Exercise 1

~~Given, $\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-4C_\alpha}{m\dot{x}} & \frac{4C_\alpha}{m} & \frac{2C_\alpha(I_r - I_f)}{m\dot{x}} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-2C_\alpha(I_f - I_r)}{I_z\dot{x}} & \frac{2C_\alpha(I_f - I_r)}{I_z} & \end{bmatrix}$~~

Given

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-4C_\alpha}{mV} & \frac{4C_\alpha}{m} & \frac{-2C_\alpha(I_f - I_r)}{mV} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-2C_\alpha(I_f - I_r)}{I_z V} & \frac{2C_\alpha(I_f - I_r)}{I_z} & \frac{-2C_\alpha(I_f^2 + I_r^2)}{I_z V} \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ \frac{2C_\alpha}{m} \\ 0 \\ \frac{2C_\alpha I_f}{I_z} \end{bmatrix} [\delta]$$

Here $V = \dot{x}$

Provided, $l_r = \overset{1.39}{\cancel{\quad}}\,m$, $l_f = \overset{1.55}{\cancel{\quad}}\,m$, $C_\alpha = 20000$

$I_z = 25854$ $m = 1888.6\,kg$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-42.359}{V} & 42.359 & \frac{-3.388}{V} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-0.2475}{V} & 0.2475 & \frac{-6.7}{V} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 21.1295 \\ 0 \\ 2.398 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P = [B \quad AB \quad A^2B \quad A^3B)$$

$$= \begin{bmatrix} 0 & 21.179 & -\frac{905.27}{V} & \frac{38418.39}{V^2} + 101 \\ 21.179 & -\frac{905.27}{V} & \frac{38418.39}{V^2} + 101.57 & -\frac{5207 - 1628607}{V^3} \\ 0 & 2.398 & -21.3/9 & 366/v^2 + 0.59 \\ 2.398 & -\frac{21.308}{V} & 366/9^2 + 0.59 & -\frac{34.39v^2 - 11966}{v^3} \end{bmatrix}$$

Because of the complexity of calculation, P & Q are calculated in python for velo = [2, 5, 8] m/s

From the code, we can see for all the three velocity the system is both controllable & observable.

Q2. From the Real values of poles vs velocity plot ~~figure~~, as the velocity increases we can see that for pole 1-3, the values start from negative to zero hence becoming less stable & for pole 4 the value increases from 0 to high positive value, which also shows the system becoming less stable.

But in the $\log_{10}(\sigma_1/\sigma_n)$ vs velocity plot we can see the log ratio approaches zero at high velocity, which shows the system becoming more & more controllable, as σ_1 value increases & pseudo full rank

In [14]:

```
import numpy as np
import matplotlib.pyplot as plt
```

In [15]:

```
Velo = [2, 5, 8]
```

In [16]:

```
for v in Velo:
    A= np.array([[0, 1], [0, 0]])
    print(A.shape)
    C = np.identity(2)
    B = np.expand_dims(np.array([0, 1/1888.6]).T, axis =1)


    P = np.hstack((B, A@B, A@A@B, A@A@A@B))
    #print(P.shape)

    Q = np.vstack((C, C@A, C@A@A, C@A@A@A))
    #print(Q.shape)
    rankP = np.linalg.matrix_rank(P)
    rankQ = np.linalg.matrix_rank(Q)
    print(f'For Velocity = {v} m/s\n')
    print(f'Rank of Controllability matrix = {rankP}')
    print(f'Rank of Observability matrix = {rankQ} \n')
```

```
(2, 2)
For Velocity = 2 m/s

Rank of Controllability matrix = 2
Rank of Observability matrix = 2

(2, 2)
For Velocity = 5 m/s

Rank of Controllability matrix = 2
Rank of Observability matrix = 2

(2, 2)
For Velocity = 8 m/s

Rank of Controllability matrix = 2
Rank of Observability matrix = 2
```

