dot{x}^2} + \frac{32C_\alpha^3 l_f (l_f - l_r)}{m^2 I_z \dot{x}^2} + \frac{8C_\alpha^3 l_f^2 (l_f - l_r)^2}{m I_z^2 \dot{x}^2} + \frac{16C_\alpha^3 l_f (l_f^2 + l_r^2) (l_f - l_r)}{m I_z^2 \dot{x}^2} & 0 \\ -\frac{128C_\alpha^4}{m^4 \dot{x}^3} - \frac{128C_\alpha^4 l_f (l_f - l_r)}{m^3 I_z \dot{x}^3} - \frac{32C_\alpha^4 l_f^2 (l_f - l_r)^2}{m^2 I_z^2 \dot{x}^3} - \frac{64C_\alpha^4 l_f (l_f^2 + l_r^2) (l_f - l_r)}{m^2 I_z^2 \dot{x}^3} & 0 \\ \frac{16C_\alpha^3 l_f (l_f - l_r)}{I_z^2 m \dot{x}^2} + \frac{16C_\alpha^3 l_f^2 (l_f^2 + l_r^2)}{I_z^3 \dot{x}^2} & 0 \\ -\frac{64C_\alpha^4 l_f (l_f - l_r)}{I_z^3 m \dot{x}^3} - \frac{64C_\alpha^4 l_f^2 (l_f^2 + l_r^2)}{I_z^4 \dot{x}^3} & 0 \end{bmatrix}$$

$$C_{lat} A_{lat} = \begin{bmatrix} 0 & -\frac{4C_\alpha}{m \dot{x}} & \frac{4C_\alpha}{m} & -\frac{2C_\alpha (l_f - l_r)}{m \dot{x}} \\ 0 & -\frac{2C_\alpha (l_f - l_r)}{I_z \dot{x}} & \frac{2C_\alpha (l_f - l_r)}{I_z} & -\frac{2C_\alpha (l_f^2 + l_r^2)}{I_z \dot{x}} \end{bmatrix}$$

$$C_{lat} A_{lat}^2 = \begin{bmatrix} 0 & \frac{16C_\alpha^2}{m^2 \dot{x}^2} + \frac{8C_\alpha^2 (l_f - l_r)}{m I_z \dot{x}^2} & -\frac{16C_\alpha^2}{m^2} - \frac{8C_\alpha^2 (l_f - l_r)}{m I_z} & \frac{8C_\alpha^2 (l_f^2 + l_r^2)}{m I_z \dot{x}^2} + \frac{4C_\alpha^2 l_f (l_f - l_r)^2}{m I_z^2 \dot{x}^2} \\ 0 & \frac{8C_\alpha^2 (l_f - l_r)}{I_z m \dot{x}^2} + \frac{4C_\alpha^2 l_f (l_f^2 + l_r^2)}{I_z^2 \dot{x}^2} & -\frac{8C_\alpha^2 (l_f - l_r)}{I_z m} - \frac{4C_\alpha^2 l_f (l_f^2 + l_r^2)}{I_z^3} & \frac{4C_\alpha^2 (l_f^2 + l_r^2) (l_f - l_r)}{I_z^2 \dot{x}^2} + \frac{2C_\alpha^2 l_f (l_f - l_r)^3}{I_z^3 \dot{x}^2} \end{bmatrix}$$

