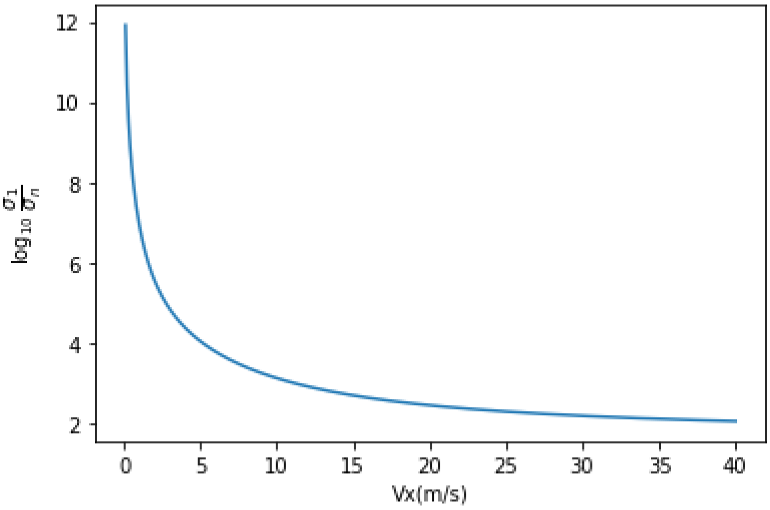
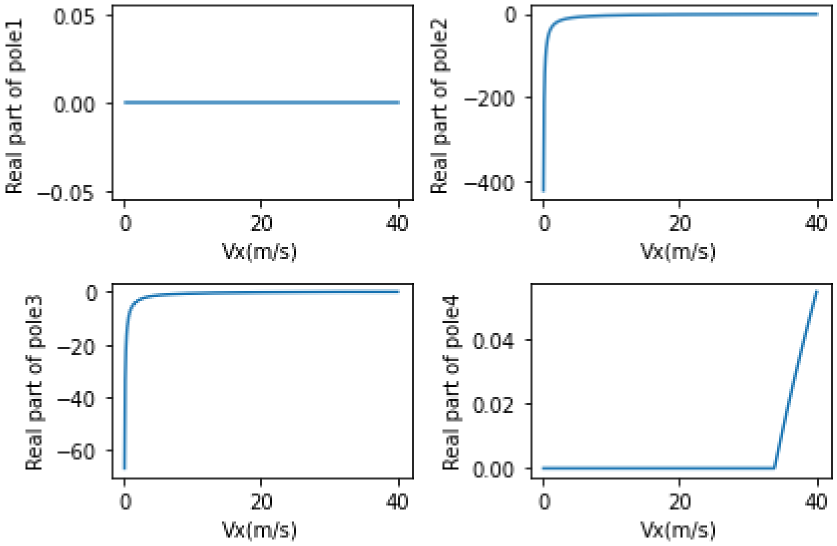
```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import matrix_rank
import keras

import control as con


A = None
for i in (2,5,8):
    Vx = i
    A = np.array([[0, 1, 0, 0],
                  [0, -42.35941968/Vx, 42.35941968, -3.388753574/Vx],
                  [0, 0, 0,1],
                  [0, -0.2475439004/Vx, 0.2475439004, -6.706273690/Vx]])
    B = np.array([[0],[21.17970984],[0],[2.398081534]])
    C = np.eye(4)
    #print (C)
    P = np.hstack((B, A@B, np.linalg.matrix_power(A, 2)@B, np.linalg.matrix_power(A, 3)@B))
    Q = np.vstack((C,C@A,C@np.linalg.matrix_power(A, 2),C@np.linalg.matrix_power(A, 3)))
    rank_control = matrix_rank(P)
    rank_observe = matrix_rank(Q)
    if(rank_control == 4):

        print("Nice!!!  The system is controllable with longitudinal velocitie =",Vx,"m/s"

    else:
        print("Sorry!!!! The system is not controllable with longitudinal velocitie =",Vx,
    if(rank_observe == 4):

        print("Nice!!!  The system is observable with longitudinal velocitie =",Vx,"m/s")

    else:
        print("Sorry!!!! The system is not observable with longitudinal velocitie =",Vx,"m
```

```
Nice!!!  The system is controllable with longitudinal velocitie = 2 m/s
Nice!!!  The system is observable with longitudinal velocitie = 2 m/s
Nice!!!  The system is controllable with longitudinal velocitie = 5 m/s
Nice!!!  The system is observable with longitudinal velocitie = 5 m/s
Nice!!!  The system is controllable with longitudinal velocitie = 8 m/s
Nice!!!  The system is observable with longitudinal velocitie = 8 m/s
```

```
velo = np.arange(0.1,40.1,0.1)
loga = []
V = []
poles = [[],[],[],[]]

for i in velo:
    Vx = i
    V.append(Vx)
    A = np.array([[0, 1, 0, 0],
                [0, -42.35941968/Vx, 42.35941968, -3.388753574/Vx],
                [0, 0, 0,1],
                [0, -0.2475439004/Vx, 0.2475439004, -6.706273690/Vx]])
    B = np.array([[0],[21.17970984],[0],[2.398081534]])
    C = np.ones(4)
    D = [0]
    D = np.array(D)

    P = np.hstack((B, A@B, np.linalg.matrix_power(A, 2)@B, np.linalg.matrix_power(A, 3)@B))
    u,s,vh = np.linalg.svd(P)
    idx = s.argsort()[::-1]
    s = s[idx]
    singular1 = s[0]
    singularn = s[-1]
    lo = np.log10(singular1/singularn)
    loga.append(lo)
    sys = con.StateSpace(A, B, C, D)

    p = con.pole(sys)

    poles[0].append(p[0].real)
    poles[1].append(p[1].real)
    poles[2].append(p[2].real)
    poles[3].append(p[3].real)

plt.plot(V, loga)
plt.xlabel('Vx(m/s)')
plt.ylabel('$\log_{10}$ $\dfrac{\sigma_1}{\sigma_n}$')
plt.show()
```



```
for i in range(4):
    plt.subplot(2, 2, 1+i)
    plt.xlabel('Vx(m/s)')
    plt.ylabel('Real part of pole'+str(i+1))
    plt.plot(V, poles[i])
plt.tight_layout()
plt.show()
```


Real part of pole1
Vx(m/s)
Real part of pole2
Vx(m/s)
Real part of pole3
Vx(m/s)
Real part of pole4
Vx(m/s)