In [17]:

```
for v in Velo:
    A= np.array([[0, 1, 0, 0],
                 [0, -42.359/v, 42.359, -3.388/v ],
                 [0, 0, 0, 1],
                 [0 , -0.2475/v, 0.2475, -6.7/v]])
    C = np.identity(4)
    B = np.expand_dims(np.array([0, 21.1795, 0, 2.398]).T, axis =1)

    P = np.hstack((B, A@B, A@A@B, A@A@A@B))
    #print(P.shape)

    Q = np.vstack((C, C@A, C@A@A, C@A@A@A))
    #print(Q.shape)
    rankP = np.linalg.matrix_rank(P)
    rankQ = np.linalg.matrix_rank(Q)
    print(f'For Velocity = {v} m/s\n')
    print(f'Rank of Controllability matrix = {rankP}')
    print(f'Rank of Observability matrix = {rankQ} \n')
```

```
For Velocity = 2 m/s

Rank of Controllability matrix = 4
Rank of Observability matrix = 4

For Velocity = 5 m/s

Rank of Controllability matrix = 4
Rank of Observability matrix = 4

For Velocity = 8 m/s

Rank of Controllability matrix = 4
Rank of Observability matrix = 4
```

From the results above we can see the system is controllable and observable in all the velocity

Part 2

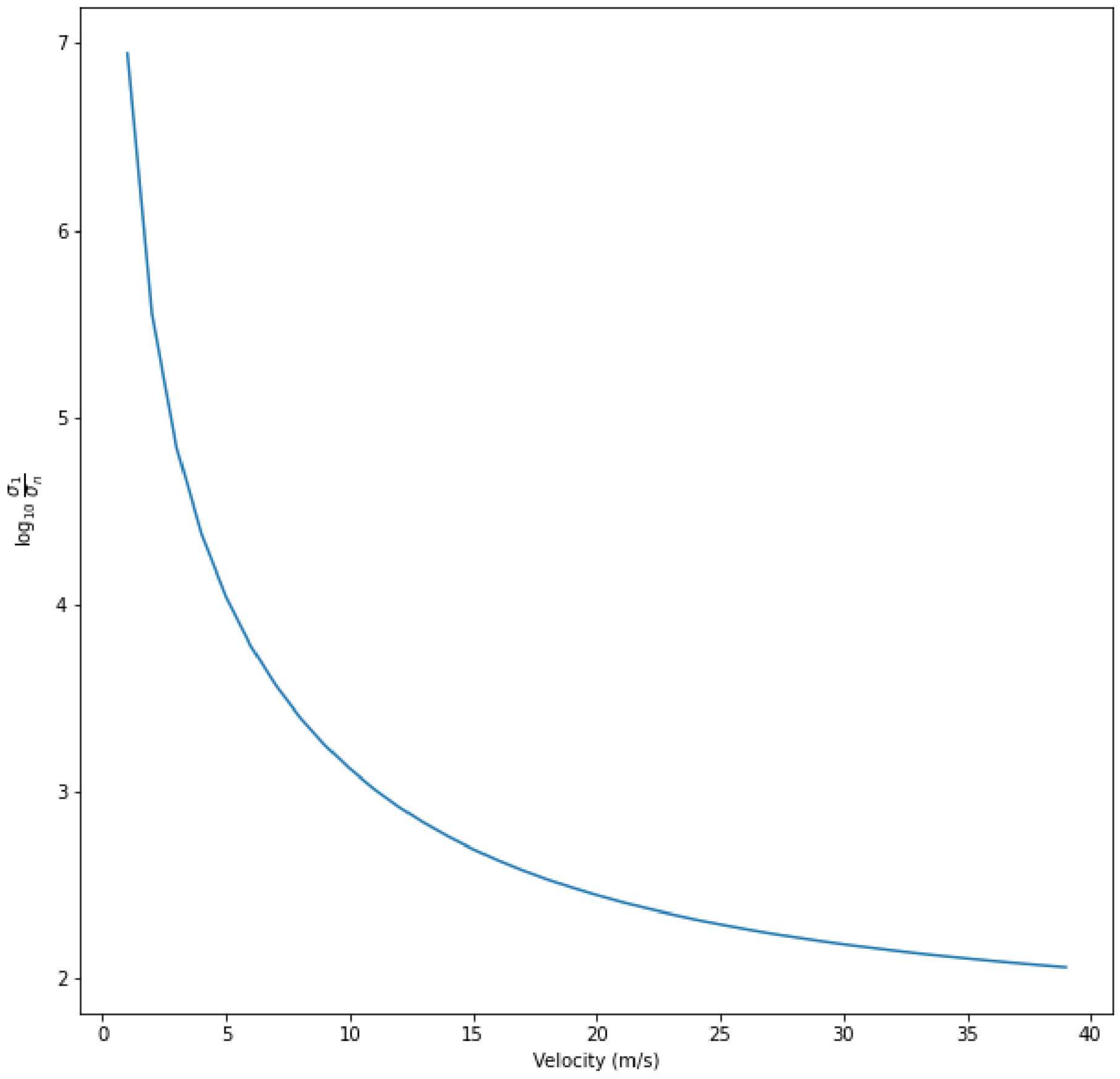
In [18]:

```
pole1 =[]
pole2 =[]
pole3 =[]
pole4 =[]
log =[]
for i in range(1, 40):
    v = i
    A= np.array([[0, 1, 0, 0],
                 [0, -42.359/v, 42.359, -3.388/v ],
                 [0, 0, 0, 1],
                 [0 , -0.2475/v, 0.2475, -6.7/v]])
    B = np.expand_dims(np.array([0, 21.1795, 0, 2.398]).T, axis =1)

    C = np.array([1, 1, 1, 1])
    #C = np.identity(4)

    D = np.array([0])
    #D = np.zeros((4,4))
    P = np.hstack((B, A@B, A@A@B, A@A@A@B))
    #print(P.shape)

    Q = np.vstack((C, C@A, C@A@A, C@A@A@A))

    U, sigma, V = np.linalg.svd(P)
    sigma1 = np.max(sigma)
    sigman = np.min(sigma)
    log.append(np.log10(sigma1 / sigman))
    Eig = np.linalg.eig(A)
    #print(Eig[0])
    #print(Eig[1])
    #S = control.StateSpace(A, B, C, D)
    #poles = control.pole(S)

    pole1.append(Eig[0][0].real)
    pole2.append(Eig[0][1].real)
    pole3.append(Eig[0][2].real)
    pole4.append(Eig[0][3].real)
```