$$C_{lat} A_{lat}^3 = \begin{bmatrix} 0 & -\frac{64C_\alpha^3}{m^3 \dot{x}^3} - \frac{32C_\alpha^3 (l_f - l_r)}{m^2 I_z \dot{x}^3} & \frac{64C_\alpha^3}{m^3 \dot{x}^2} + \frac{32C_\alpha^3 (l_f - l_r)}{m^2 I_z \dot{x}^2} & -\frac{32C_\alpha^3 (l_f^2 + l_r^2)}{m I_z^2 \dot{x}^3} - \frac{16C_\alpha^3 l_f (l_f - l_r)^2}{m I_z^2 \dot{x}^3} \\ 0 & -\frac{32C_\alpha^3 (l_f - l_r)}{I_z m^2 \dot{x}^3} - \frac{16C_\alpha^3 l_f (l_f^2 + l_r^2)}{I_z^3 \dot{x}^3} & \frac{32C_\alpha^3 (l_f - l_r)}{I_z m \dot{x}^2} + \frac{16C_\alpha^3 l_f (l_f^2 + l_r^2)}{I_z^3 \dot{x}^2} & -\frac{16C_\alpha^3 (l_f^2 + l_r^2) (l_f - l_r)}{I_z^3 \dot{x}^3} - \frac{8C_\alpha^3 l_f (l_f - l_r)^3}{I_z^3 \dot{x}^3} \end{bmatrix}$$

```
[2]: import sympy as sp
from sympy import Matrix, symbols, simplify, pprint, init_printing
sp.init_printing(use_latex=True)

# Symbols
C_alpha, m, I_z, l_f, l_r, x_dot = sp.symbols(r'C_{\alpha} m I_z l_f l_r' + r'\dot{x}', real=True)

# Assign numerical values (except x_dot)
param_values = {
    m: 1888.6,
    l_r: 1.39,
    l_f: 1.55,
    C_alpha: 20000,
    I_z: 25854
}

# State and input matrices
A_lat = sp.Matrix([
    [0, 1, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0]
])
```

```

[0, -4*C_alpha/(m*x_dot), 4*C_alpha/m, -2*C_alpha*(l_f - l_r)/(m*x_dot)],
[0, 0, 0, 1],
[0, -2*C_alpha*(l_f - l_r)/(I_z*x_dot), 2*C_alpha*(l_f - l_r)/I_z, -2*C_alpha*(l_f**2 + l_r**2)/(I_z*x_dot)]
])

B_lat = sp.Matrix([
[0, 0],
[2*C_alpha/m, 0],
[0, 0],
[2*C_alpha*l_f/I_z, 0]
])

C_lat = sp.Matrix([
[1, 0, 0, 0],
[0, 0, 1, 0]
])

P_lat = B_lat.row_join(A_lat * B_lat).row_join(A_lat**2 * B_lat).
    row_join(A_lat**3 * B_lat)
Q_lat = C_lat.col_join(C_lat * A_lat).col_join(C_lat * A_lat**2).col_join(C_lat *
    * A_lat**3)

# Substitute numerical values
A_lat_num = A_lat.subs(param_values)
B_lat_num = B_lat.subs(param_values)
C_lat_num = C_lat.subs(param_values)
P_lat_num = P_lat.subs(param_values)
Q_lat_num = Q_lat.subs(param_values)

# Simplify after substitution
A_lat_num = sp.simplify(A_lat_num)
B_lat_num = sp.simplify(B_lat_num)
P_lat_num = sp.simplify(P_lat_num)
Q_lat_num = sp.simplify(Q_lat_num)

# Display
print("A_lat (numeric except x_dot):")
display(A_lat_num)

print("B_lat (numeric):")
display(B_lat_num)

print("C_lat:")
display(C_lat_num)

print("P_lat (numeric except x_dot):")

```

```

display(P_lat_num)

print("Q_lat (numeric except x_dot):")
display(Q_lat_num)

```

A_lat (numeric except x_dot):

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{42.3594196759504}{\dot{x}} & 42.3594196759504 & -\frac{3.38875357407604}{\dot{x}} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{0.24754390036358}{\dot{x}} & 0.24754390036358 & -\frac{6.70627369072484}{\dot{x}} \end{bmatrix}$$

B_lat (numeric):

$$\begin{bmatrix} 0 & 0 \\ 21.1797098379752 & 0 \\ 0 & 0 \\ 2.39808153477218 & 0 \end{bmatrix}$$

C_lat:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

P_lat (numeric except x_dot):

$$\begin{bmatrix} 0 & 0 & 21.1797098379752 & 0 & -\frac{905.286725013534}{\dot{x}} & 0 & 101.581342148562 + \frac{38419.6858176627}{x^2} \\ 21.1797098379752 & 0 & -\frac{905.286725013534}{\dot{x}} & 0 & 101.581342148562 + \frac{38419.6858176627}{x^2} & 0 & -\frac{5208.2571924939}{\dot{x}} - \frac{1628}{x^2} \\ 0 & 0 & 2.39808153477218 & 0 & -\frac{21.325099086717}{\dot{x}} & 0 & 0.593630456507386 + \frac{367.110157814573}{x^2} \\ 2.39808153477218 & 0 & -\frac{21.325099086717}{\dot{x}} & 0 & 0.593630456507386 + \frac{367.110157814573}{x^2} & 0 & -\frac{34.4057881556766}{\dot{x}} - \frac{119}{x^2} \end{bmatrix}$$

