

Python Code:

```
import numpy as np
from numpy.linalg import matrix_rank
import control as ctrl

# Define the system matrices

# System 1
A1 = np.array([[ -4, 1], [0, 2]])
B1 = np.array([[1], [0]])

# System 2
A2 = np.array([[ -3, 2], [4, 1]])
B2 = np.array([[0], [1]])

# Check stabilizability and find K for System 1
R1 = np.hstack([B1, A1@B1]) # Controllability matrix [B AB]
if matrix_rank(R1) == A1.shape[0]: # Check if controllability matrix is full rank
    P1 = np.array([ -3, -5]) # Define desired closed-loop poles
    K1 = ctrl.place(A1, B1, P1) # Compute gain K of stabilizing static state-
feedback control law u = K*xp
    print("System 1 is stabilizable by a static state-feedback control law u =
K*xp")
    print("Stabilizing gain matrix K for System 1 with desired poles at {}
is:".format(P1))
    print(K1)
else:
    print("System 1 is not stabilizable")

# Check stabilizability and find K for System 2
R2 = np.hstack([B2, A2@B2]) # Controllability matrix [B AB]
if matrix_rank(R2) == A2.shape[0]: # Check if controllability matrix is full rank
    P2 = np.array([ -3, -5]) # Define desired closed-loop poles
    K2 = ctrl.place(A2, B2, P2) # Compute gain K of stabilizing static state-
feedback control law u = K*xp
    print("\nSystem 2 is stabilizable by a static state-feedback control law u =
K*xp")
    print("Stabilizing gain matrix K for System 2 with desired poles at {}
is:".format(P2))
    print(K2)
else:
    print("\nSystem 2 is not stabilizable")
```

Output:

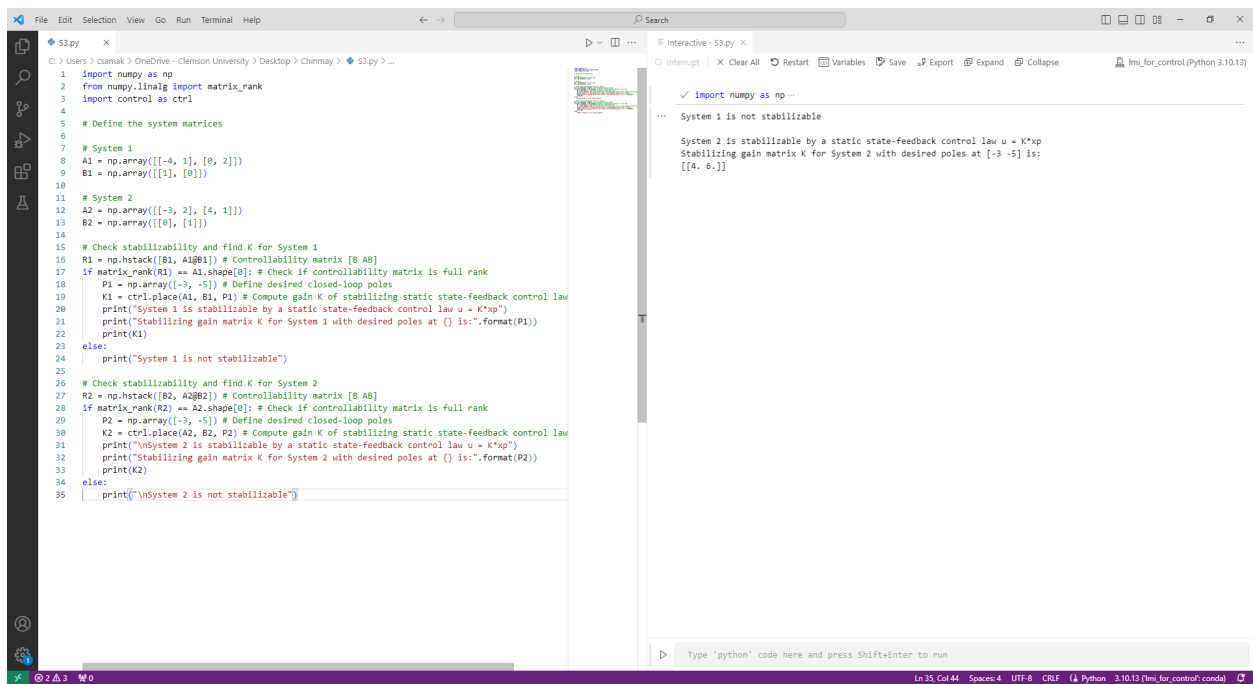
System 1 is not stabilizable

System 2 is stabilizable by a static state-feedback control law $u = K \cdot x_p$

Stabilizing gain matrix K for System 2 with desired poles at $[-3 \ -5]$ is:

$\begin{bmatrix} 4. & 6. \end{bmatrix}$

Screenshot:



The screenshot displays a Jupyter Notebook interface with a Python script for system stabilization. The code defines two systems, System 1 and System 2, and checks their stabilizability. System 1 is found to be not stabilizable, while System 2 is stabilizable. For System 2, a stabilizing gain matrix K is calculated and displayed.

```
1 import numpy as np
2 from numpy.linalg import matrix_rank
3 import control as ctrl
4
5 # Define the system matrices
6
7 # System 1
8 A1 = np.array([[ -4, 1], [0, 2]])
9 B1 = np.array([[1], [0]])
10
11 # System 2
12 A2 = np.array([[ -3, 2], [4, 1]])
13 B2 = np.array([[0], [1]])
14
15 # Check stabilizability and find K for System 1
16 R1 = np.hstack([B1, A1*B1]) # Controllability matrix [8 AB]
17 if matrix_rank(R1) == A1.shape[0]: # Check if controllability matrix is full rank
18     P1 = np.array([ -3, -5]) # Define desired closed-loop poles
19     K1 = ctrl.place(A1, B1, P1) # Compute gain K of stabilizing static state-feedback control law
20     print("System 1 is stabilizable by a static state-feedback control law u = K*xp")
21     print("Stabilizing gain matrix K for System 1 with desired poles at {} is:".format(P1))
22     print(K1)
23 else:
24     print("System 1 is not stabilizable")
25
26 # Check stabilizability and find K for System 2
27 R2 = np.hstack([B2, A2*B2]) # Controllability matrix [8 AB]
28 if matrix_rank(R2) == A2.shape[0]: # Check if controllability matrix is full rank
29     P2 = np.array([ -3, -5]) # Define desired closed-loop poles
30     K2 = ctrl.place(A2, B2, P2) # Compute gain K of stabilizing static state-feedback control law
31     print("\nSystem 2 is stabilizable by a static state-feedback control law u = K*xp")
32     print("Stabilizing gain matrix K for System 2 with desired poles at {} is:".format(P2))
33     print(K2)
34 else:
35     print("\nSystem 2 is not stabilizable")
```

The output of the code is displayed in the right-hand pane:

```
✓ import numpy as np ...
... System 1 is not stabilizable
...
System 2 is stabilizable by a static state-feedback control law u = K*xp
Stabilizing gain matrix K for System 2 with desired poles at [-3 -5] is:
[[4. 6.]]
```