Sigma plot

In [19]:

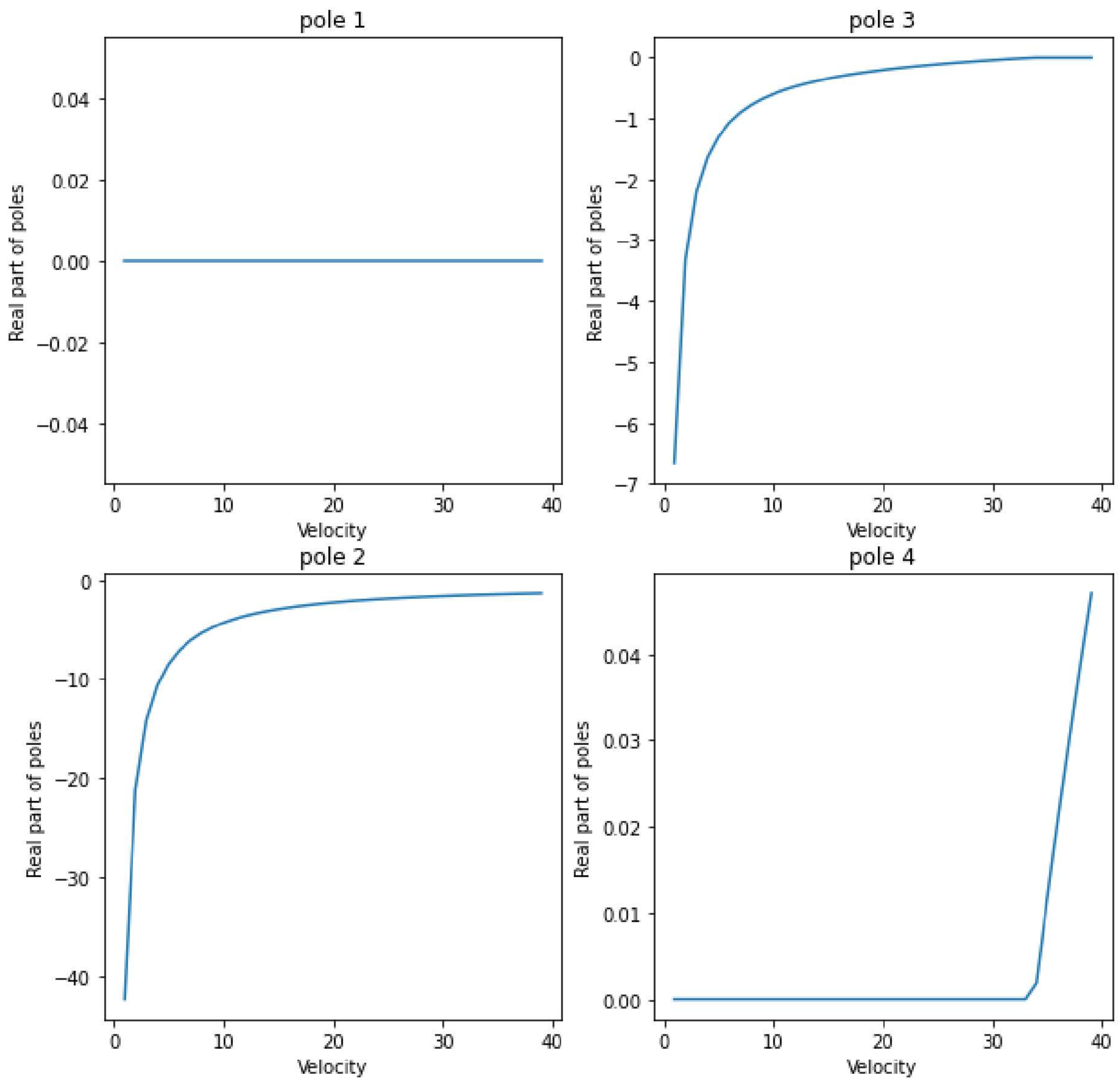
```
plt.figure(figsize = (10, 10))
plt.plot(np.arange(1, 40), log)
plt.xlabel('Velocity (m/s)')
plt.ylabel('$\log_{10}$ $\dfrac{\sigma_1}{\sigma_n}$')
plt.show()
```