Q_lat (numeric except x_dot):

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{42.3594196759504}{\dot{x}} & 42.3594196759504 & -\frac{3.38875357407604}{\dot{x}} \\ 0 & -\frac{0.24754390036358}{\dot{x}} & 0.24754390036358 & -\frac{6.70627369072484}{\dot{x}} \\ 0 & \frac{1795.1593005604}{\dot{x}^2} & -\frac{1795.1593005604}{\dot{x}} & 42.3594196759504 + \frac{166.27154376084}{\dot{x}^2} \\ 0 & \frac{12.1459131100303}{\dot{x}^2} & -\frac{12.1459131100303}{\dot{x}} & 0.24754390036358 + \frac{45.812972092006}{\dot{x}^2} \end{bmatrix}$$

[3]: x_dot_values = [2, 5, 8]

```

for x_val in x_dot_values:
    # Substitute x_dot
    P_sub = P_lat_num.subs(x_dot, x_val)
    Q_sub = Q_lat_num.subs(x_dot, x_val)

    # Compute ranks
    rank_P = P_sub.rank()

```

```

rank_Q = Q_sub.rank()

print(f"x_dot = {x_val} m/s:")
print(f" Rank of P_lat (controllability) = {rank_P}")
print(f" Rank of Q_lat (observability) = {rank_Q}\n")

```

x_dot = 2 m/s:
 Rank of P_lat (controllability) = 4
 Rank of Q_lat (observability) = 4

x_dot = 5 m/s:
 Rank of P_lat (controllability) = 4
 Rank of Q_lat (observability) = 4

x_dot = 8 m/s:
 Rank of P_lat (controllability) = 4
 Rank of Q_lat (observability) = 4

The Lateral system is controllable and observable for the respective speeds

0.2 Longitudinal Dynamics

State Matrix (\mathbf{A}_{lon})

$$\mathbf{A}_{lon} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Input Matrix (\mathbf{B}_{lon})

$$\mathbf{B}_{lon} = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix}$$

Controllability matrix (\mathbf{P}_{lon})

$$\mathbf{P}_{lon} = [\mathbf{B}_{lon} \quad \mathbf{A}_{lon}\mathbf{B}_{lon}]$$

Observability matrix (\mathbf{Q}_{lat})

$$\mathbf{Q}_{lon} = \begin{bmatrix} \mathbf{C}_{lon} \\ \mathbf{C}_{lon}\mathbf{A}_{lon} \end{bmatrix}$$

Working

$$A_{lon}B_{lon} = \begin{bmatrix} 0 & \frac{1}{m} \\ 0 & 0 \end{bmatrix}$$

$$A_{lon}^2 B_{lon} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C_{lon} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C_{lon} A_{lon} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$C_{lon} A_{lon}^2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
[4]: sp.init_printing(use_latex=True)

m = sp.symbols('m', real=True)

A_lon = sp.Matrix([[0, 1],
                  [0, 0]])

B_lon = sp.Matrix([[0, 0],
                  [0, 1/m]])

C_lon = sp.Matrix([[1, 0],
                  [0, 1]])

# Controllability matrix
P_lon = B_lon.row_join(A_lon*B_lon).subs(m, 1888.6)

# Observability matrix
Q_lon = C_lon.col_join(C_lon*A_lon).subs(m, 1888.6)

print("A_lon:")
display(A_lon)

print("B_lon:")
display(B_lon)

print("P_lon (controllability matrix):")
display(P_lon)

print("Q_lon (observability matrix):")
display(Q_lon)

print(f"P and Q don't depend on x_dot")
print(f" Rank of P_lon (controllability) = {P_lon.rank()}")

```