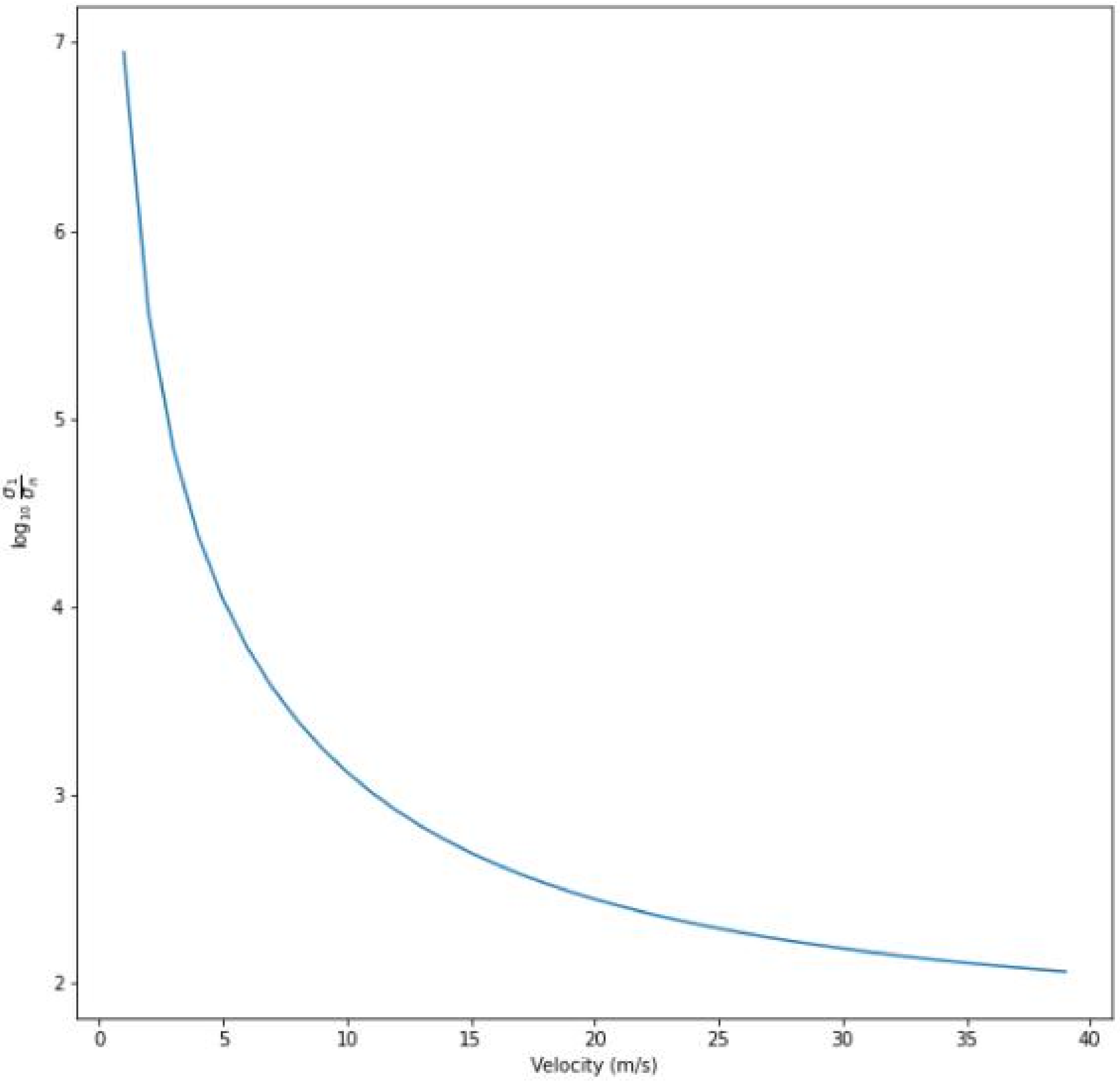


Poles plot

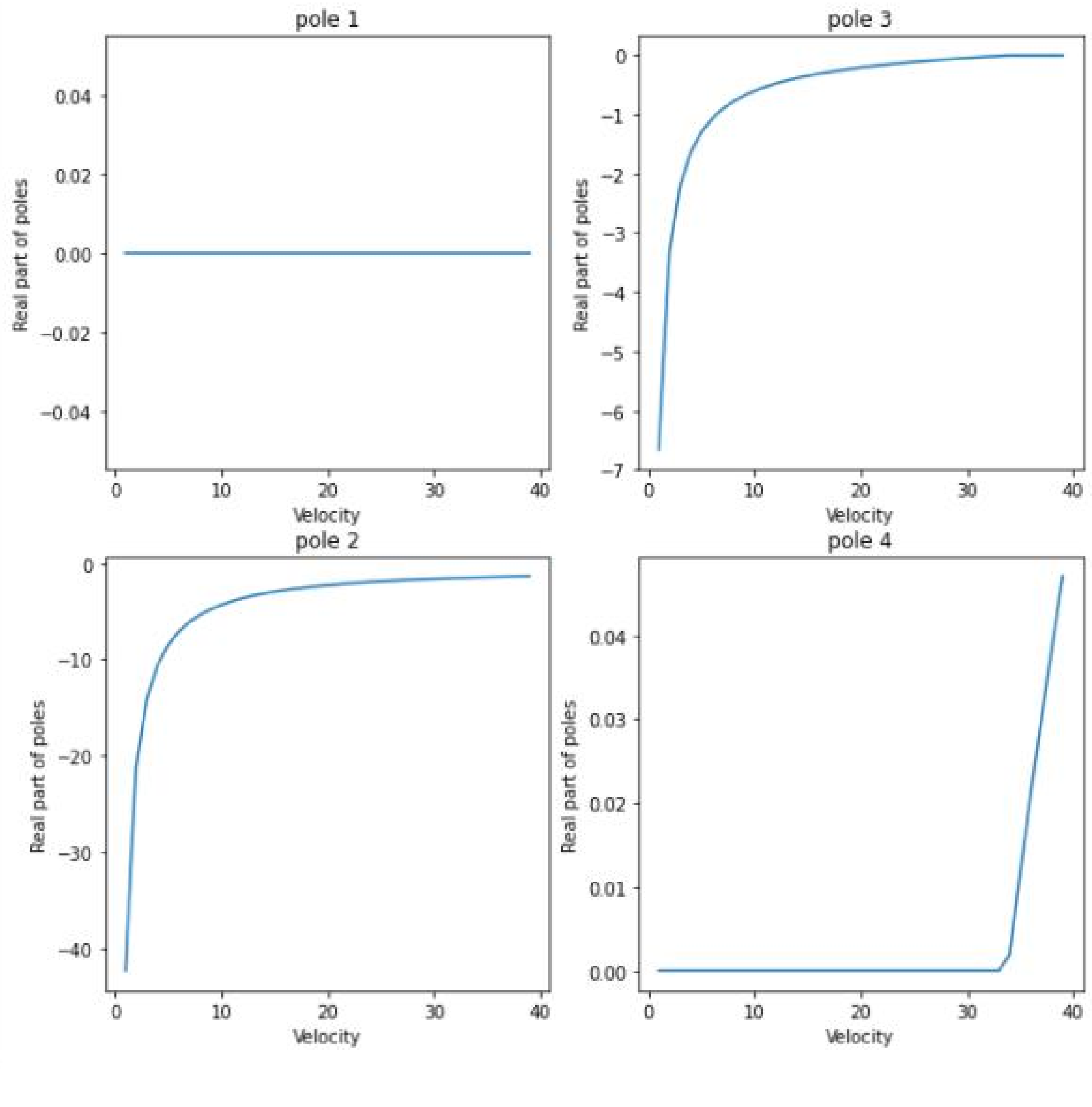
In [20]:

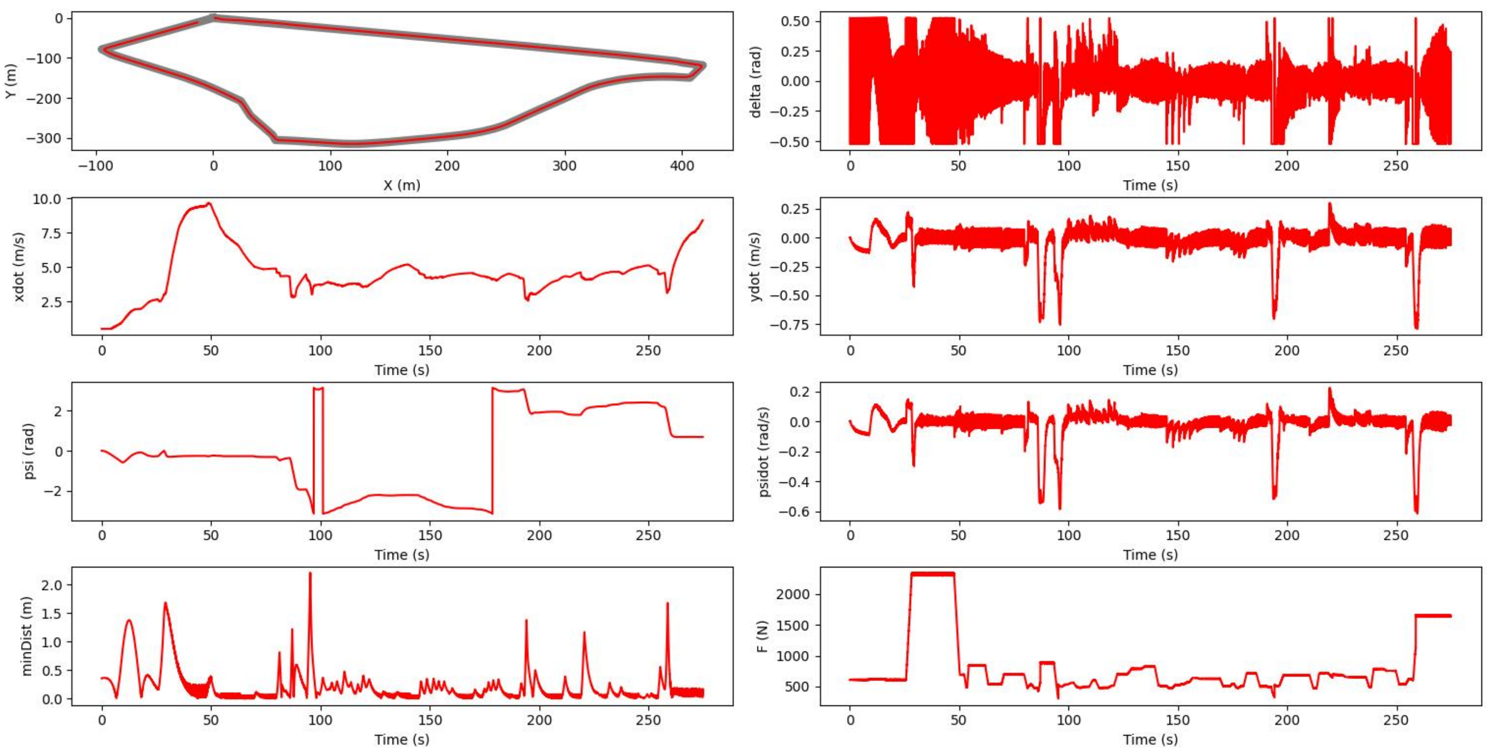
```
fig, ax = plt.subplots(2, 2, figsize=(10,10))


ax[0,0].set_title('pole 1')
ax[0,0].plot(np.arange(1, 40), pole1)



ax[1,0].set_title('pole 2')
ax[1,0].plot(np.arange(1, 40), pole2)


ax[0,1].set_title('pole 3')
ax[0,1].plot(np.arange(1, 40), pole3)

ax[1,1].set_title('pole 4')
ax[1,1].plot(np.arange(1, 40), pole4)
for a in ax.flat:
    a.set(xlabel='Velocity', ylabel='Real part of poles')
```