```
print(f"  Rank of Q_lon (observability) = {Q_lon.rank()}\n")
```

A_lon:

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

B_lon:

$$\begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix}$$

P_lon (controllability matrix):

$$\begin{bmatrix} 0 & 0 & 0 & 0.000529492745949381 \\ 0 & 0.000529492745949381 & 0 & 0 \end{bmatrix}$$

Q_lon (observability matrix):

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

P and Q don't depend on x_dot

Rank of P_lon (controllability) = 2

Rank of Q_lon (observability) = 2

The Longitudinal system is controllable and observable for the respective speeds

Robustness_with_speed

November 5, 2025

```
[2]: import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import svd, eigvals

sp.init_printing(use_latex=True)
C_alpha, m, I_z, l_f, l_r, x_dot = sp.symbols('C_alpha m I_z l_f l_r x_dot', real=True)

A_lat = sp.Matrix([
    [0, 1, 0, 0],
    [0, -4*C_alpha/(m*x_dot), 4*C_alpha/m, -2*C_alpha*(l_f - l_r)/(m*x_dot)],
    [0, 0, 0, 1],
    [0, -2*C_alpha*(l_f - l_r)/(I_z*x_dot), 2*C_alpha*(l_f - l_r)/I_z, -2*C_alpha*(l_f**2 + l_r**2)/(I_z*x_dot)]
])

B_lat = sp.Matrix([
    [0, 0],
    [2*C_alpha/m, 0],
    [0, 0],
    [2*C_alpha*l_f/I_z, 0]
])

P_lat = B_lat.row_join(A_lat*B_lat).row_join(A_lat**2*B_lat).row_join(A_lat**3*B_lat)

params = {
    C_alpha: 20000,
    m: 1888.6,
    I_z: 25854,
    l_f: 1.55,
    l_r: 1.39
}

v_values = np.linspace(1, 40, 40)
log_sigma_ratio = []
```

```

poles_real = []

for v in v_values:
    A_num = A_lat.subs(params).subs(x_dot, v)
    P_num = P_lat.subs(params).subs(x_dot, v)

    A_np = np.array(A_num).astype(np.float64)
    P_np = np.array(P_num).astype(np.float64)

    sigma = svd(P_np, compute_uv=False)
    log_sigma_ratio.append(np.log10(np.max(sigma)/np.min(sigma)))

    eigs = eigvals(A_np)
    poles_real.append(np.real(eigs))

log_sigma_ratio = np.array(log_sigma_ratio)
poles_real = np.array(poles_real)

# Find the index closest to v=20
v_desired = 10
idx_v = np.argmin(np.abs(v_values - v_desired))

# Print values at v=20
print(f"v = {v_desired} m/s:")
print(f"Controllability condition number (log10(sigma_max/sigma_min)) ="
    ↪{log_sigma_ratio[idx_v]:.4f}")
for i in range(4):
    print(f"Real part of pole {i+1} = {poles_real[idx_v, i]:.4f}")

# Plot controllability condition number
plt.figure(figsize=(10,5))
plt.plot(v_values, log_sigma_ratio, 'b', linewidth=2)
plt.plot(v_values[idx_v], log_sigma_ratio[idx_v], 'ro', markersize=8,_
    ↪label=f'v={v_desired} m/s')
plt.xlabel('v (m/s)')
plt.ylabel(r'$\log_{10}(\sigma_{\max}/\sigma_{\min})$')
plt.title('Controllability condition number vs velocity')
plt.grid(True)
plt.legend()
plt.show()

# Plot poles real part
plt.figure(figsize=(10,8))
for i in range(4):
    plt.subplot(2, 2, i+1)
    plt.plot(v_values, poles_real[:, i], 'r', linewidth=2)