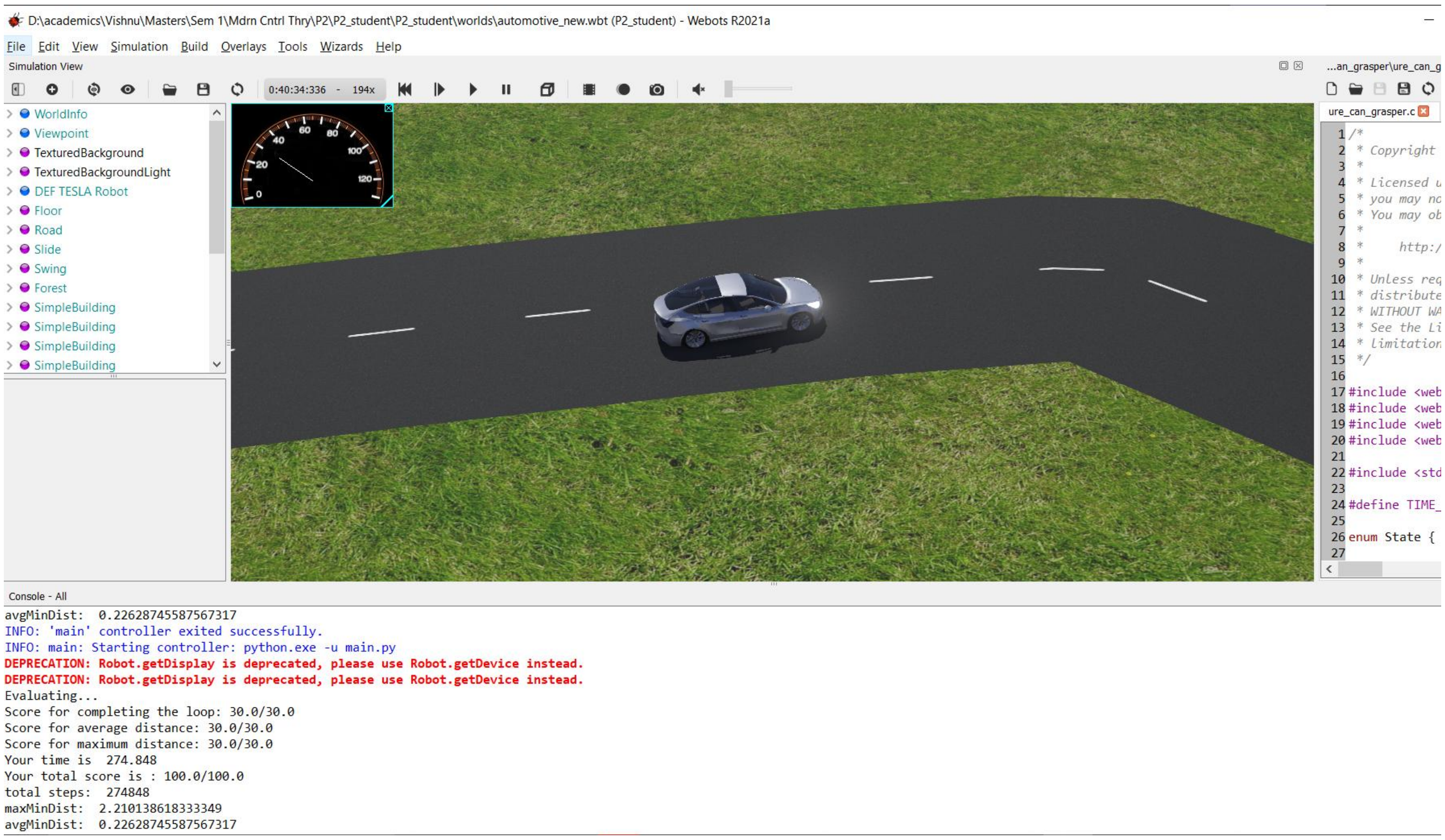




7
6
5
4
3
2
$\log_{10} \frac{\sigma_1}{\sigma_n}$
0
5
10
15
20
25
30
35
40
Velocity (m/s)


pole 1
pole 3
pole 2
pole 4
Real part of poles
Velocity
0.04
0.02
0.00
−0.02
−0.04
0
−1
−2
−3
−4
−5
−6
−7
−10
−20
−30
−40
0.03
0.01
10
20
30
40


Y (m)
X (m)
delta (rad)
xdot (m/s)
ydot (m/s)
psi (rad)
psidot (rad/s)
minDist (m)
F (N)
Time (s)



```
/*
 * Copyright
 *
 * Licensed u
 * you may nc
 * You may ob
 *
 *     http:/
 *
 * Unless req
 * distribute
 * WITHOUT WA
 * See the Li
 * limitation
 */

#include <web
#include <web
#include <web
#include <web

#include <std

#define TIME_

enum State {
```

```
avgMinDist:  0.22628745587567317
INFO: 'main' controller exited successfully.
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: 30.0/30.0
Your time is  274.848
Your total score is : 100.0/100.0
total steps:  274848
maxMinDist:  2.210138618333349
avgMinDist:  0.22628745587567317
```