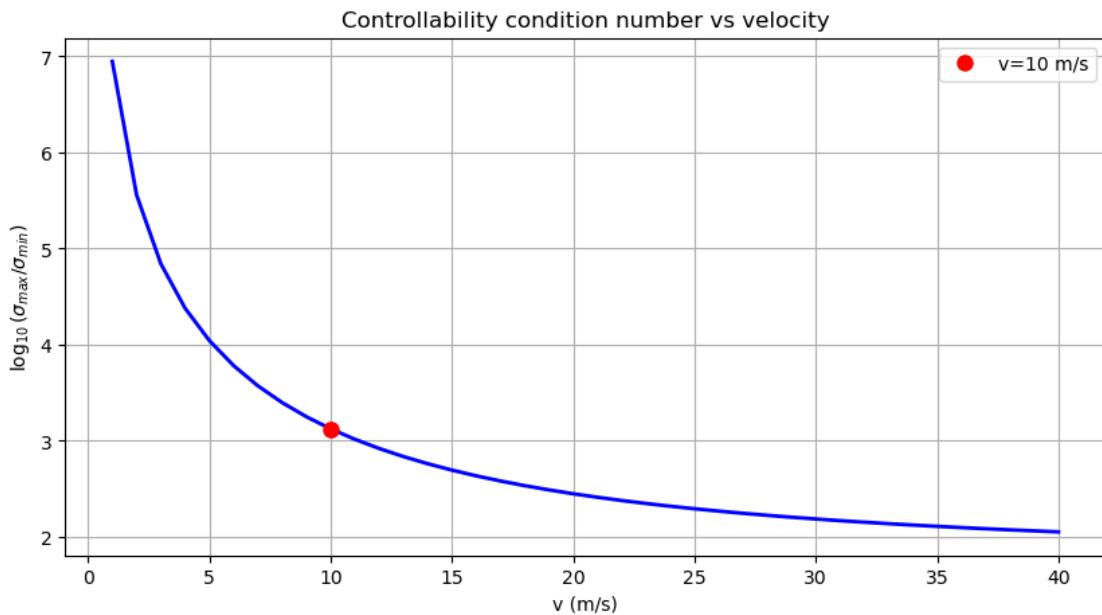
```

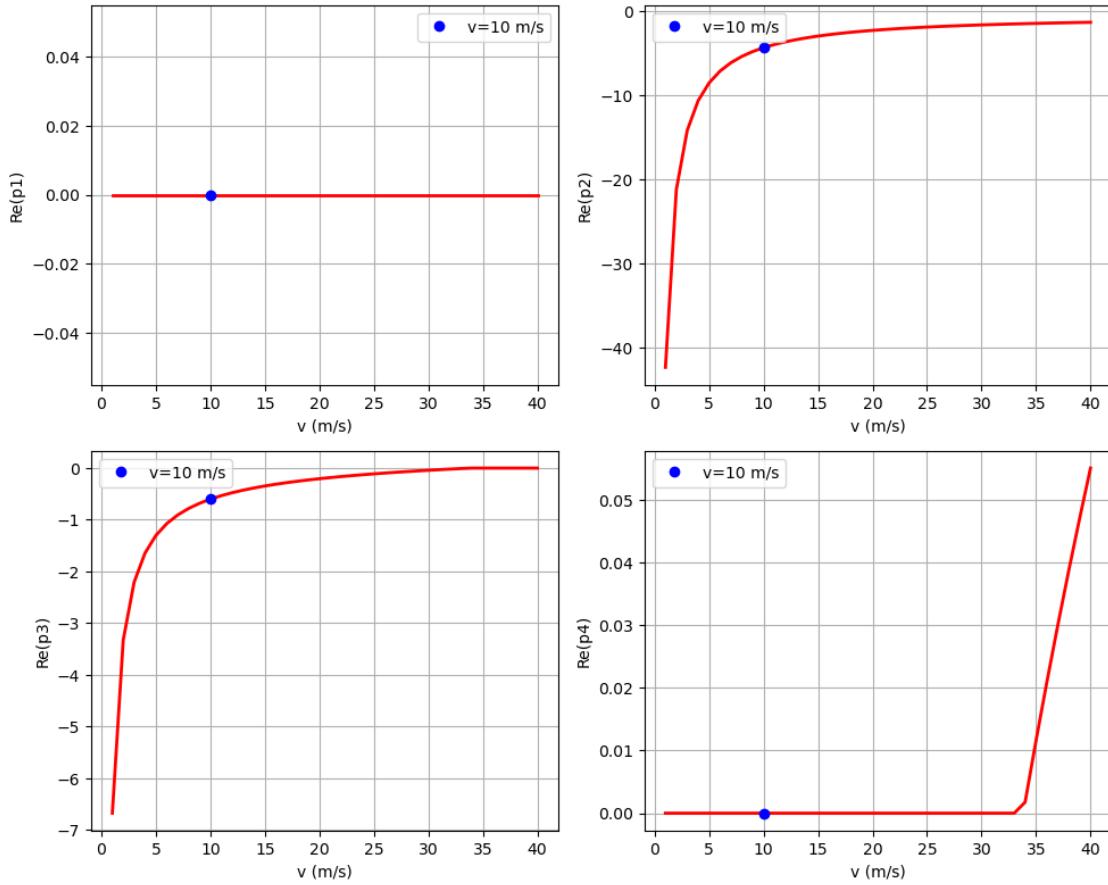
```

plt.plot(v_values[idx_v], poles_real[idx_v, i], 'bo', markersize=6, label=f'v={v_desired} m/s')
plt.xlabel('v (m/s)')
plt.ylabel(f'Re(p{i+1})')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

```

v = 10 m/s:
 Controllability condition number ($\log_{10}(\sigma_{\max}/\sigma_{\min})$) = 3.1247
 Real part of pole 1 = 0.0000
 Real part of pole 2 = -4.3063
 Real part of pole 3 = -0.6002
 Real part of pole 4 = 0.0000





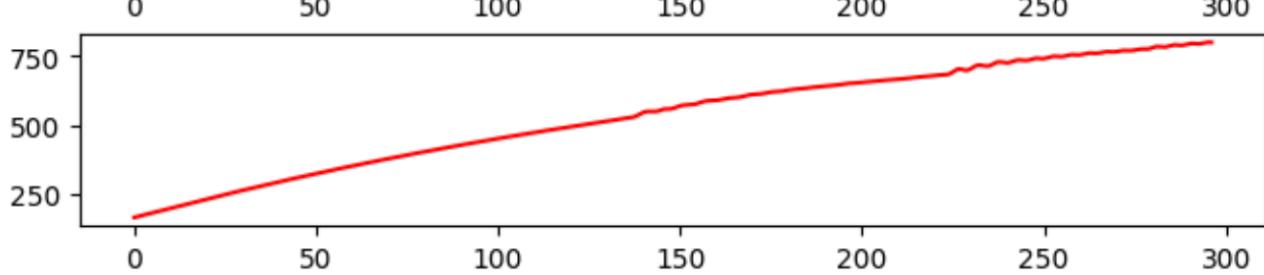
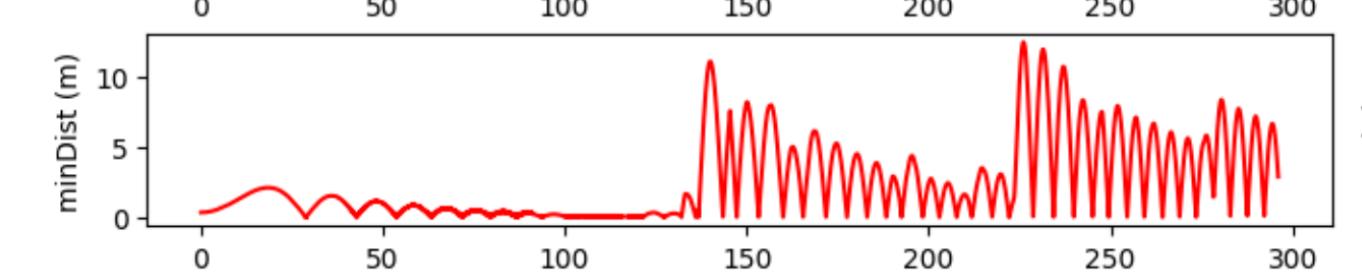
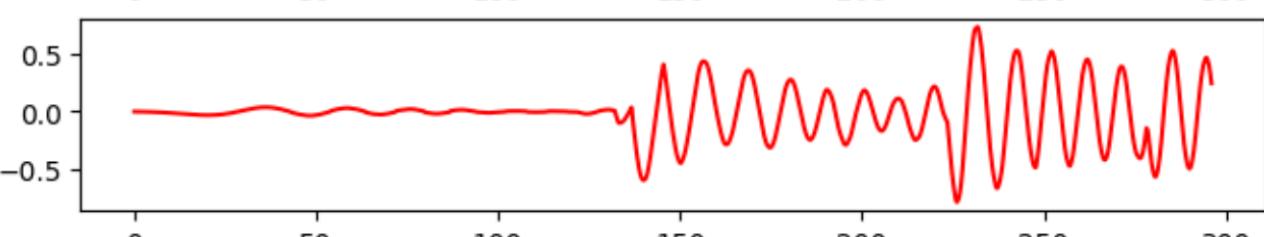
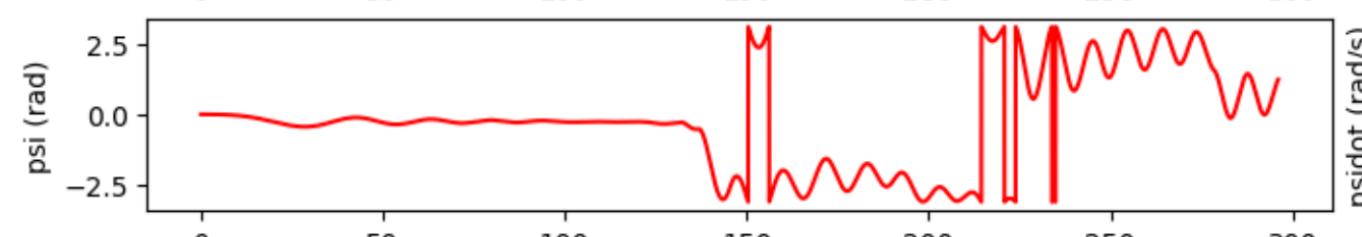
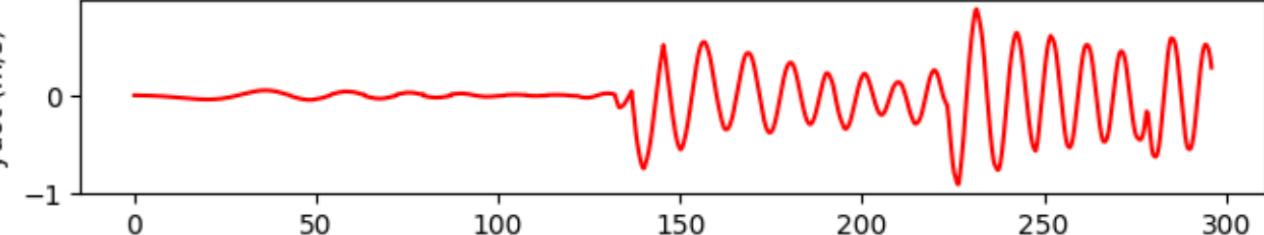
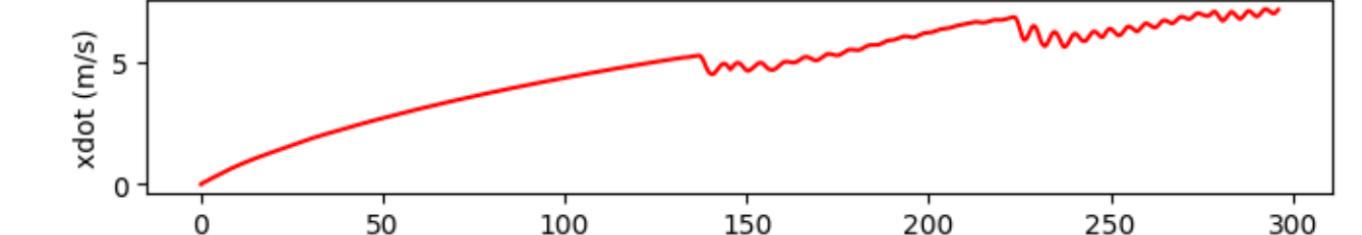
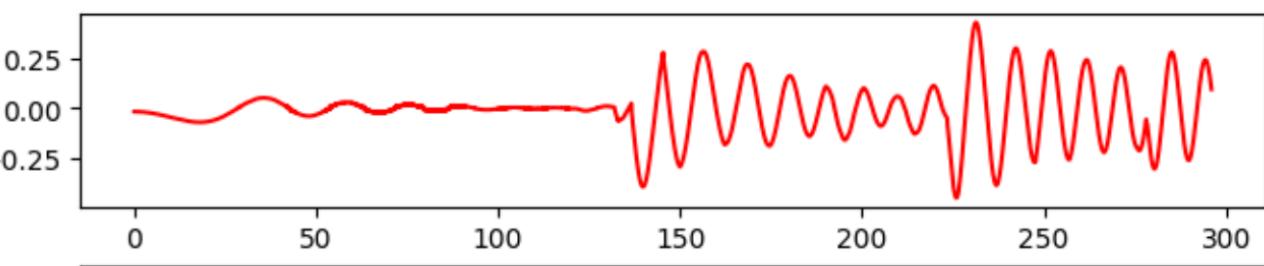
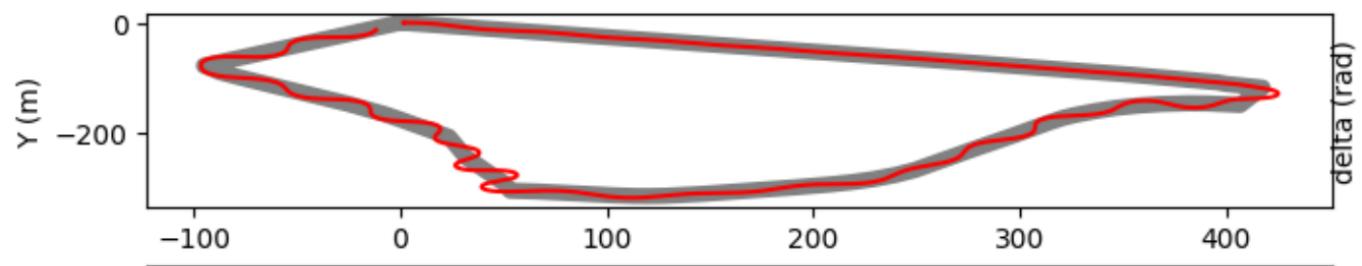
0.0.1 Conclusions on Controllability and Stability

Controllability:

- The plot of $\log_{10}(\sigma_{\max}/\sigma_{\min})$ (from your previous message) decreases with velocity, indicating that the controllability condition improves as speed increases. The system is less controllable at low speeds and becomes more controllable at higher speeds.

Stability:

- Three poles have negative real parts at low speeds, indicating stability.
- One pole remains at zero real part for most speeds but starts becoming positive after about 33 m/s, indicating the system transitions from stable/marginally stable to unstable at high velocities.
- Two poles approach zero from negative values, suggesting marginal stability zones at higher speeds.



File Edit View Simulation Build Overlays Tools Wizards Help

Simulation View

...CMU\SEM_1\24677-MCT\Proj\Proj_2\controllers\main\your_controller.py

- > ● WorldInfo
- > ● Viewpoint
- > ● TexturedBackground
- > ● TexturedBackgroundLight
- > ● DEF TESLA Robot
- > ● Floor "floor"
- > ● Road "road"
- > ● Slide "slide"
- > ● Swing "swing"
- > ● Forest

Selection: Forest (Transform)

Node

DEF: []



```
your_controller.py x
35     def __init__(self, trajectory):
36         super().__init__(trajectory)
37
38         # --- Vehicle constants ---
39         self.lr = 1.39
40         self.lf = 1.55
41         self.Ca = 20000
42         self.Iz = 25854
43         self.m = 1888.6
44         self.g = 9.81
45
46         # --- Controller ---
47         self.delT = 0.032
48         self.longitudinal_pid = PID(Kp=16.58, Ki=0.482,
49                                     self.desired_speed = 10
50
51         # --- Lateral gain ---
52         self.K = self.design_lateral_controller()
53         # print("Lateral gain K:", self.K)
54
55     def design_lateral_controller(self):
56         """Calculates the state-feedback gain vector K
57
58         lr, lf, Ca, Iz, m, v_design, delT = self.lr, se
```

Console - All

INFO: main: Starting controller: python.exe -u main.py
DEPRECATION: Robot.getDisplay is deprecated, please use Robot.getDevice instead.
DEPRECATION: Robot.getDisplay is deprecated, please use Robot.getDevice instead.

Evaluating...

Score for completing the loop: 30.0/30.0
 Score for average distance: 30.0/30.0
 Score for maximum distance: 29.066046913847153/30.0
 Your time is 296.032
 Your total score is : 99.06604691384715/100.0
 total steps: 296032
 maxMinDist: 12.398901988499045
 avgMinDist: 2.4940928754875693
 INFO: 'main' controller exited successfully.